Logistic Regression

11/5/2020

Loading required libraries

```
library(cluster)
library(data.table)
library(magrittr)
library(stringr)
library(ggplot2)
library(knitr)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.2     v dplyr   1.0.2
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::between()   masks data.table::between()
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::set_names() masks magrittr::set_names()
## x purrr::transpose() masks data.table::transpose()
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(FactoMineR)
library(nFactors)
```

```
## Loading required package: lattice

##
## Attaching package: 'nFactors'

## The following object is masked from 'package:lattice':
##
##     parallel
```

```
library(GGally)
```

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

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

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

```
library(gvlma)
library(leaps)
library(relaimpo)
```

```
## Loading required package: boot

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

## The following object is masked from 'package:psych':
##
##     logit

## Loading required package: survey
```

```
## Loading required package: grid

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:boot':
##
##     aml

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart

## Loading required package: mitools

## This is the global version of package relaimpo.

## If you are a non-US user, a version with the interesting additional metric pmvd is available

## from Ulrike Groempings web site at prof.beuth-hochschule.de/groemping.
```

```r
library(cowplot)
library(regclass)
```

```
## Loading required package: bestglm

## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

##
## Attaching package: 'VGAM'
```

```
## The following object is masked from 'package:survey':
##
##     calibrate

## The following objects are masked from 'package:boot':
##
##     logit, simplex

## The following objects are masked from 'package:psych':
##
##     fisherz, logistic, logit

## The following object is masked from 'package:tidyr':
##
##     fill

## Loading required package: rpart

## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:psych':
##
##     outlier

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

## The following object is masked from 'package:ggplot2':
##
##     margin

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.

##
## Attaching package: 'regclass'

## The following object is masked from 'package:lattice':
##
##     qq
```

```r
library(e1071)
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:VGAM':
##
##     predictors

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

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
#library(FFally)
```

Data Loading

```r
Lending_Data <- read_csv('Lending_Data.csv')
```

```
## Parsed with column specification:
## cols(
##   member_id = col_character(),
##   loan_status = col_character(),
##   int_rate = col_character(),
##   Bin_int = col_double(),
##   dti = col_double(),
##   Bin_dti = col_double(),
##   Default_flag = col_double(),
##   No_of_Enquiry = col_double(),
##   enq_buckets = col_character(),
##   annual_inc = col_double(),
##   Income_bins = col_double(),
```

```
##   home_ownership = col_character(),
##   purpose = col_character(),
##   open_acc = col_double(),
##   emp_length = col_character(),
##   verification_status = col_character(),
##   delinq_2yrs = col_double(),
##   loan_amnt = col_double(),
##   Bins_loan_amt = col_double()
## )
```

```r
Lend = copy(Lending_Data)
Lend = setDT(Lend)
#view(Lend)
str(Lend)
```

```
## Classes 'data.table' and 'data.frame':   35808 obs. of  19 variables:
##  $ member_id          : chr  "LC1" "LC10" "LC100" "LC1000" ...
##  $ loan_status        : chr  "Charged Off" "Fully Paid" "Fully Paid" "Fully Paid" ...
##  $ int_rate           : chr  "11.71%" "15.96%" "10.65%" "12.69%" ...
##  $ Bin_int            : num  10 16 8 11 22 1 23 10 5 16 ...
##  $ dti                : num  1.06 2.61 11.34 14 13.01 ...
##  $ Bin_dti            : num  2 3 11 14 13 11 5 10 24 14 ...
##  $ Default_flag       : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ No_of_Enquiry      : num  0 1 1 1 0 0 3 0 1 2 ...
##  $ enq_buckets        : chr  "0" "1-4" "1-4" "1-4" ...
##  $ annual_inc         : num  110000 135000 75000 51000 41500 ...
##  $ Income_bins        : num  9 11 6 4 3 4 12 7 6 4 ...
##  $ home_ownership     : chr  "MORTGAGE" "RENT" "MORTGAGE" "RENT" ...
##  $ purpose            : chr  "credit_card" "other" "educational" "credit_card" ...
##  $ open_acc           : num  6 3 7 5 8 5 4 7 6 9 ...
##  $ emp_length         : chr  "LT 1year" "10+ years" "2 years" "1 year" ...
##  $ verification_status: chr  "Not Verified" "Source Verified" "Source Verified" "Source Verified" ..
##  $ delinq_2yrs        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ loan_amnt          : num  7000 2000 12000 9350 6000 ...
##  $ Bins_loan_amt      : num  6 2 10 8 5 8 5 10 2 8 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   member_id = col_character(),
##   ..   loan_status = col_character(),
##   ..   int_rate = col_character(),
##   ..   Bin_int = col_double(),
##   ..   dti = col_double(),
##   ..   Bin_dti = col_double(),
##   ..   Default_flag = col_double(),
##   ..   No_of_Enquiry = col_double(),
##   ..   enq_buckets = col_character(),
##   ..   annual_inc = col_double(),
##   ..   Income_bins = col_double(),
##   ..   home_ownership = col_character(),
##   ..   purpose = col_character(),
##   ..   open_acc = col_double(),
##   ..   emp_length = col_character(),
##   ..   verification_status = col_character(),
##   ..   delinq_2yrs = col_double(),
```

```
##   ..   loan_amnt = col_double(),
##   ..   Bins_loan_amt = col_double()
##   .. )
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
Logistic_training_final <- read_csv('Logistic_training_final.csv')
```

```
## Parsed with column specification:
## cols(
##   loan_status = col_double(),
##   roi = col_double(),
##   loan_amnt = col_double(),
##   inq_last_6mths = col_character(),
##   purpose = col_character(),
##   revol_util = col_double(),
##   Late_fee_bin = col_character(),
##   term = col_double(),
##   total_pymnt = col_double()
## )
```

```r
Lending_log = copy(Logistic_training_final)
Lending_log = setDT(Lending_log)

#view(Lend)
str(Lending_log)
```

```
## Classes 'data.table' and 'data.frame':   39786 obs. of  9 variables:
##  $ loan_status   : num  1 0 1 1 1 0 0 0 1 0 ...
##  $ roi           : num  0.12 0.15 0.08 0.22 0.17 0.06 0.08 0.12 0.11 0.16 ...
##  $ loan_amnt     : num  35000 9500 3800 12400 4000 ...
##  $ inq_last_6mths: chr  "zero" "one" "three" "three" ...
##  $ purpose       : chr  "small_business" "other" "car" "debt_consolidation" ...
##  $ revol_util    : num  0.06 0.85 0.39 0.78 0.83 0.36 0.58 0.49 0.69 0.27 ...
##  $ Late_fee_bin  : chr  "0" "0" "0" "0" ...
##  $ term          : num  60 36 36 60 60 36 36 60 36 60 ...
##  $ total_pymnt   : num  11602 9617 1064 2128 829 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   loan_status = col_double(),
##   ..   roi = col_double(),
##   ..   loan_amnt = col_double(),
##   ..   inq_last_6mths = col_character(),
##   ..   purpose = col_character(),
##   ..   revol_util = col_double(),
##   ..   Late_fee_bin = col_character(),
##   ..   term = col_double(),
##   ..   total_pymnt = col_double()
##   .. )
##  - attr(*, ".internal.selfref")=<externalptr>
```

Data Cleaning

```
Lend[, member_id := factor(member_id)]
Lend[, loan_status := factor(loan_status)]
Lend[, home_ownership := factor(home_ownership)]
Lend[, purpose := factor(purpose)]
Lend[, verification_status := factor(verification_status)]

Lend[, int_rate := gsub('[%]', '', int_rate)]
Lend[, int_rate := trimws(int_rate)]
Lend[, int_rate := suppressWarnings(as.numeric(int_rate))]

Lend[open_acc %in% c(1, 2, 3, 4, 5), 'x' := 'LT5']
Lend[open_acc %in% c(6, 7, 8, 9, 10), 'x' := '6-10']
Lend[open_acc %in% c(11, 12, 13, 14, 15), 'x' := '11-15']
Lend[open_acc > 15, 'x' := '15+']
Lend = Lend %>% rename(no_of_acct = x)
str(Lend)
```

```
## Classes 'data.table' and 'data.frame':   35808 obs. of  20 variables:
##  $ member_id          : Factor w/ 35808 levels "LC1","LC10","LC100",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ loan_status        : Factor w/ 2 levels "Charged Off",..: 1 2 2 2 2 2 2 2 2 2 ...
##  $ int_rate           : num  11.7 16 10.7 12.7 19.7 ...
##  $ Bin_int            : num  10 16 8 11 22 1 23 10 5 16 ...
##  $ dti                : num  1.06 2.61 11.34 14 13.01 ...
##  $ Bin_dti            : num  2 3 11 14 13 11 5 10 24 14 ...
##  $ Default_flag       : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ No_of_Enquiry      : num  0 1 1 1 0 0 3 0 1 2 ...
##  $ enq_buckets        : chr  "0" "1-4" "1-4" "1-4" ...
##  $ annual_inc         : num  110000 135000 75000 51000 41500 ...
##  $ Income_bins        : num  9 11 6 4 3 4 12 7 6 4 ...
##  $ home_ownership     : Factor w/ 5 levels "MORTGAGE","NONE",..: 1 5 1 5 1 1 1 5 5 1 ...
##  $ purpose            : Factor w/ 14 levels "car","credit_card",..: 2 10 4 2 3 3 8 2 10 3 ...
##  $ open_acc           : num  6 3 7 5 8 5 4 7 6 9 ...
##  $ emp_length         : chr  "LT 1year" "10+ years" "2 years" "1 year" ...
##  $ verification_status: Factor w/ 3 levels "Not Verified",..: 1 2 2 2 3 3 1 1 1 2 ...
##  $ delinq_2yrs        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ loan_amnt          : num  7000 2000 12000 9350 6000 ...
##  $ Bins_loan_amt      : num  6 2 10 8 5 8 5 10 2 8 ...
##  $ no_of_acct         : chr  "6-10" "LT5" "6-10" "LT5" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   member_id = col_character(),
##   ..   loan_status = col_character(),
##   ..   int_rate = col_character(),
##   ..   Bin_int = col_double(),
##   ..   dti = col_double(),
##   ..   Bin_dti = col_double(),
##   ..   Default_flag = col_double(),
##   ..   No_of_Enquiry = col_double(),
##   ..   enq_buckets = col_character(),
##   ..   annual_inc = col_double(),
##   ..   Income_bins = col_double(),
```

```
##   ..   home_ownership = col_character(),
##   ..   purpose = col_character(),
##   ..   open_acc = col_double(),
##   ..   emp_length = col_character(),
##   ..   verification_status = col_character(),
##   ..   delinq_2yrs = col_double(),
##   ..   loan_amnt = col_double(),
##   ..   Bins_loan_amt = col_double()
##   .. )
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "index")= int
##   ..- attr(*, "__open_acc")= int  75 113 157 195 377 382 458 611 628 642 ...
```

Data Splitting

```
#Training Testing

## 10% of the sample size
smp_size = floor(0.10 * nrow(Lend))

## set the seed to make our partition reproducible
set.seed(123)
train_ind = sample(seq_len(nrow(Lend)), size = smp_size)

train = Lend[train_ind, ]
test = Lend[-train_ind, ]
```

Logistic Regression

```
head(Lending_log) # you see data, but no column names
```

```
##    loan_status  roi loan_amnt inq_last_6mths            purpose revol_util
## 1:           1 0.12     35000           zero     small_business       0.06
## 2:           0 0.15      9500            one              other       0.85
## 3:           1 0.08      3800          three                car       0.39
## 4:           1 0.22     12400          three debt_consolidation       0.78
## 5:           1 0.17      4000           zero              other       0.83
## 6:           0 0.06      7500           zero            medical       0.36
##    Late_fee_bin term total_pymnt
## 1:            0   60   11601.600
## 2:            0   36    9616.540
## 3:            0   36    1064.070
## 4:            0   60    2127.630
## 5:            0   60     829.140
## 6:            0   36    7835.776
```

```
xtabs(~ loan_status + roi, data = Lending_log)
```

```
##            roi
## loan_status 0.05 0.06 0.07 0.08 0.09  0.1 0.11 0.12 0.13 0.14 0.15 0.16 0.17
##           0  553 1481 3087 3148 1396 3073 4480 2769 4032 2399 2133 2118 1149
##           1   20   51  185  238  124  364  611  477  734  491  498  572  381
##            roi
## loan_status 0.18 0.19  0.2 0.21 0.22 0.23 0.24 0.25
##           0  900  644  342  248  116   28   19    1
##           1  301  243  160  111   79   21    9    0
```

```
#xtabs(~ loan_status + loan_amnt, data = Lending_log)
xtabs(~ loan_status + inq_last_6mths, data = Lending_log)
```

```
##            inq_last_6mths
## loan_status eight  five  four   one seven   six three   two  zero
##           0    12   111   282  9975    25    48  2225  4403 17035
##           1     2    25    54  1864    11    16   561   835  2302
```

```
xtabs(~ loan_status + purpose, data = Lending_log)
```

```
##            purpose
## loan_status   car credit_card debt_consolidation educational home_improvement
##           0  1391        4589              15884         269             2634
##           1   160         548               2792          56              351
##            purpose
## loan_status house major_purchase medical moving other renewable_energy
##           0   323           1966     589    491  3364               84
##           1    59            222     106     92   637               19
##            purpose
## loan_status small_business vacation wedding
##           0           1352      328     852
##           1            479       53      96
```

```
xtabs(~ loan_status + revol_util, data = Lending_log)
```

```
##            revol_util
## loan_status    0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09  0.1 0.11 0.12
##           0 1054  368  300  287  282  295  316  331  336  371  301  278  300
##           1  188   34   30   26   28   32   25   28   28   22   21   29   39
##            revol_util
## loan_status 0.13 0.14 0.15 0.16 0.17 0.18 0.19  0.2 0.21 0.22 0.23 0.24 0.25
##           0  294  309  305  300  321  330  307  315  322  341  333  374  327
##           1   30   28   38   33   32   40   40   28   37   37   33   40   43
##            revol_util
## loan_status 0.26 0.27 0.28 0.29  0.3 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38
##           0  342  380  340  328  327  395  344  346  380  348  340  373  385
##           1   41   47   45   55   41   53   49   50   59   60   39   44   51
##            revol_util
## loan_status 0.39  0.4 0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49  0.5 0.51
##           0  351  366  373  370  351  365  372  380  368  371  400  386  362
##           1   60   55   55   54   57   56   65   63   65   64   58   67   49
##            revol_util
## loan_status 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59  0.6 0.61 0.62 0.63 0.64
```

```
##           0  386  320  387  415  376  365  391  341  357  373  360  381  334
##           1   69   54   66   55   65   70   73   64   77   55   59   61   66
##            revol_util
## loan_status 0.65 0.66 0.67 0.68 0.69  0.7 0.71 0.72 0.73 0.74 0.75 0.76 0.77
##           0  374  342  383  316  386  382  335  344  328  307  328  323  367
##           1   67   82   85   69   63   60   56   78   77   79   73   66   63
##            revol_util
## loan_status 0.78 0.79  0.8 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89  0.9
##           0  288  339  343  336  305  309  312  312  282  260  293  293  301
##           1   71   61   77   79   70   74   65   66   68   56   64   66   78
##            revol_util
## loan_status 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99    1
##           0  256  271  279  284  266  282  233  228  201  102
##           1   70   59   71   69   78   54   66   70   69   26
```

```
xtabs(~ loan_status + Late_fee_bin, data = Lending_log)
```

```
##            Late_fee_bin
## loan_status     0  0to1   GT1
##           0 32911     9  1196
##           1  4797     2   871
```

```
xtabs(~ loan_status + term, data = Lending_log)
```

```
##            term
## loan_status    36    60
##           0 25869  8247
##           1  3227  2443
```

```
#xtabs(~ loan_status + total_pymnt, data = Lending_log)
```

```
logistic_lend <- glm(loan_status ~ purpose, data = Lending_log)
summary(logistic_lend)
```

```
##
## Call:
## glm(formula = loan_status ~ purpose, data = Lending_log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.2616  -0.1495  -0.1495  -0.1067   0.8987
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.103159   0.008837  11.674  < 2e-16 ***
## purposecredit_card          0.003518   0.010083   0.349  0.72718
## purposedebt_consolidation   0.046337   0.009196   5.039 4.71e-07 ***
## purposeeducational          0.069148   0.021231   3.257  0.00113 **
## purposehome_improvement     0.014429   0.010893   1.325  0.18533
## purposehouse                0.051291   0.019878   2.580  0.00988 **
## purposemajor_purchase      -0.001697   0.011552  -0.147  0.88323
```

```
## purposemedical              0.049359   0.015886   3.107  0.00189 **
## purposemoving               0.054645   0.016907   3.232  0.00123 **
## purposeother                0.056051   0.010410   5.385 7.31e-08 ***
## purposerenewable_energy     0.081307   0.035412   2.296  0.02168 *
## purposesmall_business       0.158446   0.012010  13.193  < 2e-16 ***
## purposevacation             0.035948   0.019899   1.807  0.07085 .
## purposewedding             -0.001893   0.014347  -0.132  0.89501
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1211171)
##
##     Null deviance: 4862.0  on 39785  degrees of freedom
## Residual deviance: 4817.1  on 39772  degrees of freedom
## AIC: 28936
##
## Number of Fisher Scoring iterations: 2
```

```
ll_null <- logistic_lend$null.deviance / -2
ll_proposed <- logistic_lend$deviance / -2
```

```
ll_null
```

```
## [1] -2430.977
```

```
ll_proposed
```

```
## [1] -2408.535
```

McFadden's Pseudo R^2 = [LL(Null) - LL(Proposed)] / LL(Null)

```
(ll_null - ll_proposed) / ll_null
```

```
## [1] 0.009231585
```

chi-square value = 2*(LL(Proposed) - LL(Null)) p-value = 1 - pchisq(chi-square value, df = 2-1)

```
1 - pchisq(2 * (ll_proposed - ll_null), df = 1)
```

```
## [1] 2.091083e-11
```

```
1 - pchisq((logistic_lend$null.deviance - logistic_lend$deviance), df = 1)
```

```
## [1] 2.091083e-11
```

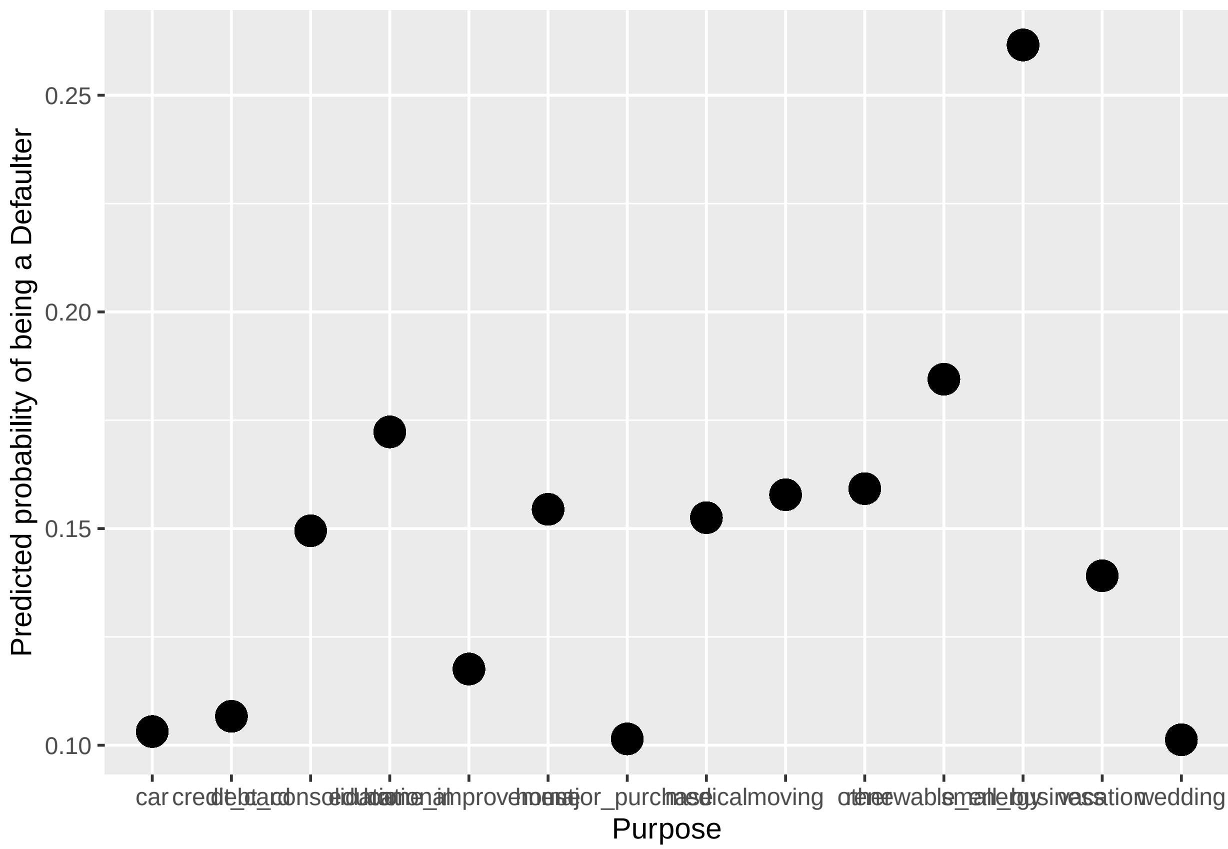
Lastly, let's see what this logistic regression predicts, given the interest rate (and no other data about them).

```r
predicted_data <- data.frame(probability_of_default = logistic_lend$fitted.values,
                             purpose = Lending_log$purpose)
head(predicted_data)
```

```
##   probability_of_default            purpose
## 1              0.2616057     small_business
## 2              0.1592102              other
## 3              0.1031593                car
## 4              0.1494967 debt_consolidation
## 5              0.1592102              other
## 6              0.1525180            medical
```

We can plot the data...

```r
ggplot(data = predicted_data,
       aes(x = purpose, y = probability_of_default)) +
  geom_point( size = 5) +
  xlab("Purpose") +
  ylab("Predicted probability of being a Defaulter")
```



Now we will use all of the data available to predict loan defaulters.

```r
str(Lending_log)
```

```
## Classes 'data.table' and 'data.frame':   39786 obs. of  9 variables:
##  $ loan_status   : num  1 0 1 1 1 0 0 0 1 0 ...
##  $ roi           : num  0.12 0.15 0.08 0.22 0.17 0.06 0.08 0.12 0.11 0.16 ...
##  $ loan_amnt     : num  35000 9500 3800 12400 4000 ...
##  $ inq_last_6mths: chr  "zero" "one" "three" "three" ...
##  $ purpose       : chr  "small_business" "other" "car" "debt_consolidation" ...
##  $ revol_util    : num  0.06 0.85 0.39 0.78 0.83 0.36 0.58 0.49 0.69 0.27 ...
##  $ Late_fee_bin  : chr  "0" "0" "0" "0" ...
##  $ term          : num  60 36 36 60 60 36 36 60 36 60 ...
##  $ total_pymnt   : num  11602 9617 1064 2128 829 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   loan_status = col_double(),
##   ..   roi = col_double(),
##   ..   loan_amnt = col_double(),
##   ..   inq_last_6mths = col_character(),
##   ..   purpose = col_character(),
##   ..   revol_util = col_double(),
##   ..   Late_fee_bin = col_character(),
##   ..   term = col_double(),
##   ..   total_pymnt = col_double()
##   .. )
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
logistic_complex <- glm(loan_status ~ roi + loan_amnt + inq_last_6mths + purpose +
                          revol_util + Late_fee_bin + term + total_pymnt,
                        data = Lending_log)
summary(logistic_complex)
```

```
##
## Call:
## glm(formula = loan_status ~ roi + loan_amnt + inq_last_6mths +
##     purpose + revol_util + Late_fee_bin + term + total_pymnt,
##     data = Lending_log)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.28208  -0.11625  -0.05176   0.00496   1.59360
##
## Coefficients:
##                             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)               -3.806e-01  7.032e-02   -5.413 6.25e-08 ***
## roi                        1.694e+00  4.759e-02   35.597  < 2e-16 ***
## loan_amnt                  5.206e-05  3.863e-07  134.761  < 2e-16 ***
## inq_last_6mthsfive         1.377e-01  7.323e-02    1.881 0.060029 .
## inq_last_6mthsfour         1.584e-01  7.118e-02    2.225 0.026116 *
## inq_last_6mthsone          1.492e-01  6.980e-02    2.138 0.032516 *
## inq_last_6mthsseven        2.124e-01  8.216e-02    2.585 0.009730 **
## inq_last_6mthssix          2.232e-01  7.697e-02    2.900 0.003733 **
## inq_last_6mthsthree        1.621e-01  6.992e-02    2.318 0.020469 *
## inq_last_6mthstwo          1.449e-01  6.984e-02    2.075 0.037974 *
## inq_last_6mthszero         1.425e-01  6.979e-02    2.041 0.041251 *
## purposecredit_card         6.954e-03  7.727e-03    0.900 0.368105
## purposedebt_consolidation  2.193e-02  7.068e-03    3.103 0.001916 **
```

```
## purposeeducational        4.022e-02  1.597e-02    2.518 0.011807 *
## purposehome_improvement    1.144e-02  8.237e-03    1.389 0.164857
## purposehouse               2.483e-02  1.498e-02    1.657 0.097559 .
## purposemajor_purchase      5.835e-04  8.688e-03    0.067 0.946451
## purposemedical             2.312e-02  1.193e-02    1.937 0.052693 .
## purposemoving              3.004e-02  1.270e-02    2.365 0.018051 *
## purposeother               2.800e-02  7.858e-03    3.563 0.000367 ***
## purposerenewable_energy    3.708e-02  2.655e-02    1.397 0.162538
## purposesmall_business      4.834e-02  9.166e-03    5.274 1.34e-07 ***
## purposevacation            2.862e-02  1.493e-02    1.916 0.055310 .
## purposewedding            -4.188e-03  1.081e-02   -0.388 0.698367
## revol_util                 3.968e-02  5.533e-03    7.171 7.57e-13 ***
## Late_fee_bin0to1           7.492e-02  7.866e-02    0.952 0.340887
## Late_fee_binGT1            2.051e-01  5.943e-03   34.519  < 2e-16 ***
## term                       3.537e-03  1.458e-04   24.255  < 2e-16 ***
## total_pymnt               -4.992e-05  3.100e-07 -161.005  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.06800793)
##
##     Null deviance: 4862.0  on 39785  degrees of freedom
## Residual deviance: 2703.8  on 39757  degrees of freedom
## AIC: 5988.8
##
## Number of Fisher Scoring iterations: 2
```

Now calculate the overall "Pseudo R-squared" and its p-value

```
ll.null <- logistic_complex$null.deviance / -2
ll.proposed <- logistic_complex$deviance / -2
```

McFadden's Pseudo R^2 = [LL(Null) - LL(Proposed)] / LL(Null)

```
(ll.null - ll.proposed) / ll.null
```

```
## [1] 0.443888
```

The p-value for the R^2

```
1 - pchisq(2 * (ll.proposed - ll.null),
           df = (length(logistic_complex$coefficients) - 1))
```

```
## [1] 0
```

now we can plot the data

```
predicted.data <- data.frame(probability.of.lend = logistic_complex$fitted.values,
                             Default = Lending_log$loan_status)
summary(predicted.data)
```

```
##  probability.of.lend    Default
##  Min.   :-0.70075    Min.   :0.0000
##  1st Qu.: 0.02650    1st Qu.:0.0000
##  Median : 0.08248    Median :0.0000
##  Mean   : 0.14251    Mean   :0.1425
##  3rd Qu.: 0.17107    3rd Qu.:0.0000
##  Max.   : 2.17593    Max.   :1.0000
```

```r
str(Lend)
```

```
## Classes 'data.table' and 'data.frame':   35808 obs. of  20 variables:
##  $ member_id          : Factor w/ 35808 levels "LC1","LC10","LC100",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ loan_status        : Factor w/ 2 levels "Charged Off",..: 1 2 2 2 2 2 2 2 2 2 ...
##  $ int_rate           : num  11.7 16 10.7 12.7 19.7 ...
##  $ Bin_int            : num  10 16 8 11 22 1 23 10 5 16 ...
##  $ dti                : num  1.06 2.61 11.34 14 13.01 ...
##  $ Bin_dti            : num  2 3 11 14 13 11 5 10 24 14 ...
##  $ Default_flag       : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ No_of_Enquiry      : num  0 1 1 1 0 0 3 0 1 2 ...
##  $ enq_buckets        : chr  "0" "1-4" "1-4" "1-4" ...
##  $ annual_inc         : num  110000 135000 75000 51000 41500 ...
##  $ Income_bins        : num  9 11 6 4 3 4 12 7 6 4 ...
##  $ home_ownership     : Factor w/ 5 levels "MORTGAGE","NONE",..: 1 5 1 5 1 1 1 5 5 1 ...
##  $ purpose            : Factor w/ 14 levels "car","credit_card",..: 2 10 4 2 3 3 8 2 10 3 ...
##  $ open_acc           : num  6 3 7 5 8 5 4 7 6 9 ...
##  $ emp_length         : chr  "LT 1year" "10+ years" "2 years" "1 year" ...
##  $ verification_status: Factor w/ 3 levels "Not Verified",..: 1 2 2 2 3 3 1 1 1 2 ...
##  $ delinq_2yrs        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ loan_amnt          : num  7000 2000 12000 9350 6000 ...
##  $ Bins_loan_amt      : num  6 2 10 8 5 8 5 10 2 8 ...
##  $ no_of_acct         : chr  "6-10" "LT5" "6-10" "LT5" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   member_id = col_character(),
##   ..   loan_status = col_character(),
##   ..   int_rate = col_character(),
##   ..   Bin_int = col_double(),
##   ..   dti = col_double(),
##   ..   Bin_dti = col_double(),
##   ..   Default_flag = col_double(),
##   ..   No_of_Enquiry = col_double(),
##   ..   enq_buckets = col_character(),
##   ..   annual_inc = col_double(),
##   ..   Income_bins = col_double(),
##   ..   home_ownership = col_character(),
##   ..   purpose = col_character(),
##   ..   open_acc = col_double(),
##   ..   emp_length = col_character(),
##   ..   verification_status = col_character(),
##   ..   delinq_2yrs = col_double(),
##   ..   loan_amnt = col_double(),
##   ..   Bins_loan_amt = col_double()
##   .. )
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
##  - attr(*, "index")= int
##   ..- attr(*, "__open_acc")= int  75 113 157 195 377 382 458 611 628 642 ...
```

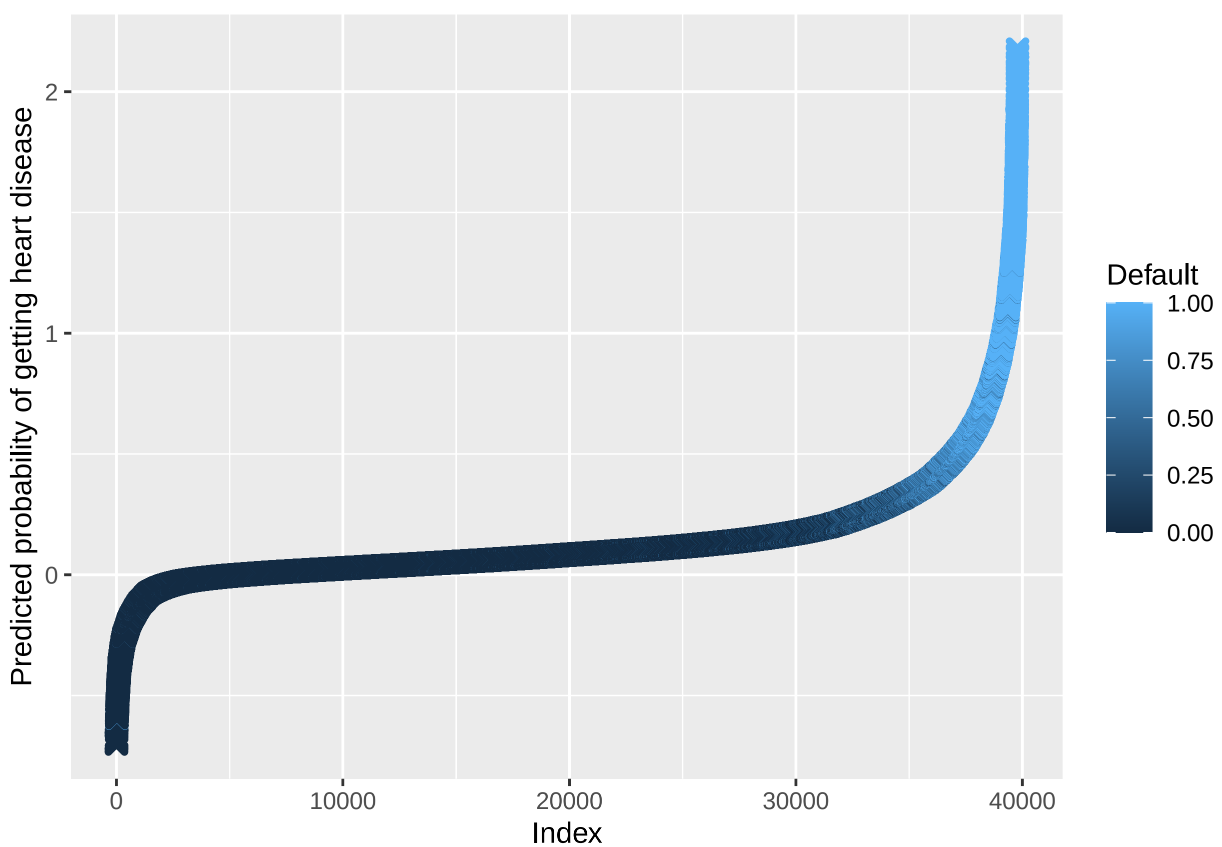
```
NROW(predicted.data)
```

```
## [1] 39786
```

```
predicted.data <- predicted.data[order(predicted.data$probability.of.lend,
                                       decreasing = FALSE),]
predicted.data$rank <- 1:nrow(predicted.data)
```

Lastly, we can plot the predicted probabilities for each sample having

```
## loan default and color by whether or not they actually had defaulted
ggplot(data = predicted.data,
       aes(x = rank, y = probability.of.lend)) +
  geom_point(aes(color = Default), alpha = 1, shape = 4, stroke = 2) +
  xlab("Index") +
  ylab("Predicted probability of getting heart disease")
```



Few packages for confusion matrix. Lets look at them one by one

```
NROW(logistic_complex$fitted.values)
```

```
## [1] 39786
```

```r
confusion_matrix(logistic_complex)
```

```
##          Predicted 0 Predicted 1 Total
## Actual 0       33821         295 34116
## Actual 1        3267        2403  5670
## Total          37088        2698 39786
```

```r
pdata <- predict(logistic_complex, newdata = Lending_log, type = "response" )
head(pdata)
```

```
##          1          2          3          4          5          6
## 1.47092341 0.22631842 0.20449263 0.95855879 0.48984490 0.02751662
```

```r
summary(pdata)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.70075  0.02650  0.08248  0.14251  0.17107  2.17593
```

```r
pdataF <- as.factor(ifelse(test = as.numeric(pdata > 0.5) == 0,
                           yes = 0, no = 1))
NROW(pdataF)
```

```
## [1] 39786
```

```r
NROW(Lending_log$loan_status)
```

```
## [1] 39786
```

```r
head(pdataF)
```

```
## [1] 1 0 0 1 0 0
## Levels: 0 1
```

```r
head(Lending_log$loan_status)
```

```
## [1] 1 0 1 1 1 0
```

```r
Lending_log[, loan_status := factor(loan_status)]
confusionMatrix(pdataF, Lending_log$loan_status)
```

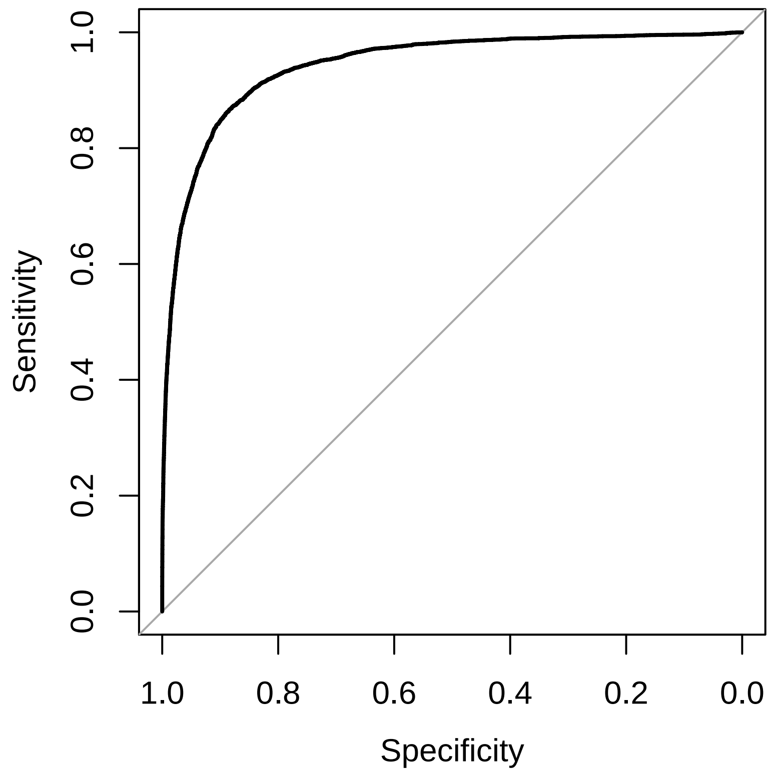
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 33821  3267
##          1   295  2403
##
```

```
##                Accuracy : 0.9105
##                  95% CI : (0.9076, 0.9133)
##     No Information Rate : 0.8575
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5313
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9914
##             Specificity : 0.4238
##          Pos Pred Value : 0.9119
##          Neg Pred Value : 0.8907
##              Prevalence : 0.8575
##          Detection Rate : 0.8501
##    Detection Prevalence : 0.9322
##       Balanced Accuracy : 0.7076
##
##        'Positive' Class : 0
##
```

```r
par(pty = "s")
roc(Lending_log$loan_status, logistic_complex$fitted.values, plot = TRUE)
```

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

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




```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     plot
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 0.9425
```

NOTE: By default, roc() uses specificity on the x-axis and the values rangefrom 1 to 0. This makes the graph look like what we would expect, but the x-axis itself might induce a headache. To use 1-specificity (i.e. the False Positive Rate) on the x-axis, set "legacy.axes" to TRUE.

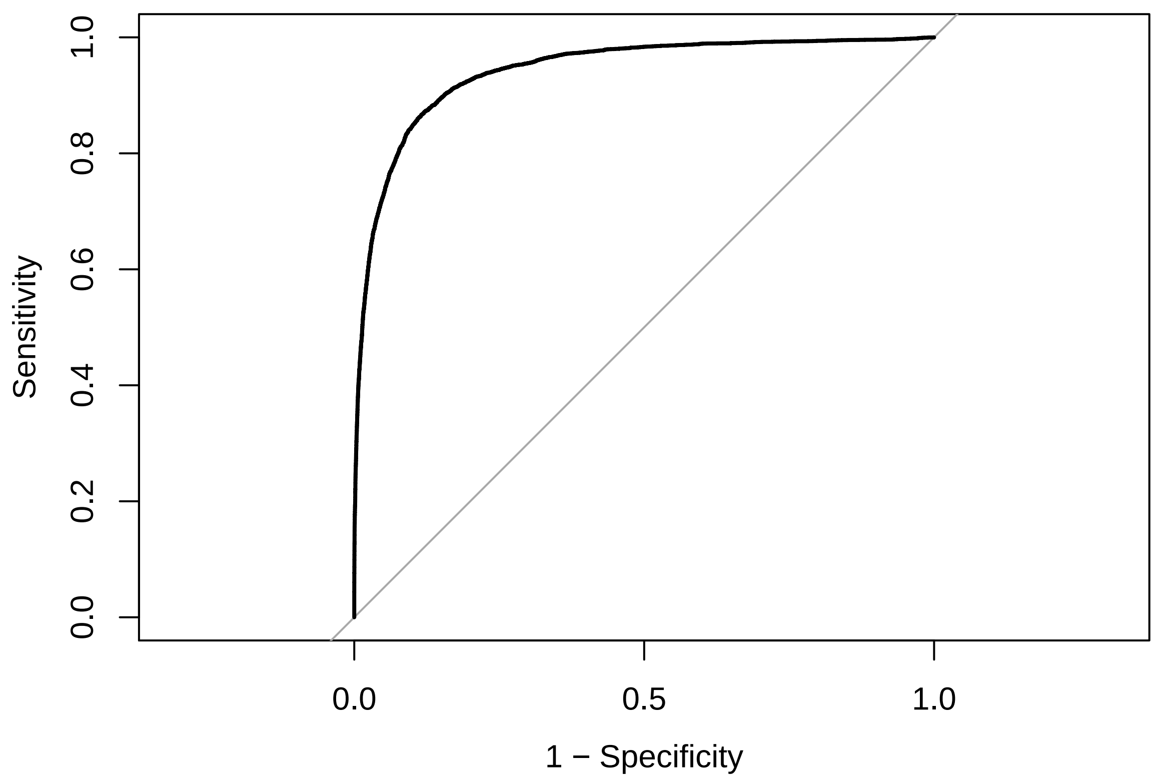
```r
head(logistic_complex$fitted.values)
```

```
##          1          2          3          4          5          6
## 1.47092341 0.22631842 0.20449263 0.95855879 0.48984490 0.02751662
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE)
```

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

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

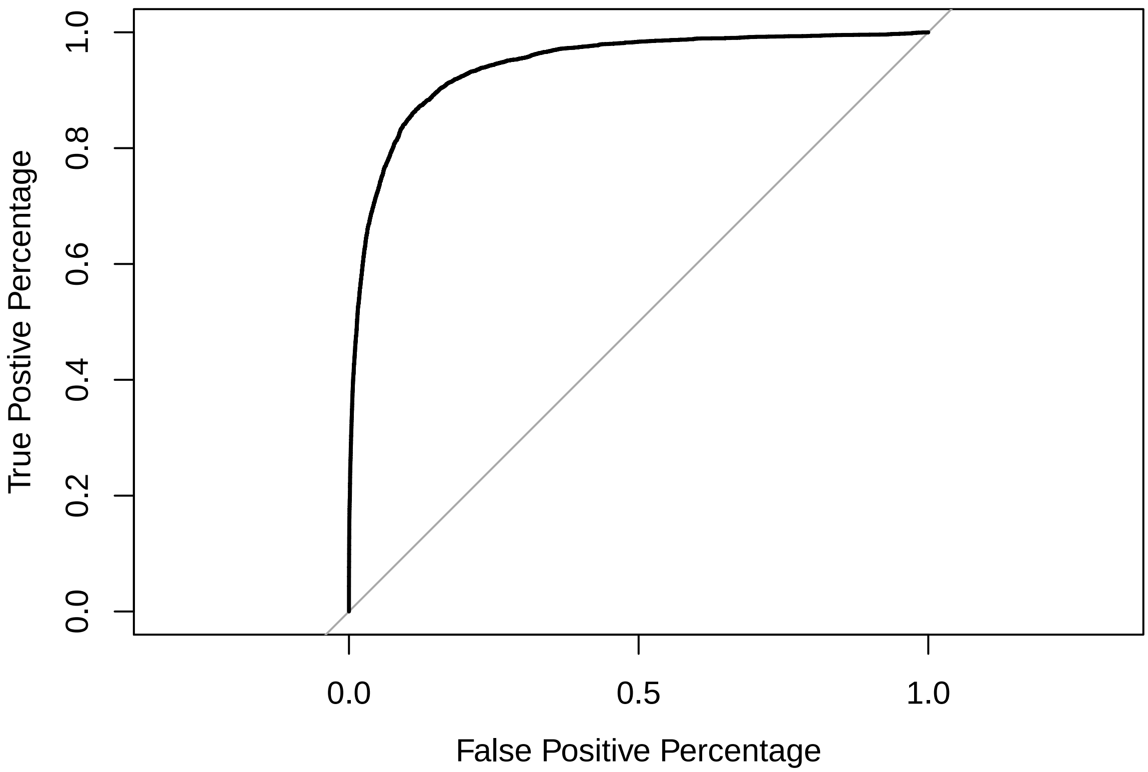



```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     plot
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 0.9425
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab ="True Postive Percentage")
```

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

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

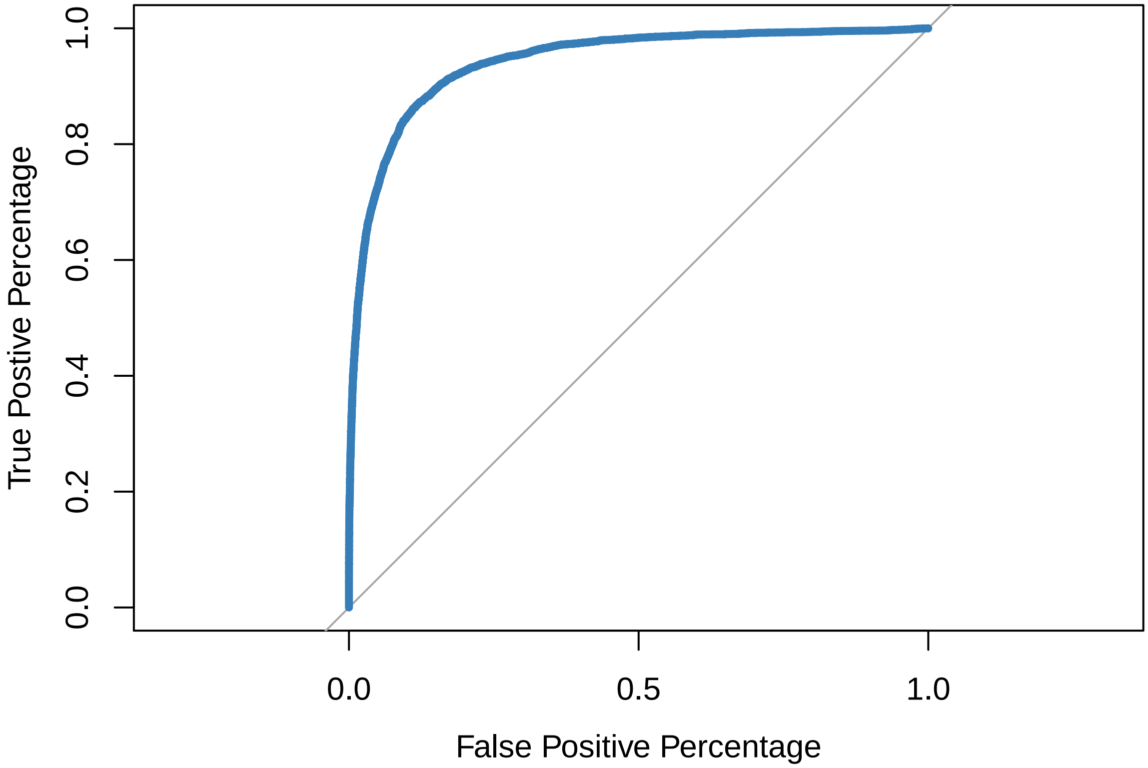



```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     plot
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 0.9425
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8", lwd = 4)
```

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

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

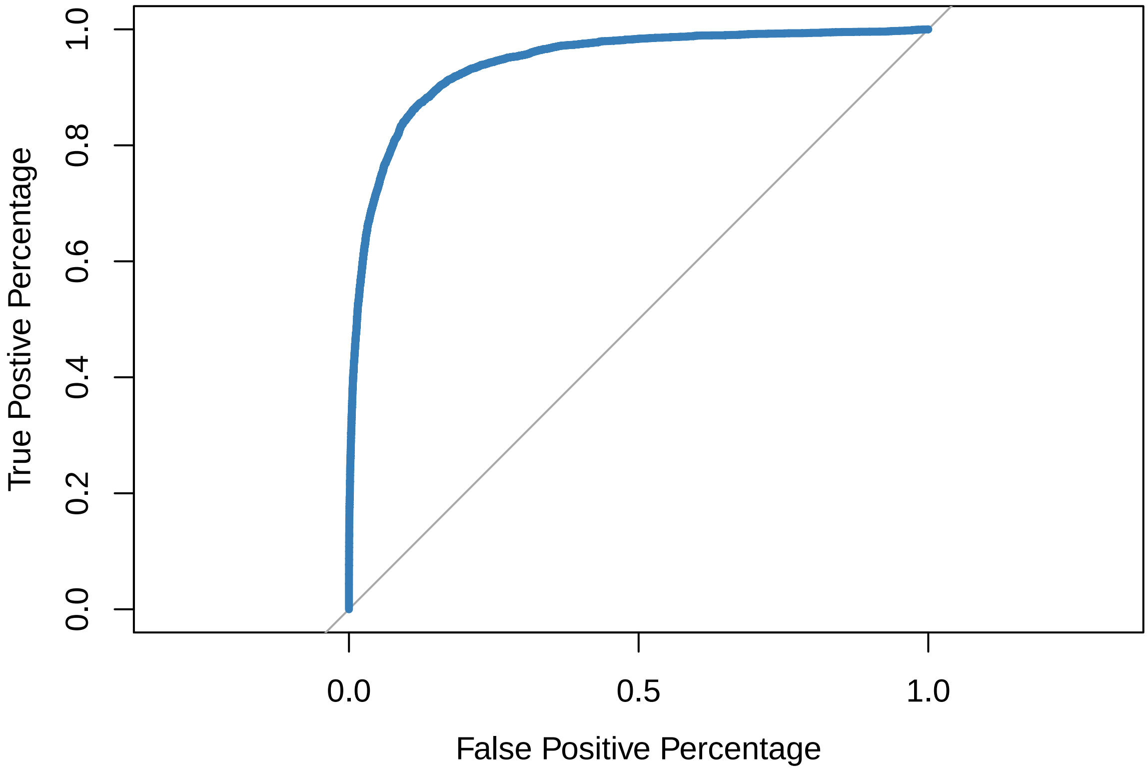



```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     plot
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 0.9425
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8",
    lwd = 4)
```

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

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




```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     plot
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 0.9425
```

If we want to find out the optimal threshold we can store the data used to make the ROC graph in a variable...

```
roc.info <- roc(Lending_log$loan_status,
                logistic_complex$fitted.values,
                legacy.axes = TRUE)
```

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

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

```
str(roc.info)
```

```
## List of 15
##  $ percent            : logi FALSE
##  $ sensitivities      : num [1:39781] 1 1 1 1 1 1 1 1 1 1 ...
##  $ specificities      : num [1:39781] 0.00 2.93e-05 5.86e-05 8.79e-05 1.17e-04 ...
```

```
##  $ thresholds        : num [1:39781] -Inf -0.699 -0.693 -0.688 -0.686 ...
##  $ direction         : chr "<"
##  $ cases             : Named num [1:5670] 1.471 0.204 0.959 0.49 0.614 ...
##   ..- attr(*, "names")= chr [1:5670] "1" "3" "4" "5" ...
##  $ controls          : Named num [1:34116] 0.2263 0.0275 0.0658 -0.0141 -0.2758 ...
##   ..- attr(*, "names")= chr [1:34116] "2" "6" "7" "8" ...
##  $ fun.sesp          :function (thresholds, controls, cases, direction)
##  $ auc               : 'auc' num 0.943
##   ..- attr(*, "partial.auc")= logi FALSE
##   ..- attr(*, "percent")= logi FALSE
##   ..- attr(*, "roc")=List of 15
##   .. ..$ percent           : logi FALSE
##   .. ..$ sensitivities     : num [1:39781] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..$ specificities     : num [1:39781] 0.00 2.93e-05 5.86e-05 8.79e-05 1.17e-04 ...
##   .. ..$ thresholds        : num [1:39781] -Inf -0.699 -0.693 -0.688 -0.686 ...
##   .. ..$ direction         : chr "<"
##   .. ..$ cases             : Named num [1:5670] 1.471 0.204 0.959 0.49 0.614 ...
##   .. .. ..- attr(*, "names")= chr [1:5670] "1" "3" "4" "5" ...
##   .. ..$ controls          : Named num [1:34116] 0.2263 0.0275 0.0658 -0.0141 -0.2758 ...
##   .. .. ..- attr(*, "names")= chr [1:34116] "2" "6" "7" "8" ...
##   .. ..$ fun.sesp          :function (thresholds, controls, cases, direction)
##   .. ..$ auc               : 'auc' num 0.943
##   .. .. ..- attr(*, "partial.auc")= logi FALSE
##   .. .. ..- attr(*, "percent")= logi FALSE
##   .. .. ..- attr(*, "roc")=List of 8
##   .. .. .. ..$ percent      : logi FALSE
##   .. .. .. ..$ sensitivities: num [1:39781] 1 1 1 1 1 1 1 1 1 1 ...
##   .. .. .. ..$ specificities: num [1:39781] 0.00 2.93e-05 5.86e-05 8.79e-05 1.17e-04 ...
##   .. .. .. ..$ thresholds   : num [1:39781] -Inf -0.699 -0.693 -0.688 -0.686 ...
##   .. .. .. ..$ direction    : chr "<"
##   .. .. .. ..$ cases        : Named num [1:5670] 1.471 0.204 0.959 0.49 0.614 ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:5670] "1" "3" "4" "5" ...
##   .. .. .. ..$ controls     : Named num [1:34116] 0.2263 0.0275 0.0658 -0.0141 -0.2758 ...
##   .. .. .. .. ..- attr(*, "names")= chr [1:34116] "2" "6" "7" "8" ...
##   .. .. .. ..$ fun.sesp     :function (thresholds, controls, cases, direction)
##   .. .. .. ..- attr(*, "class")= chr "roc"
##   .. ..$ call              : language roc.default(response = Lending_log$loan_status, predictor = log
##   .. ..$ original.predictor: Named num [1:39786] 1.471 0.226 0.204 0.959 0.49 ...
##   .. .. ..- attr(*, "names")= chr [1:39786] "1" "2" "3" "4" ...
##   .. ..$ original.response : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 2 1 ...
##   .. ..$ predictor         : Named num [1:39786] 1.471 0.226 0.204 0.959 0.49 ...
##   .. .. ..- attr(*, "names")= chr [1:39786] "1" "2" "3" "4" ...
##   .. ..$ response          : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 2 1 ...
##   .. ..$ levels            : chr [1:2] "0" "1"
##   .. ..- attr(*, "class")= chr "roc"
##  $ call              : language roc.default(response = Lending_log$loan_status, predictor = logistic
##  $ original.predictor: Named num [1:39786] 1.471 0.226 0.204 0.959 0.49 ...
##   ..- attr(*, "names")= chr [1:39786] "1" "2" "3" "4" ...
##  $ original.response : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 2 1 ...
##  $ predictor         : Named num [1:39786] 1.471 0.226 0.204 0.959 0.49 ...
##   ..- attr(*, "names")= chr [1:39786] "1" "2" "3" "4" ...
##  $ response          : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 1 1 2 1 ...
##  $ levels            : chr [1:2] "0" "1"
##  - attr(*, "class")= chr "roc"
```

tpp = true positive percentage fpp = false positive precentage

```
roc.df <- data.frame(tpp = roc.info$sensitivities*100,
                     fpp = (1 - roc.info$specificities)*100,
                     thresholds = roc.info$thresholds)
```

fpp = false positive precentage

```
#roc.df
```

```
head(roc.df)
```

```
##   tpp       fpp thresholds
## 1 100 100.00000       -Inf
## 2 100  99.99707 -0.6985952
## 3 100  99.99414 -0.6929406
## 4 100  99.99121 -0.6879148
## 5 100  99.98828 -0.6862639
## 6 100  99.98534 -0.6839933
```

head() will show us the values for the upper right-hand corner of the ROC graph, when the threshold is so low (negative infinity) that every single sample is called "obese".

Thus TPP = 100% and FPP = 100%

```
tail(roc.df)
```

```
##              tpp fpp thresholds
## 39776 0.08818342   0   2.101262
## 39777 0.07054674   0   2.117864
## 39778 0.05291005   0   2.135212
## 39779 0.03527337   0   2.149559
## 39780 0.01763668   0   2.164533
## 39781 0.00000000   0        Inf
```

tail() will show us the values for the lower left-hand corner
of the ROC graph, when the threshold is so high (infinity)

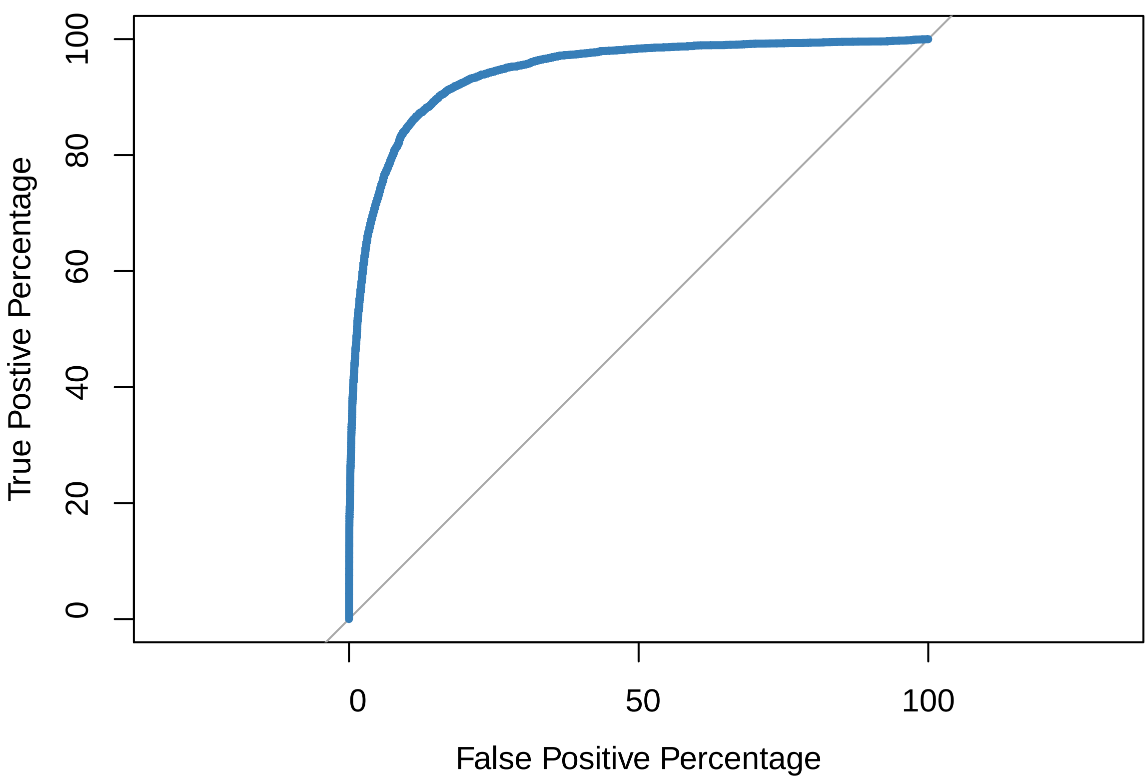
that every single sample is called "not obese". Thus, TPP = 0% and FPP = 0% now let's look at the thresholds between TPP 60% and 80%

```
head(roc.df[roc.df$tpp > 60 & roc.df$tpp < 80, ])
```

```
##            tpp      fpp thresholds
## 32669 79.98236 7.553641  0.2391854
## 32670 79.96473 7.553641  0.2392082
## 32671 79.96473 7.550709  0.2392261
## 32672 79.96473 7.547778  0.2392271
## 32673 79.94709 7.547778  0.2392481
## 32674 79.92945 7.547778  0.2392768
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8", lwd = 4, percent = TRUE)
```

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

## Setting direction: controls < cases
```

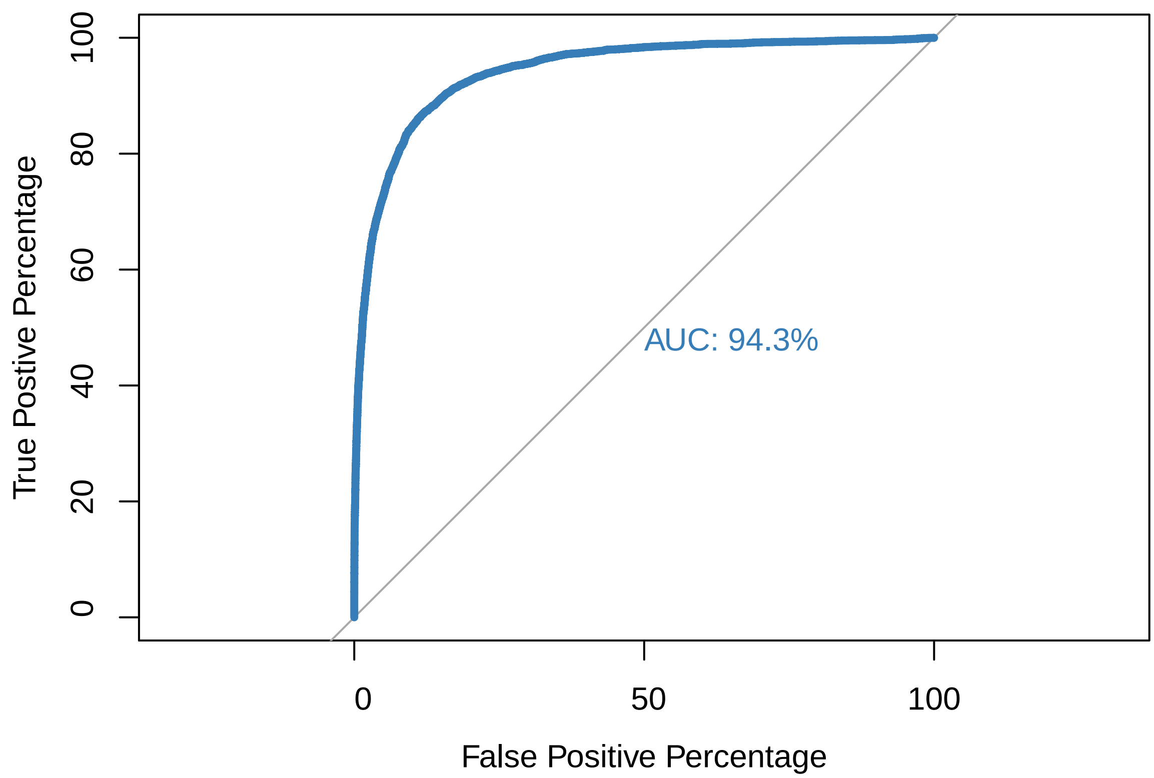



```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     perc
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 94.25%
```

```r
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab="False Positive Percentage",
```

```
    ylab="True Postive Percentage",
    col="#377eb8",
    lwd=4,
    percent=TRUE,
    print.auc=TRUE)
```

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

## Setting direction: controls < cases
```

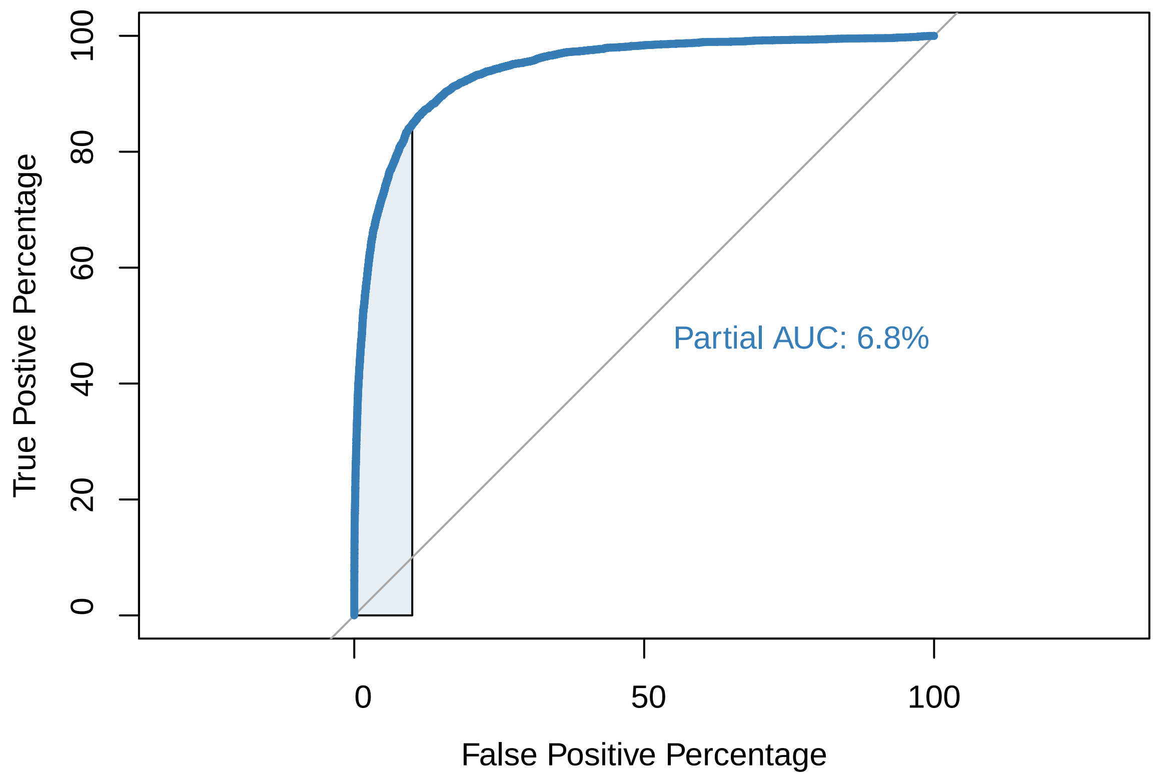



```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     perc
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Len
## Area under the curve: 94.25%
```

```
roc(Lending_log$loan_status,
    logistic_complex$fitted.values,
    plot = TRUE,
    legacy.axes = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8",
```

```
    lwd = 4,
    percent = TRUE,
    print.auc = TRUE,
    partial.auc = c(100, 90),
    auc.polygon = TRUE,
    auc.polygon.col = "#377eb822",
    print.auc.x = 45)
```

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

## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_complex$fitted.values,     perc
##
## Data: logistic_complex$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Lend
## Partial area under the curve (specificity 100%-90%): 6.801%
```

Lets do two roc plots to understand which model is better

```
roc(Lending_log$loan_status,
    logistic_lend$fitted.values,
```

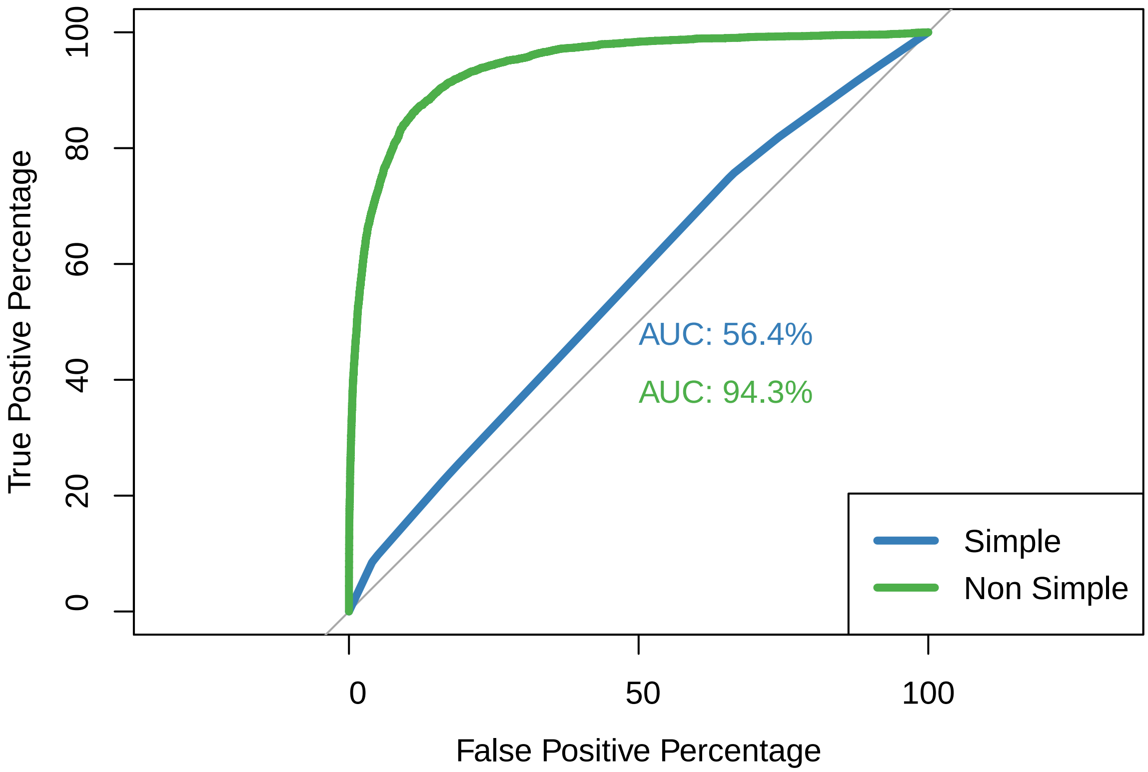
```
    plot = TRUE,
    legacy.axes = TRUE,
    percent = TRUE,
    xlab = "False Positive Percentage",
    ylab = "True Postive Percentage",
    col = "#377eb8",
    lwd = 4,
    print.auc = TRUE)
```

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

## Setting direction: controls < cases

##
## Call:
## roc.default(response = Lending_log$loan_status, predictor = logistic_lend$fitted.values,     percent
##
## Data: logistic_lend$fitted.values in 34116 controls (Lending_log$loan_status 0) < 5670 cases (Lendin
## Area under the curve: 56.39%
```

```
#Lets add the other graph

plot.roc(Lending_log$loan_status,
         logistic_complex$fitted.values,
         percent=TRUE,
         col="#4daf4a",
         lwd = 4,
         print.auc = TRUE,
         add = TRUE,
         print.auc.y = 40)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
legend("bottomright",
       legend = c("Simple", "Non Simple"),
       col = c("#377eb8", "#4daf4a"),
       lwd = 4) # Make it user friendly
```



Reset the par area back to the default setting

```
par(pty = "m")
```

Conclusion:

We can conclude that our complex model can predit loan defaulters with a good accuracy of 94.3% as compared to a simple model using the loan purpose which had an accuracy of 56.4%. Though not forgetting that complex models dont work in every situation. We must always keep our model as parsimonious as possible. To make our model parsimonious we got rid of many features which were not useful using techniques like factor analysi and clustering.