Multinomial Logistic Regression

xtabs(~ loan_status + grade, data=loan)

```
             grade
loan_status     1     2     3     4     5     6     7
          0   571  2026  4763  9210 13313  7995  1701
          1   397  1630  5268 14051 31392 33859 16125
```

logistic_simple <- glm(loan_status ~ grade, data=loan, family="binomial")
summary(logistic_simple)

```
Call:
glm(formula = loan_status ~ grade, family = "binomial", data = loan)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0224  -1.1812   0.6544   0.8060   1.5935

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.414218   0.024964  -56.65   <2e-16 ***
grade        0.474403   0.004965   95.55   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 168252  on 142300  degrees of freedom
Residual deviance: 158411  on 142299  degrees of freedom
AIC: 158415

Number of Fisher Scoring iterations: 4
```

xtabs(~ loan_status + purpose, data=loan)
logistic_simple <- glm(loan_status ~ purpose, data=loan, family="binomial")
summary(logistic_simple)

Now calculate the overall "Pseudo R-squared" and its p-value
NOTE: Since we are doing logistic regression...
Null devaiance = 2*(0 - LogLikelihood(null model))
= -2*LogLikihood(null model)
Residual deviance = 2*(0 - LogLikelihood(proposed model))
= -2*LogLikelihood(proposed model)
ll.null <- logistic_simple$null.deviance/-2
ll.proposed <- logistic_simple$deviance/-2
ll.null
ll.proposed

```
> ll.null <- logistic_simple$null.deviance/-2
> ll.proposed <- logistic_simple$deviance/-2
> ll.null
[1] -84126.08
> ll.proposed
[1] -84125.3
```

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

```
> ## McFadden's Pseudo R^2 = [ LL(Null) - LL(Proposed) ] / LL(Null)
> (ll.null - ll.proposed) / ll.null
[1] 9.261561e-06
```

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

```
> 1 - pchisq(2*(ll.proposed - ll.null), df=1)
[1] 0.2119176
> 1 - pchisq((logistic_simple$null.deviance - logistic_simple$deviance), df=1)
[1] 0.2119176
```

Lastly, let's see what this logistic regression predicts, given
predicted.data <-
data.frame(probability.of.paying.loan=logistic_simple$fitted.values,grade=loan$grade)
predicted.data

```
   probability.of.paying.loan grade
1                   0.7210518     4
3                   0.7210518     7
4                   0.7210518     6
8                   0.7245667     5
9                   0.7245667     4
11                  0.7210518     3
14                  0.7210518     7
18                  0.7245667     6
19                  0.7245667     6
20                  0.7245667     5
22                  0.7210518     6
23                  0.7210518     7
24                  0.7210518     7
33                  0.7210518     5
40                  0.7245667     7
45                  0.7210518     6
48                  0.7245667     4
```

We can plot the data...
xtabs(~ probability.of.paying.loan + grade, data=predicted.data)

```
                              grade
probability.of.paying.loan     1     2     3     4     5     6     7
         0.721051813802562   733  2833  7572 17448 34053 32937 13835
         0.724566737609738   235   823  2459  5813 10652  8917  3991
```

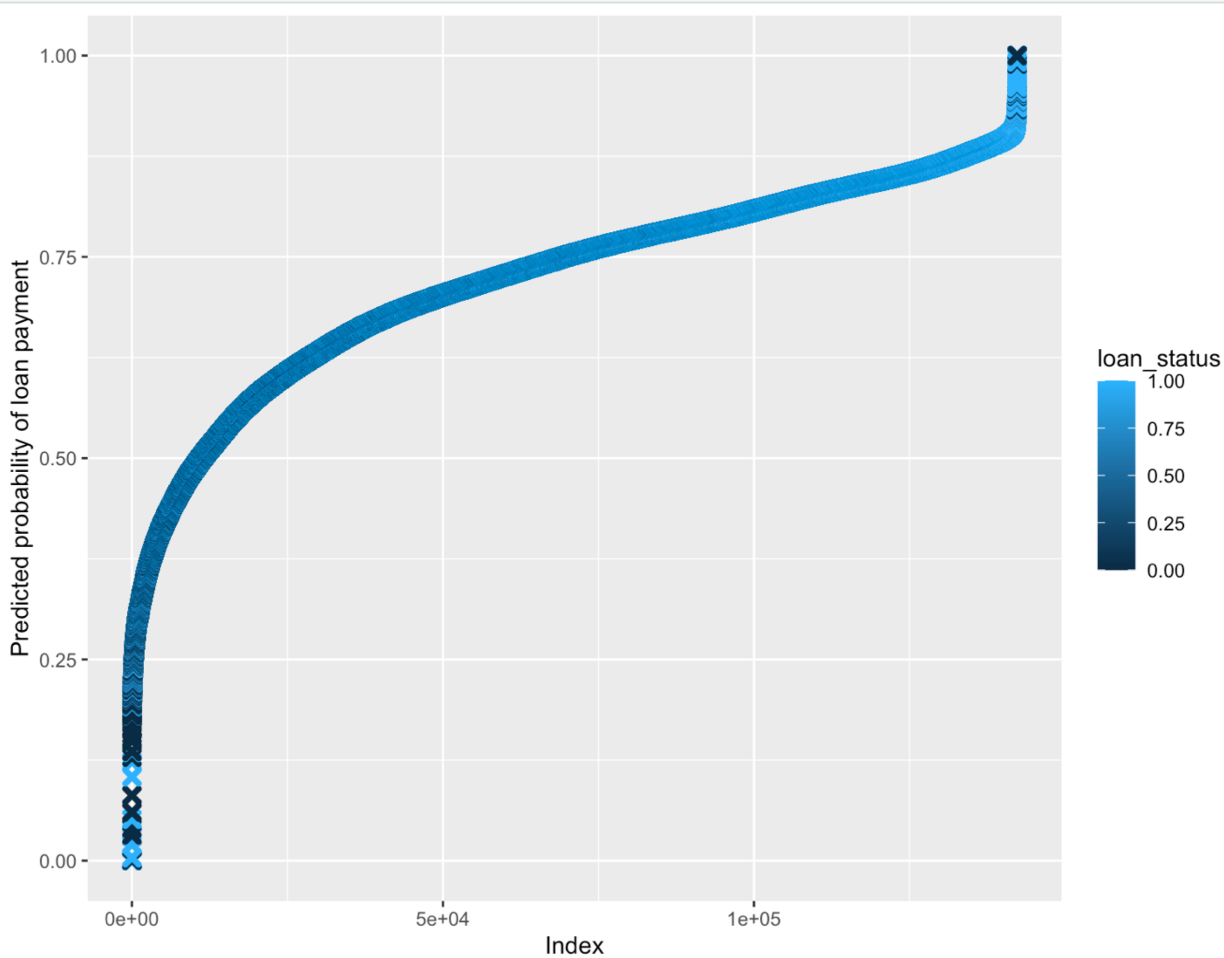
```
logistic <- glm(loan_status ~ ., data=loan, family="binomial")
summary(logistic)
```

```
Call:
glm(formula = loan_status ~ ., family = "binomial", data = loan)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-5.6240  -1.0923   0.6299   0.7984   4.4806  

Coefficients: (1 not defined because of singularities)
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)     1.196e+02  1.710e+01   6.994 2.67e-12 ***
loan_amnt      -1.125e-05  5.261e-06  -2.138   0.0325 *  
funded_amnt            NA         NA      NA       NA    
installment    -1.287e-04  1.615e-04  -0.797   0.4253    
int_rate       -4.807e-02  4.701e-03 -10.225  < 2e-16 ***
issue_d        -5.907e-02  8.507e-03  -6.944 3.82e-12 ***
grade           1.978e-01  1.810e-02  10.928  < 2e-16 ***
purpose        -6.204e-02  1.514e-02  -4.099 4.15e-05 ***
dti            -1.064e-02  7.020e-04 -15.151  < 2e-16 ***
emp_length      2.824e-03  1.748e-03   1.616   0.1062    
home_ownership  3.775e-01  1.315e-02  28.710  < 2e-16 ***
annual_inc      1.465e-06  1.481e-07   9.893  < 2e-16 ***
term            2.752e-01  3.603e-02   7.637 2.23e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 168252  on 142300  degrees of freedom
Residual deviance: 156276  on 142289  degrees of freedom
AIC: 156300

Number of Fisher Scoring iterations: 4
```

```
ll.null <- logistic$null.deviance/-2
ll.proposed <- logistic$deviance/-2
## McFadden's Pseudo R^2 = [ LL(Null) - LL(Proposed) ] / LL(Null)
(ll.null - ll.proposed) / ll.null
## The p-value for the R^2
1 - pchisq(2*(ll.proposed - ll.null), df=(length(logistic$coefficients)-1))
```

```
> (ll.null - ll.proposed) / ll.null
[1] 0.07117708
> ## The p-value for the R^2
> 1 - pchisq(2*(ll.proposed - ll.null), df=(length(logistic$coefficients)-1))
[1] 0
```

```
## now we can plot the data
predicted.data <-
data.frame(probability.of.paying.loan=logistic$fitted.values,loan_status=loan$loan_status)
predicted.data <- predicted.data[order(predicted.data$probability.of.paying.loan,
decreasing=FALSE),]
predicted.data$rank <- 1:nrow(predicted.data)

 ## Lastly, we can plot the predicted probabilities for each sample paying
## loan and color by whether or not they actually pay the loan or not
ggplot(data=predicted.data, aes(x=rank, y=probability.of.paying.loan)) +
  geom_point(aes(color=loan_status), alpha=1, shape=4, stroke=2) +
  xlab("Index") +
  ylab("Predicted probability of loan payment")
```



```
# Few packages for confusion matrix. Lets look at them one by one
install.packages("regclass")
library(regclass)
confusion_matrix(logistic)
library(caret)

pdata <- predict(logistic,newdata=loan,type="response")
pdata
```

```
> pdata
        1         3         4         8         9        11        14        18        19        20        22 
0.5372256 0.8545863 0.7879645 0.7016461 0.6861068 0.5078753 0.8835897 0.7550511 0.7525959 0.6836958 0.7694102 
       23        24        33        40        45        48        50        56        61        65        68 
0.8612741 0.8198593 0.7122294 0.8532080 0.8416768 0.5425573 0.5804803 0.4736943 0.8338780 0.8235543 0.6657863 
       69        71        76        78        82        83        91        92        93        94        96 
0.5568653 0.7259451 0.5387195 0.8751485 0.4791480 0.6213514 0.5677197 0.8363989 0.6591944 0.8243296 0.7490349 
      100       101       104       105       106       110       114       115       122       126       128 
0.7865855 0.5127731 0.8757932 0.7775466 0.8147454 0.6849723 0.7102367 0.8830833 0.8184064 0.6678870 0.7367208 
      131       136       140       142       143       147       149       154       158       159       172 
0.7669207 0.6452953 0.2680037 0.7817289 0.4977444 0.8473658 0.8189059 0.7132797 0.7323481 0.7139369 0.8697600 
      174       175       183       190       196       199       201       213       217       219       224 
0.7407917 0.5986127 0.8354004 0.7985019 0.5281355 0.7322565 0.4029765 0.5868775 0.8947102 0.3768119 0.7743728 
      225       226       231       233       234       239       240       244       246       252       258 
```

```
loan$loan_status
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 [56] 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0
[111] 1 1 1 0 1 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
[166] 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1
[221] 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1
[276] 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1
```

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

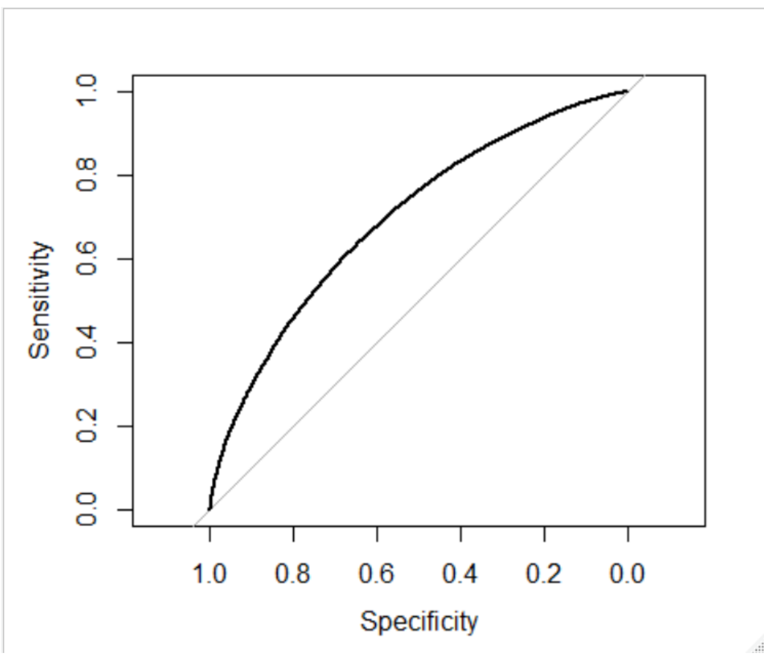
```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1
 [56] 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
[111] 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
[166] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[221] 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
[276] 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
install.packages("e1071")
library(e1071)
confusionMatrix(pdataF,as.factor(loan$loan_status))
```

```
> confusionMatrix(pdataF,as.factor(loan$loan_status))
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  3892  3230
         1 27323 96273

               Accuracy : 0.7663
                 95% CI : (0.764, 0.7686)
    No Information Rate : 0.7612
    P-Value [Acc > NIR] : 8.489e-06

                  Kappa : 0.1254

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.12468
            Specificity : 0.96754
         Pos Pred Value : 0.54648
         Neg Pred Value : 0.77893
             Prevalence : 0.23880
         Detection Rate : 0.02977
   Detection Prevalence : 0.05448
      Balanced Accuracy : 0.54611

       'Positive' Class : 0
```

library(pROC)
roc(loan$loan_status,logistic$fitted.values,plot=TRUE)

```
> roc(loan$loan_status,logistic$fitted.values,plot=TRUE)
Setting levels: control = 0, case = 1
Setting direction: controls < cases

Call:
roc.default(response = loan$loan_status, predictor = logistic$fitted.values,     plo
t = TRUE)

Data: logistic$fitted.values in 31215 controls (loan$loan_status 0) < 99503 cases (l
oan$loan_status 1).
Area under the curve: 0.696
>
```

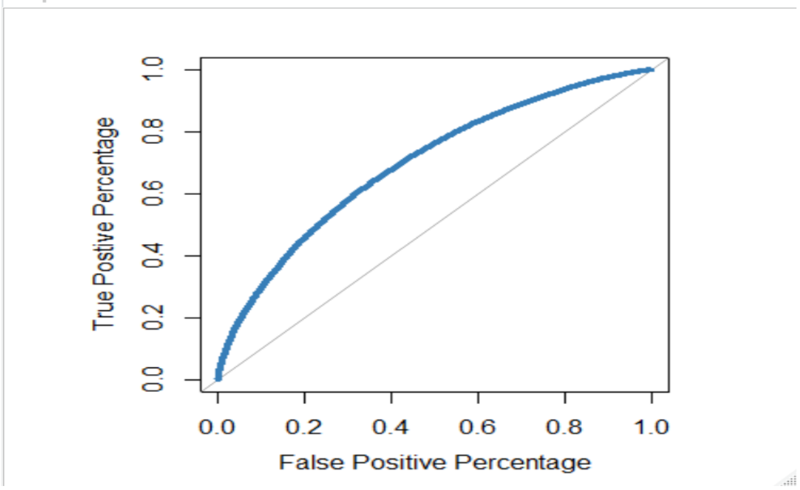


par(pty = "s")

NOTE: By default, roc() uses specificity on the x-axis and the values range
from 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.
#roc(loan$grade, glm.fit$fitted.values, plot=TRUE, legacy.axes=TRUE)

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

```
> roc(loan$loan_status,logistic$fitted.values,plot=TRUE, legacy.axes=TRUE, xlab="Fal
se Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4)
Setting levels: control = 0, case = 1
Setting direction: controls < cases

Call:
roc.default(response = loan$loan_status, predictor = logistic$fitted.values,     plo
t = TRUE, legacy.axes = TRUE, xlab = "False Positive Percentage",     ylab = "True P
ostive Percentage", col = "#377eb8", lwd = 4)

Data: logistic$fitted.values in 31215 controls (loan$loan_status 0) < 99503 cases (l
oan$loan_status 1).
Area under the curve: 0.696
>
```



#roc(loan$loan_status,logistic$fitted.values,plot=TRUE, legacy.axes=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4)
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(loan$loan_status, logistic$fitted.values, legacy.axes=TRUE)
str(roc.info)

```
List of 15
 $ percent          : logi FALSE
 $ sensitivities    : num [1:130716] 1 1 1 1 1 ...
 $ specificities    : num [1:130716] 0.00 0.00 3.20e-05 6.41e-05 9.61e-05 ...
 $ thresholds       : num [1:130716] -Inf 0.167 0.182 0.187 0.187 ...
 $ direction        : chr "<"
 $ cases            : Named num [1:99503] 0.904 0.848 0.835 0.702 0.95 ...
  ..- attr(*, "names")= chr [1:99503] "1" "2" "3" "4" ...
 $ controls         : Named num [1:31215] 0.945 0.876 0.609 0.878 0.895 ...
  ..- attr(*, "names")= chr [1:31215] "27" "135" "329" "361" ...
 $ fun.sesp         :function (thresholds, controls, cases, direction)
 $ auc              : 'auc' num 0.696
  ..- attr(*, "partial.auc")= logi FALSE
  ..- attr(*, "percent")= logi FALSE
  ..- attr(*, "roc")=List of 15
  .. ..$ percent          : logi FALSE
  .. ..$ sensitivities    : num [1:130716] 1 1 1 1 1 ...
  .. ..$ specificities    : num [1:130716] 0.00 0.00 3.20e-05 6.41e-05 9.61e-05 ...
  .. ..$ thresholds       : num [1:130716] -Inf 0.167 0.182 0.187 0.187 ...
  .. ..$ direction        : chr "<"
  .. ..$ cases            : Named num [1:99503] 0.904 0.848 0.835 0.702 0.95 ...
  .. .. ..- attr(*, "names")= chr [1:99503] "1" "2" "3" "4" ...
  .. ..$ controls         : Named num [1:31215] 0.945 0.876 0.609 0.878 0.895 ...
  .. .. ..- attr(*, "names")= chr [1:31215] "27" "135" "329" "361" ...
  .. ..$ fun.sesp         :function (thresholds, controls, cases, direction)
  .. ..$ auc              : 'auc' num 0.696
  .. .. ..- attr(*, "partial.auc")= logi FALSE
  .. .. ..- attr(*, "percent")= logi FALSE
  .. .. ..- attr(*, "roc")=List of 8
```

```
roc.df <- data.frame(tpp=roc.info$sensitivities*100, ## tpp = true positive percentage
                     fpp=(1 - roc.info$specificities)*100, ## fpp = false positive precentage
                     thresholds=roc.info$thresholds)
roc.df
```

```
> roc.df
         tpp       fpp thresholds
1  100.00000 100.00000       -Inf
2   99.99900 100.00000  0.1673491
3   99.99900  99.99680  0.1817184
4   99.99900  99.99359  0.1865814
5   99.99900  99.99039  0.1872915
6   99.99900  99.98719  0.1908892
7   99.99900  99.98398  0.1948157
8   99.99900  99.98078  0.1957164
9   99.99900  99.97757  0.1962113
10  99.99799  99.97757  0.1976654
11  99.99699  99.97757  0.1991788
12  99.99699  99.97437  0.2002878
13  99.99699  99.97117  0.2012987
14  99.99699  99.96796  0.2023670
15  99.99699  99.96476  0.2031951
16  99.99699  99.96156  0.2035219
17  99.99699  99.95835  0.2038186
18  99.99598  99.95835  0.2039926
19  99.99498  99.95835  0.2041858
20  99.99498  99.95515  0.2043695
21  99.99498  99.95195  0.2044938
22  99.99498  99.94874  0.2050438
23  99.99397  99.94874  0.2056024
24  99.99397  99.94554  0.2057853
25  99.99397  99.94234  0.2060304
```

head(roc.df) ## head() will show us the values for the upper right-hand corner of the ROC graph, when the threshold is so low

```
        tpp       fpp thresholds
1 100.000 100.00000       -Inf
2  99.999 100.00000  0.1673491
3  99.999  99.99680  0.1817184
4  99.999  99.99359  0.1865814
5  99.999  99.99039  0.1872915
6  99.999  99.98719  0.1908892
```

(negative infinity) that every single sample is called "obese".
Thus TPP = 100% and FPP = 100%
tail(roc.df) ## tail() will show us the values for the lower left-hand corner

```
               tpp         fpp thresholds
130711 0.004019979 0.003203588  0.9999913
130712 0.004019979 0.000000000  0.9999956
130713 0.003014984 0.000000000  0.9999997
130714 0.002009990 0.000000000  0.9999998
130715 0.001004995 0.000000000  1.0000000
130716 0.000000000 0.000000000        Inf
```

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%
roc.df[roc.df$tpp > 60 & roc.df$tpp < 80,]

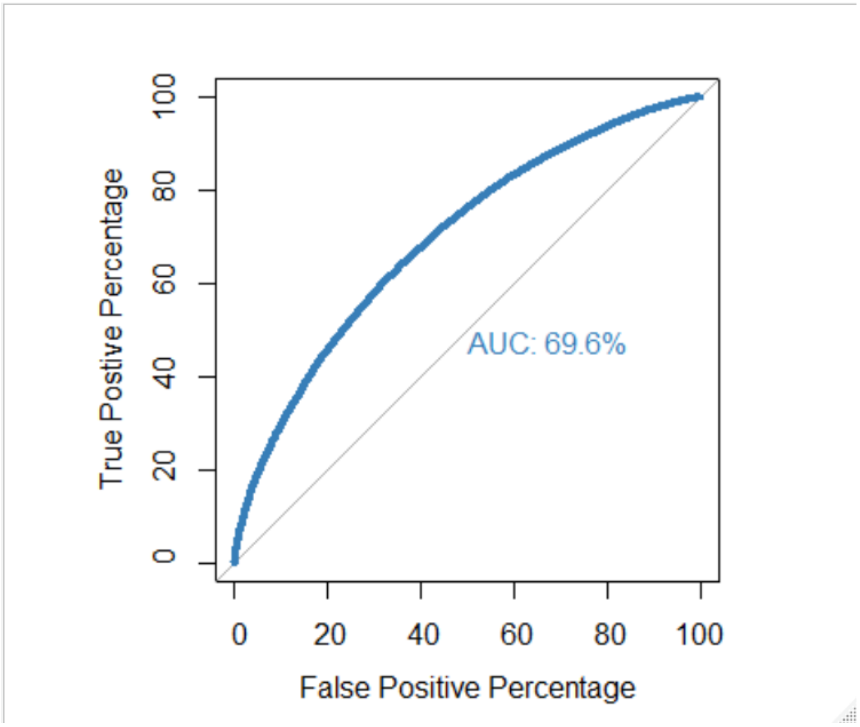
```
          tpp      fpp thresholds
33970 79.99960 54.93192  0.7072097
33971 79.99859 54.93192  0.7072121
33972 79.99859 54.92872  0.7072180
33973 79.99759 54.92872  0.7072251
33974 79.99759 54.92552  0.7072286
33975 79.99759 54.92231  0.7072312
33976 79.99658 54.92231  0.7072341
33977 79.99658 54.91911  0.7072376
33978 79.99558 54.91911  0.7072416
33979 79.99457 54.91911  0.7072427
33980 79.99357 54.91911  0.7072444
33981 79.99256 54.91911  0.7072543
33982 79.99156 54.91911  0.7072632
33983 79.99055 54.91911  0.7072660
33984 79.98955 54.91911  0.7072687
33985 79.98854 54.91911  0.7072709
33986 79.98754 54.91911  0.7072727
33987 79.98653 54.91911  0.7072754
33988 79.98553 54.91911  0.7072790
33989 79.98553 54.91591  0.7072830
33990 79.98452 54.91591  0.7072863
33991 79.98452 54.91270  0.7072953
33992 79.98352 54.91270  0.7073048
33993 79.98352 54.90950  0.7073089
33994 79.98352 54.90630  0.7073176
33995 79.98251 54.90630  0.7073242
33996 79.98151 54.90630  0.7073251
33997 79.98050 54.90630  0.7073255
33998 79.97950 54.90630  0.7073264
```

roc(loan$loan_status,logistic$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 = loan$loan_status, predictor = logistic$fitted.values,     per
cent = TRUE, plot = TRUE, legacy.axes = TRUE, xlab = "False Positive Percentage",
  ylab = "True Postive Percentage", col = "#377eb8", lwd = 4)

Data: logistic$fitted.values in 31215 controls (loan$loan_status 0) < 99503 cases (l
oan$loan_status 1).
Area under the curve: 69.6%
```

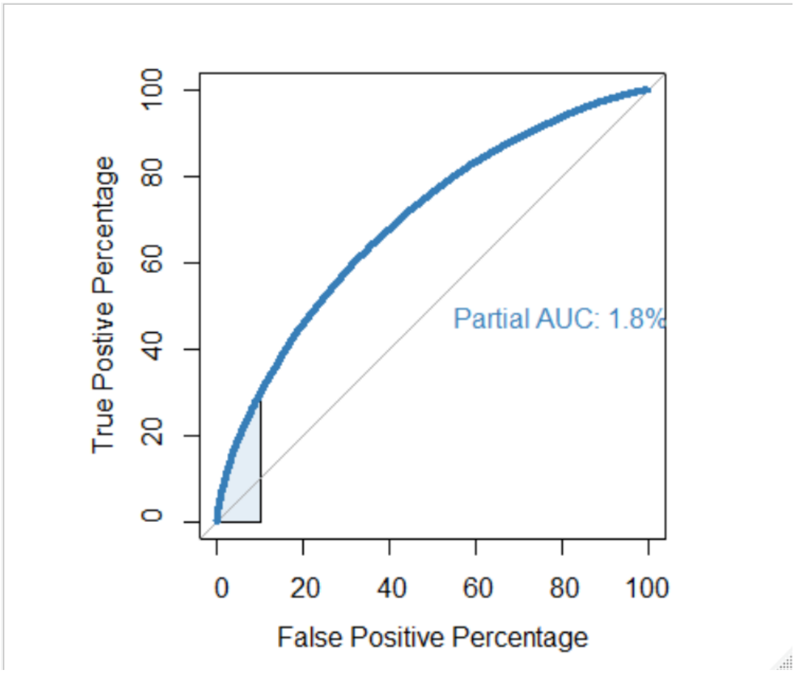
roc(loan$loan_status,logistic$fitted.values,plot=TRUE, legacy.axes=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4, percent=TRUE, print.auc=TRUE)

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

Call:
roc.default(response = loan$loan_status, predictor = logistic$fitted.values,     per
cent = TRUE, plot = TRUE, legacy.axes = TRUE, xlab = "False Positive Percentage",
  ylab = "True Postive Percentage", col = "#377eb8", lwd = 4,     print.auc = TRUE)

Data: logistic$fitted.values in 31215 controls (loan$loan_status 0) < 99503 cases (l
oan$loan_status 1).
Area under the curve: 69.6%
```



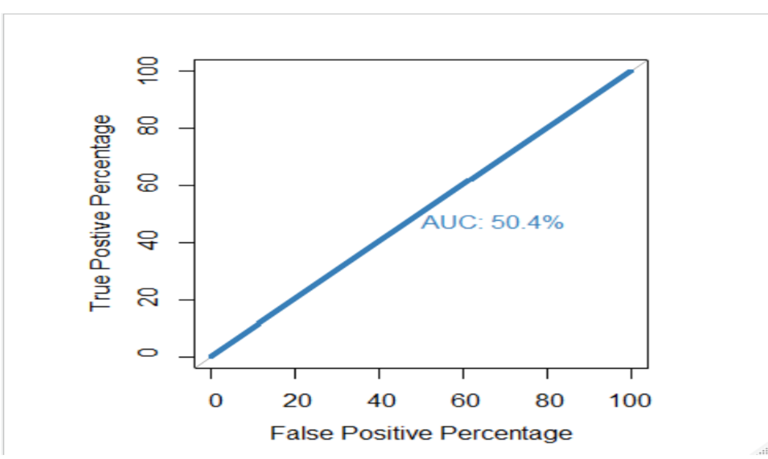
roc(loan$loan_status,logistic$fitted.values,plot=TRUE, legacy.axes=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4, percent=TRUE, print.auc=TRUE, partial.auc=c(100, 90), auc.polygon = TRUE, auc.polygon.col = "#377eb822", print.auc.x=45)

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

Call:
roc.default(response = loan$loan_status, predictor = logistic$fitted.values,     per
cent = TRUE, plot = TRUE, legacy.axes = TRUE, xlab = "False Positive Percentage",
  ylab = "True Postive Percentage", col = "#377eb8", lwd = 4,     print.auc = TRUE,
 partial.auc = c(100, 90), auc.polygon = TRUE,     auc.polygon.col = "#377eb822", pr
int.auc.x = 45)

Data: logistic$fitted.values in 31215 controls (loan$loan_status 0) < 99503 cases (l
oan$loan_status 1).
Partial area under the curve (specificity 100%-90%): 1.795%
```



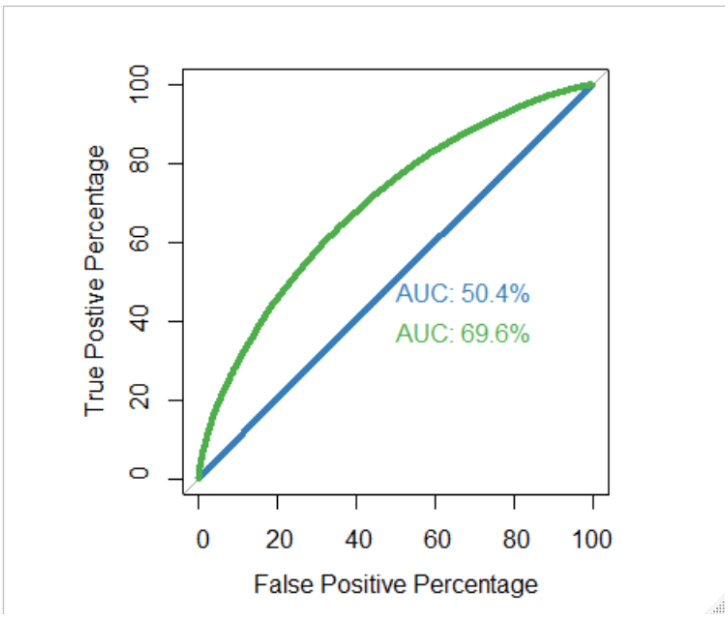
# Lets do two roc plots to understand which model is better
roc(loan$loan_status, logistic_simple$fitted.values, plot=TRUE, legacy.axes=TRUE, percent=TRUE, xlab="False Positive Percentage", ylab="True Postive Percentage", col="#377eb8", lwd=4, print.auc=TRUE)

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

Call:
roc.default(response = loan$loan_status, predictor = logistic_simple$fitted.values,
    percent = TRUE, plot = TRUE, legacy.axes = TRUE, xlab = "False Positive Percenta
ge",     ylab = "True Postive Percentage", col = "#377eb8", lwd = 4,     print.auc =
 TRUE)

Data: logistic_simple$fitted.values in 31215 controls (loan$loan_status 0) < 99503 c
ases (loan$loan_status 1).
Area under the curve: 50.39%
There were 27 warnings (use warnings() to see them)
```

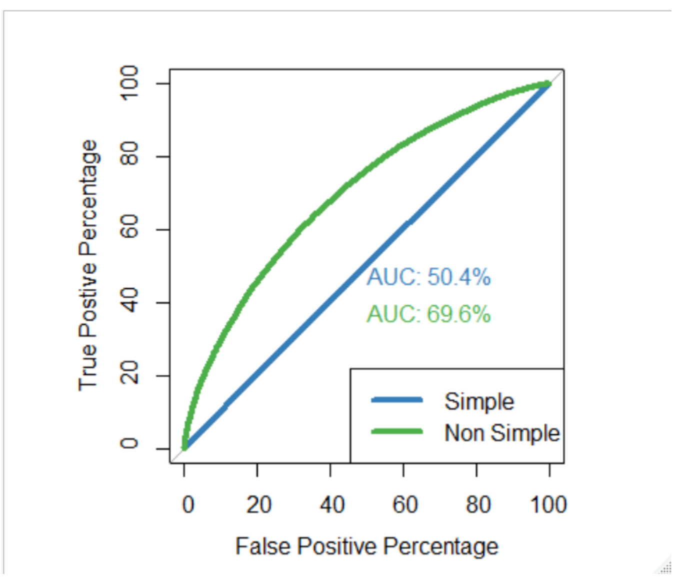


# Lets add the other graph
plot.roc(loan$loan_status, logistic$fitted.values, percent=TRUE, col="#4daf4a", lwd=4, print.auc=TRUE, add=TRUE, print.auc.y=40)

```
d=4, print.auc=TRUE, add=TRUE, print.a
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



#Conclusion- SO, according to the confusion matrix, the accuracy of logistic regression
 # is 76.63% i.e., the model classifies values correctly 76 times out of 100.