R Notebook

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2

## -- Attaching packages ------------------------------------------------------------------------------------

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.0     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.3

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.3

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

```r
library(mice)
```

```
## Warning: package 'mice' was built under R version 3.4.2

## Loading required package: lattice

##
## Attaching package: 'mice'

## The following object is masked from 'package:tidyr':
##
##     complete
```

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

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

```r
library(caTools)
```

Predicting loan repayment

In the lending industry, investors provide loans to borrowers in exchange for the promise of repayment with interest. If the borrower repays the loan, then the lender profits from the interest. However, if the borrower is unable to repay the loan, then the lender loses money. Therefore, lenders face the problem of predicting the risk of a borrower being unable to repay a loan.

To address this problem, we will use publicly available data from LendingClub.com, a website that connects borrowers and investors over the Internet. This dataset represents 9,578 3-year loans that were funded through the LendingClub.com platform between May 2007 and February 2010. The binary dependent variable not.fully.paid indicates that the loan was not paid back in full (the borrower either defaulted or the loan was "charged off," meaning the borrower was deemed unlikely to ever pay it back).

To predict this dependent variable, we will use the following independent variables available to the investor when deciding whether to fund a loan:

- **credit.policy**: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.
- **purpose**: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").
- **int.rate**: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.
- **installment**: The monthly installments ($) owed by the borrower if the loan is funded.
- **log.annual.inc**: The natural log of the self-reported annual income of the borrower.
- **dti**: The debt-to-income ratio of the borrower (amount of debt divided by annual income).
- **fico**: The FICO credit score of the borrower.
- **days.with.cr.line**: The number of days the borrower has had a credit line.
- **revol.bal**: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).
- **revol.util**: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).
- **inq.last.6mths**: The borrower's number of inquiries by creditors in the last 6 months.
- **delinq.2yrs**: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
- **pub.rec**: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

1.1) Preparing the Dataset

What proportion of the loans in the dataset were not paid in full?

```r
loans = read.csv('loans.csv')

str(loans)
```

```
## 'data.frame':    9578 obs. of  14 variables:
##  $ credit.policy    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ purpose          : Factor w/ 7 levels "all_other","credit_card",..: 3 2 3 3 2 2 3 1 5 3 ...
##  $ int.rate         : num  0.119 0.107 0.136 0.101 0.143 ...
##  $ installment      : num  829 228 367 162 103 ...
##  $ log.annual.inc   : num  11.4 11.1 10.4 11.4 11.3 ...
##  $ dti              : num  19.5 14.3 11.6 8.1 15 ...
##  $ fico             : int  737 707 682 712 667 727 667 722 682 707 ...
##  $ days.with.cr.line: num  5640 2760 4710 2700 4066 ...
##  $ revol.bal        : int  28854 33623 3511 33667 4740 50807 3839 24220 69909 5630 ...
##  $ revol.util       : num  52.1 76.7 25.6 73.2 39.5 51 76.8 68.6 51.1 23 ...
##  $ inq.last.6mths   : int  0 0 1 1 0 0 0 0 1 1 ...
##  $ delinq.2yrs      : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ pub.rec          : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ not.fully.paid   : int  0 0 0 0 0 0 1 1 0 0 ...
```

```r
summary(loans)
```

```
##  credit.policy                 purpose        int.rate
##  Min.   :0.000   all_other         :2331   Min.   :0.0600
##  1st Qu.:1.000   credit_card       :1262   1st Qu.:0.1039
##  Median :1.000   debt_consolidation:3957   Median :0.1221
##  Mean   :0.805   educational       : 343   Mean   :0.1226
##  3rd Qu.:1.000   home_improvement  : 629   3rd Qu.:0.1407
##  Max.   :1.000   major_purchase    : 437   Max.   :0.2164
##                  small_business    : 619
##   installment     log.annual.inc       dti              fico
##  Min.   : 15.67   Min.   : 7.548   Min.   : 0.000   Min.   :612.0
##  1st Qu.:163.77   1st Qu.:10.558   1st Qu.: 7.213   1st Qu.:682.0
##  Median :268.95   Median :10.928   Median :12.665   Median :707.0
##  Mean   :319.09   Mean   :10.932   Mean   :12.607   Mean   :710.8
##  3rd Qu.:432.76   3rd Qu.:11.290   3rd Qu.:17.950   3rd Qu.:737.0
##  Max.   :940.14   Max.   :14.528   Max.   :29.960   Max.   :827.0
##                   NA's   :4
##  days.with.cr.line   revol.bal         revol.util     inq.last.6mths
##  Min.   :  179     Min.   :      0   Min.   :  0.00   Min.   : 0.000
##  1st Qu.: 2820     1st Qu.:   3187   1st Qu.: 22.70   1st Qu.: 0.000
##  Median : 4140     Median :   8596   Median : 46.40   Median : 1.000
##  Mean   : 4562     Mean   :  16914   Mean   : 46.87   Mean   : 1.572
##  3rd Qu.: 5730     3rd Qu.:  18250   3rd Qu.: 71.00   3rd Qu.: 2.000
##  Max.   :17640     Max.   :1207359   Max.   :119.00   Max.   :33.000
##  NA's   :29                          NA's   :62       NA's   :29
##   delinq.2yrs         pub.rec       not.fully.paid
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 0.0000   Median :0.0000   Median :0.0000
##  Mean   : 0.1638   Mean   :0.0621   Mean   :0.1601
##  3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :13.0000   Max.   :5.0000   Max.   :1.0000
##  NA's   :29        NA's   :29
```

```r
table(loans$not.fully.paid) #1: not pay in full
```

```
##
##    0    1
## 8045 1533
```

```r
prop = 1533 / (1533 + 8045)

prop
```

```
## [1] 0.1600543
```

Explanation

From table(loans$not.fully.paid), we see that 1533 loans were not paid, and 8045 were fully paid. Therefore, the proportion of loans not paid is 1533/(1533+8045)=0.1601.

1.2) Preparing the Dataset

Which of the following variables has at least one missing observation?

```r
sum(is.na(loans))
```

```
## [1] 182
```

```r
summary(loans)
```

```
##  credit.policy                 purpose        int.rate     
##  Min.   :0.000   all_other         :2331   Min.   :0.0600  
##  1st Qu.:1.000   credit_card       :1262   1st Qu.:0.1039  
##  Median :1.000   debt_consolidation:3957   Median :0.1221  
##  Mean   :0.805   educational       : 343   Mean   :0.1226  
##  3rd Qu.:1.000   home_improvement  : 629   3rd Qu.:0.1407  
##  Max.   :1.000   major_purchase    : 437   Max.   :0.2164  
##                  small_business    : 619                   
##   installment     log.annual.inc        dti              fico      
##  Min.   : 15.67   Min.   : 7.548   Min.   : 0.000   Min.   :612.0  
##  1st Qu.:163.77   1st Qu.:10.558   1st Qu.: 7.213   1st Qu.:682.0  
##  Median :268.95   Median :10.928   Median :12.665   Median :707.0  
##  Mean   :319.09   Mean   :10.932   Mean   :12.607   Mean   :710.8  
##  3rd Qu.:432.76   3rd Qu.:11.290   3rd Qu.:17.950   3rd Qu.:737.0  
##  Max.   :940.14   Max.   :14.528   Max.   :29.960   Max.   :827.0  
##                   NA's   :4                                        
##  days.with.cr.line   revol.bal         revol.util     inq.last.6mths  
##  Min.   :  179     Min.   :      0   Min.   :  0.00   Min.   : 0.000  
##  1st Qu.: 2820     1st Qu.:   3187   1st Qu.: 22.70   1st Qu.: 0.000  
##  Median : 4140     Median :   8596   Median : 46.40   Median : 1.000  
##  Mean   : 4562     Mean   :  16914   Mean   : 46.87   Mean   : 1.572  
##  3rd Qu.: 5730     3rd Qu.:  18250   3rd Qu.: 71.00   3rd Qu.: 2.000  
##  Max.   :17640     Max.   :1207359   Max.   :119.00   Max.   :33.000  
##  NA's   :29                          NA's   :62       NA's   :29      
##   delinq.2yrs         pub.rec       not.fully.paid  
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median : 0.0000   Median :0.0000   Median :0.0000  
##  Mean   : 0.1638   Mean   :0.0621   Mean   :0.1601  
##  3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000  
##  Max.   :13.0000   Max.   :5.0000   Max.   :1.0000  
##  NA's   :29        NA's   :29                       
```

Explanation From summary(loans), we can read that log.annual.inc, days.with.cr.line, revol.util, inq.last.6mths, delinq.2yrs and pub.rec are missing values.

1.3) Preparing the Dataset

Which of the following is the best reason to fill in the missing values for these variables instead of removing observations with missing data? (Hint: you can use the subset() function to build a data frame with the observations missing at least one value. To test if a variable, for example pub.rec, is missing a value, use is.na(pub.rec).)

```r
test1 = subset(loans, is.na(pub.rec) | is.na(days.with.cr.line) | is.na(revol.util) |
                 is.na(inq.last.6mths) | is.na(delinq.2yrs) | is.na(log.annual.inc))

head(test1)
```

```
##     credit.policy            purpose int.rate installment log.annual.inc
## 782             1          all_other   0.1134       98.70      10.530495
```

```
## 804               1        educational   0.1103       52.41      10.532096
## 840               1 debt_consolidation   0.1134      263.20      10.714418
## 858               1 debt_consolidation   0.1229       23.35       9.852194
## 1214              1     major_purchase   0.1064      182.39      11.264464
## 1281              1        credit_card   0.1633      264.91      10.819778
##        dti fico days.with.cr.line revol.bal revol.util inq.last.6mths
## 782   7.72  677          1680.000         0         NA              1
## 804  15.84  682          1829.958         0         NA              0
## 840   8.75  682          2490.000         0         NA              1
## 858  12.38  662          1199.958         0         NA              1
## 1214  4.26  697          4140.958         0         NA              0
## 1281 10.80  667          5249.958         0         NA              0
##      delinq.2yrs pub.rec not.fully.paid
## 782            0       0              1
## 804            0       0              0
## 840            1       0              1
## 858            0       0              0
## 1214           0       1              0
## 1281           0       0              1
```

```r
table(test1$not.fully.paid)
```

```
##
##  0  1
## 50 12
```

```r
12/62
```

```
## [1] 0.1935484
```

Ans: We want to be able to predict risk for all borrowers, instead of just the ones with all data reported.

Explanation Answering this question requires analyzing the loans with missing data. We can build a data frame limited to observations with some missing data with the following command:

missing = subset(loans, is.na(log.annual.inc) | is.na(days.with.cr.line) | is.na(revol.util) | is.na(inq.last.6mths) | is.na(delinq.2yrs) | is.na(pub.rec))

From nrow(missing), we see that only 62 of 9578 loans have missing data; removing this small number of observations would not lead to overfitting. From table(missing$not.fully.paid), we see that 12 of 62 loans with missing data were not fully paid, or 19.35%. This rate is similar to the 16.01% across all loans, so the form of biasing described is not an issue. However, to predict risk for loans with missing data we need to fill in the missing values instead of removing the observations.

1.4) Preparing the Dataset

For the rest of this problem, we'll be using a revised version of the dataset that has the missing values filled in with multiple imputation (which was discussed in the Recitation of this Unit). To ensure everybody has the same data frame going forward, you can either run the commands below in your R console (if you haven't already, run the command install.packages("mice") first), or you can download and load into R the dataset we created after running the imputation: loans_imputed.csv.

Note that to do this imputation, we set vars.for.imputation to all variables in the data frame except for not.fully.paid, to impute the values using all of the other independent variables.

```r
set.seed(144)
```

```r
vars_for_imputation = setdiff(names(loans), "not.fully.paid")

imputed = complete(mice(loans[vars_for_imputation]))
```

```
##
##  iter imp variable
##   1   1  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   1   2  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   1   3  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   1   4  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   1   5  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   2   1  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   2   2  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   2   3  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   2   4  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   2   5  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   3   1  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   3   2  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   3   3  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   3   4  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   3   5  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   4   1  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   4   2  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   4   3  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   4   4  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   4   5  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   5   1  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   5   2  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   5   3  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   5   4  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
##   5   5  log.annual.inc  days.with.cr.line  revol.util  inq.last.6mths  delinq.2yrs  pub.rec
```

```r
loans[vars_for_imputation] = imputed

sum(is.na(loans))
```

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

Ans: We predicted missing variable values using the available independent variables for each observation.

Explanation

Imputation predicts missing variable values for a given observation using the variable values that are reported. We called the imputation on a data frame with the dependent variable not.fully.paid removed, so we predicted the missing values using only other independent variables.

2.1) Prediction Models

Now that we have prepared the dataset, we need to split it into a training and testing set. To ensure everybody obtains the same split, set the random seed to 144 (even though you already did so earlier in the problem) and use the sample.split function to select the 70% of observations for the training set (the dependent variable for sample.split is not.fully.paid). Name the data frames train and test.

Now, use logistic regression trained on the training set to predict the dependent variable not.fully.paid using all the independent variables.

Which independent variables are significant in our model? (Significant variables have at least one star, or a Pr(>|z|) value less than 0.05.) Select all that apply.

```r
set.seed(144)

split = sample.split(loans$not.fully.paid, SplitRatio = 0.7)

train = subset(loans, split == TRUE)

test = subset(loans, split == FALSE)

nrow(train)
```

```
## [1] 6705
```

```r
nrow(test)
```

```
## [1] 2873
```

```r
mod_1 = glm(not.fully.paid ~ ., data = train, family = 'binomial')

summary(mod_1)
```

```
##
## Call:
## glm(formula = not.fully.paid ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2097  -0.6214  -0.4950  -0.3601   2.6414
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        9.260e+00  1.554e+00   5.958 2.55e-09 ***
## credit.policy     -3.327e-01  1.011e-01  -3.292 0.000995 ***
## purpose2          -6.100e-01  1.344e-01  -4.538 5.67e-06 ***
## purpose3          -3.181e-01  9.185e-02  -3.463 0.000534 ***
## purpose4           1.386e-01  1.753e-01   0.791 0.429074
## purpose5           1.774e-01  1.479e-01   1.199 0.230496
## purpose6          -4.783e-01  2.009e-01  -2.381 0.017260 *
## purpose7           4.159e-01  1.419e-01   2.932 0.003373 **
## int.rate           6.202e-01  2.085e+00   0.297 0.766111
## installment        1.279e-03  2.093e-04   6.110 9.96e-10 ***
## log.annual.inc    -4.357e-01  7.151e-02  -6.093 1.11e-09 ***
## dti                4.733e-03  5.501e-03   0.861 0.389508
## fico              -9.406e-03  1.707e-03  -5.510 3.60e-08 ***
## days.with.cr.line  3.174e-06  1.587e-05   0.200 0.841463
## revol.bal          3.103e-06  1.169e-06   2.653 0.007966 **
## revol.util         1.796e-03  1.532e-03   1.172 0.241022
## inq.last.6mths     8.386e-02  1.577e-02   5.317 1.06e-07 ***
## delinq.2yrs       -7.794e-02  6.532e-02  -1.193 0.232814
## pub.rec            3.191e-01  1.134e-01   2.814 0.004899 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 5896.6  on 6704  degrees of freedom
## Residual deviance: 5485.7  on 6686  degrees of freedom
## AIC: 5523.7
##
## Number of Fisher Scoring iterations: 5
```

Explanation

The following steps are needed to split the data:

library(caTools)

set.seed(144)

spl = sample.split(loans$not.fully.paid, 0.7)

train = subset(loans, spl == TRUE)

test = subset(loans, spl == FALSE)

The model can be trained and summarized with the following commands:

mod = glm(not.fully.paid~., data=train, family="binomial")

summary(mod)

Variables that are significant have at least one star in the coefficients table of the summary output. Note that some have a positive coefficient (meaning that higher values of the variable lead to an increased risk of defaulting) and some have a negative coefficient (meaning that higher values of the variable lead to a decreased risk of defaulting).

2.2) Prediction Models

Consider two loan applications, which are identical other than the fact that the borrower in Application A has FICO credit score 700 while the borrower in Application B has FICO credit score 710.

Let Logit(A) be the log odds of loan A not being paid back in full, according to our logistic regression model, and define Logit(B) similarly for loan B. What is the value of Logit(A) - Logit(B)?

Answer:0.09317 Explanation Because Application A is identical to Application B other than having a FICO score 10 lower, its predicted log odds differ by -0.009317 * -10 = 0.09317 from the predicted log odds of Application B.

Now, let O(A) be the odds of loan A not being paid back in full, according to our logistic regression model, and define O(B) similarly for loan B. What is the value of O(A)/O(B)? (HINT: Use the mathematical rule that exp(A + B + C) = exp(A)*exp(B)*exp(C). Also, remember that exp() is the exponential function in R.)

Answer: 1.0976 Explanation Using the answer from the previous question, the predicted odds of loan A not being paid back in full are exp(0.09317) = 1.0976 times larger than the predicted odds for loan B. Intuitively, it makes sense that loan A should have higher odds of non-payment than loan B, since the borrower has a worse credit score.

2.3) Prediction Models

Predict the probability of the test set loans not being paid back in full (remember type="response" for the predict function). Store these predicted probabilities in a variable named predicted.risk and add it to your test set (we will use this variable in later parts of the problem). Compute the confusion matrix using a threshold of 0.5.

What is the accuracy of the logistic regression model? Input the accuracy as a number between 0 and 1.

```r
predict_test = predict(mod_1, type = 'response', newdata = test)
```

```
## Warning: contrasts dropped from factor purpose
```

```r
test$predicted_risk = predict_test

table(test$not.fully.paid, predict_test >  0.5)
```

```
##
##     FALSE TRUE
##   0  2400   13
##   1   457    3
```

```r
accuracy = (2400 + 3) / (2400 + 13 + 457 + 3)

accuracy
```

```
## [1] 0.8364079
```

What is the accuracy of the baseline model? Input the accuracy as a number between 0 and 1.

```r
accu_baseline = (2400 + 13) / (2400 + 13 + 457 + 3)

accu_baseline
```

```
## [1] 0.8398886
```

Explanation

The confusion matrix can be computed with the following commands:

test$predicted.risk = predict(mod, newdata=test, type="response")

table(test*not.fully.paid, test*predicted.risk $>$ 0.5)

2403 predictions are correct (accuracy 2403/2873=0.8364), while 2413 predictions would be correct in the baseline model of guessing every loan would be paid back in full (accuracy 2413/2873=0.8399).

2.4) Prediction Models

Use the ROCR package to compute the test set AUC.

```r
ROCR_pred_test = prediction(predict_test, test$not.fully.paid)

auc = as.numeric(performance(ROCR_pred_test, "auc")@y.values)

auc
```

```
## [1] 0.6721337
```

The model has poor accuracy at the threshold 0.5. But despite the poor accuracy, we will see later how an investor can still leverage this logistic regression model to make profitable investments.

Explanation

The test set AUC can be computed with the following commands:

library(ROCR)

pred = prediction(test*predicted.risk, test*not.fully.paid)

as.numeric(performance(pred, "auc")@y.values)

3.1) A "Smart Baseline"

In the previous problem, we built a logistic regression model that has an AUC significantly higher than the AUC of 0.5 that would be obtained by randomly ordering observations.

However, LendingClub.com assigns the interest rate to a loan based on their estimate of that loan's risk. This variable, int.rate, is an independent variable in our dataset. In this part, we will investigate using the loan's interest rate as a "smart baseline" to order the loans according to risk.

Using the training set, build a bivariate logistic regression model (aka a logistic regression model with a single independent variable) that predicts the dependent variable not.fully.paid using only the variable int.rate.

The variable int.rate is highly significant in the bivariate model, but it is not significant at the 0.05 level in the model trained with all the independent variables. What is the most likely explanation for this difference?

```r
mod_bivariate = glm(not.fully.paid ~ int.rate, data = train, family = 'binomial')

summary(mod_bivariate)
```

```
##
## Call:
## glm(formula = not.fully.paid ~ int.rate, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0547  -0.6271  -0.5442  -0.4361   2.2914
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.6726     0.1688  -21.76   <2e-16 ***
## int.rate     15.9214     1.2702   12.54   <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: 5896.6  on 6704  degrees of freedom
## Residual deviance: 5734.8  on 6703  degrees of freedom
## AIC: 5738.8
##
## Number of Fisher Scoring iterations: 4
```

Ans: int.rate is correlated with other risk-related variables, and therefore does not incrementally improve the model when those other variables are included.

Explanation

To train the bivariate model, run the following command:

bivariate = glm(not.fully.paid~int.rate, data=train, family="binomial")

summary(bivariate)

Decreased significance between a bivariate and multivariate model is typically due to correlation. From cor(train$int.rate, train$fico), we can see that the interest rate is moderately well correlated with a borrower's credit score.

Training/testing set split rarely has a large effect on the significance of variables (this can be verified in this case by trying out a few other training/testing splits), and the models were trained on the same observations.

3.2) A "Smart Baseline"

Make test set predictions for the bivariate model. What is the highest predicted probability of a loan not being paid in full on the testing set?

```
predict_test_bivariate = predict(mod_bivariate, type = 'response',newdata = test)

head(sort(abs(predict_test_bivariate), decreasing = TRUE))
```

```
##      5869      7314      9477      6433      9249      9338
## 0.4266240 0.4145967 0.4145967 0.3999925 0.3999925 0.3999925
```

```
summary(predict_test_bivariate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06196 0.11549 0.15077 0.15963 0.18928 0.42662
```

With a logistic regression cutoff of 0.5, how many loans would be predicted as not being paid in full on the testing set?

```
table(test$not.fully.paid, predict_test_bivariate > 0.5)
```

```
##
##     FALSE
##   0  2413
##   1   460
```

Explanation

Make and summarize the test set predictions with the following code:

pred.bivariate = predict(bivariate, newdata=test, type="response")

summary(pred.bivariate)

According to the summary function, the maximum predicted probability of the loan not being paid back is 0.4266, which means no loans would be flagged at a logistic regression cutoff of 0.5.

3.3) A "Smart Baseline"

What is the test set AUC of the bivariate model?

```
ROCR_pred_test_bivariate = prediction(predict_test_bivariate, test$not.fully.paid)

auc = as.numeric(performance(ROCR_pred_test_bivariate, "auc")@y.values)

auc
```

```
## [1] 0.6239081
```

Explanation

The AUC can be computed with:

prediction.bivariate = prediction(pred.bivariate, test$not.fully.paid)

as.numeric(performance(prediction.bivariate, "auc")@y.values)

4.1) Computing the Profitablity of an Investment

While thus far we have predicted if a loan will be paid back or not, an investor needs to identify loans that are expected to be profitable. If the loan is paid back in full, then the investor makes interest on the loan.

However, if the loan is not paid back, the investor loses the money invested. Therefore, the investor should seek loans that best balance this risk and reward.

To compute interest revenue, consider a $c investment in a loan that has an annual interest rate r over a period of t years. Using continuous compounding of interest, this investment pays back c times exp(rt) dollars by the end of the t years, where exp(rt) is e raised to the r*t power.

How much does a 10 dollar investment with an annual interest rate of 6% pay back after 3 years, using continuous compounding of interest? Hint: remember to convert the percentage to a proportion before doing the math. Enter the number of dollars, without the $ sign.

Ans: 11.97 Explanation

In this problem, we have c=10, r=0.06, and t=3. Using the formula above, the final value is $10 exp(0.06 3) =$ 11.97.

4.2) Computing the Profitability of an Investment

While the investment has value c * exp(rt) dollars after collecting interest, the investor had to pay $c for the investment. What is the profit to the investor if the investment is paid back in full?

Explanation

A person's profit is what they get minus what they paid for it. In this case, the investor gets c * exp(rt) but paid c, yielding a profit of c * exp(rt) - c.

4.3) Computing the Profitability of an Investment

Now, consider the case where the investor made a $c investment, but it was not paid back in full. Assume, conservatively, that no money was received from the borrower (often a lender will receive some but not all of the value of the loan, making this a pessimistic assumption of how much is received). What is the profit to the investor in this scenario?

Explanation

A person's profit is what they get minus what they paid for it. In this case, the investor gets no money but paid c dollars, yielding a profit of -c dollars.

5.1) A Simple Investment Strategy

In the previous subproblem, we concluded that an investor who invested c dollars in a loan with interest rate r for t years makes c * (exp(rt) - 1) dollars of profit if the loan is paid back in full and -c dollars of profit if the loan is not paid back in full (pessimistically).

In order to evaluate the quality of an investment strategy, we need to compute this profit for each loan in the test set. For this variable, we will assume a $1 investment (aka c=1). To create the variable, we first assign to the profit for a fully paid loan, exp(rt)-1, to every observation, and we then replace this value with -1 in the cases where the loan was not paid in full. All the loans in our dataset are 3-year loans, meaning t=3 in our calculations. Enter the following commands in your R console to create this new variable:

```r
test$profit = exp(test$int.rate*3) - 1

test$profit[test$not.fully.paid == 1] = -1
```

What is the maximum profit of a 10-dollar investment in any loan in the testing set (do not include the $ sign in your answer)?

```r
summary(test$profit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.0000  0.2858  0.4111  0.2094  0.4980  0.8895
```

Explanation

From summary(test$profit), we see the maximum profit for a $1 investment in any loan is $0.8895. Therefore, the maximum profit of a $10 investment is 10 times as large, or $8.895.

6.1) An Investment Strategy Based on Risk

A simple investment strategy of equally investing in all the loans would yield profit 20.94 dollars for a $100 investment. But this simple investment strategy does not leverage the prediction model we built earlier in this problem. As stated earlier, investors seek loans that balance reward with risk, in that they simultaneously have high interest rates and a low risk of not being paid back.

To meet this objective, we will analyze an investment strategy in which the investor only purchases loans with a high interest rate (a rate of at least 15%), but amongst these loans selects the ones with the lowest predicted risk of not being paid back in full. We will model an investor who invests $1 in each of the most promising 100 loans.

First, use the subset() function to build a data frame called highInterest consisting of the test set loans with an interest rate of at least 15%.

```r
high_interest = subset(test, int.rate >= 0.15)

summary(high_interest)
```

```
##   credit.policy                   purpose       int.rate
##   Min.   :0.0000   all_other         : 80   Min.   :0.1501
##   1st Qu.:0.0000   credit_card       : 41   1st Qu.:0.1545
##   Median :1.0000   debt_consolidation:203   Median :0.1607
##   Mean   :0.6064   educational       : 15   Mean   :0.1651
##   3rd Qu.:1.0000   home_improvement  : 19   3rd Qu.:0.1719
##   Max.   :1.0000   major_purchase    : 12   Max.   :0.2121
##                    small_business    : 67
##    installment      log.annual.inc        dti             fico
##   Min.   : 21.59   Min.   : 8.476   Min.   : 0.00   Min.   :612.0
##   1st Qu.:187.14   1st Qu.:10.636   1st Qu.: 8.34   1st Qu.:662.0
##   Median :350.02   Median :11.002   Median :14.93   Median :672.0
##   Mean   :407.87   Mean   :11.014   Mean   :14.53   Mean   :676.9
##   3rd Qu.:565.02   3rd Qu.:11.408   3rd Qu.:20.56   3rd Qu.:687.0
##   Max.   :926.83   Max.   :13.305   Max.   :29.96   Max.   :802.0
##
##   days.with.cr.line   revol.bal        revol.util     inq.last.6mths
##   Min.   :  419     Min.   :     0   Min.   :  0.00   Min.   : 0.000
##   1st Qu.: 2340     1st Qu.:  4490   1st Qu.: 43.90   1st Qu.: 0.000
##   Median : 3900     Median : 10971   Median : 69.10   Median : 1.000
##   Mean   : 4117     Mean   : 19717   Mean   : 63.82   Mean   : 2.176
##   3rd Qu.: 5403     3rd Qu.: 22917   3rd Qu.: 88.00   3rd Qu.: 3.000
##   Max.   :14580     Max.   :226936   Max.   :103.10   Max.   :10.000
##
##    delinq.2yrs        pub.rec         not.fully.paid   predicted_risk
##   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.06876
```

```
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.18094
##  Median :0.0000   Median :0.00000   Median :0.0000   Median :0.22630
##  Mean   :0.2334   Mean   :0.09153   Mean   :0.2517   Mean   :0.24664
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.29670
##  Max.   :4.0000   Max.   :3.00000   Max.   :1.0000   Max.   :0.58114
##
##      profit
##  Min.   :-1.0000
##  1st Qu.:-1.0000
##  Median : 0.5992
##  Mean   : 0.2251
##  3rd Qu.: 0.6380
##  Max.   : 0.8895
##
```

What is the average profit of a 1-dollar investment in one of these high-interest loans (do not include the $ sign in your answer)?

```
high_interest$profit = exp(high_interest$int.rate*3) - 1

high_interest$profit[high_interest$not.fully.paid == 1] = -1

mean(high_interest$profit)
```

```
## [1] 0.2251015
```

What proportion of the high-interest loans were not paid back in full?

```
table(high_interest$not.fully.paid)
```

```
##
##   0   1
## 327 110
```

```
110 / (110 + 327)
```

```
## [1] 0.2517162
```

```
predict_high_interest = predict(mod_1, type = "response", newdata = high_interest)
```

```
## Warning: contrasts dropped from factor purpose
```

```
table(high_interest$not.fully.paid, predict_high_interest > 0.5)
```

```
##
##     FALSE TRUE
##   0   322    5
##   1   109    1
```

Explanation

The following two commands build the data frame highInterest and summarize the profit variable.

highInterest = subset(test, int.rate >= 0.15)

summary(highInterest$profit)

We read that the mean profit is $0.2251.

To obtain the breakdown of whether the loans were paid back in full, we can use

table(highInterest$not.fully.paid)

110 of the 437 loans were not paid back in full, for a proportion of 0.2517.

6.2) An Investment Strategy Based on Risk

Next, we will determine the 100th smallest predicted probability of not paying in full by sorting the predicted risks in increasing order and selecting the 100th element of this sorted list. Find the highest predicted risk that we will include by typing the following command into your R console:

```
cutoff = sort(high_interest$predicted_risk, decreasing = FALSE)[100]

cutoff
```

```
## [1] 0.1769593
```

Use the subset() function to build a data frame called selectedLoans consisting of the high-interest loans with predicted risk not exceeding the cutoff we just computed. Check to make sure you have selected 100 loans for investment.

```
selected_loans = subset(high_interest, predicted_risk <= cutoff)

str(selected_loans)
```

```
## 'data.frame':    100 obs. of  16 variables:
##  $ credit.policy    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ purpose          : Factor w/ 7 levels "all_other","credit_card",..: 7 2 3 1 3 1 5 2 3 2 ...
##   ..- attr(*, "contrasts")= num [1:7, 1:6] 0 1 0 0 0 0 0 0 0 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr  "all_other" "credit_card" "debt_consolidation" "educational" ...
##   .. .. ..$ : chr  "2" "3" "4" "5" ...
##  $ int.rate         : num  0.15 0.153 0.158 0.159 0.156 ...
##  $ installment      : num  225 444 420 246 245 ...
##  $ log.annual.inc   : num  12.3 11 11.5 11.5 10.8 ...
##  $ dti              : num  6.45 19.52 18.55 24.19 2.72 ...
##  $ fico             : int  677 667 667 667 672 687 702 667 672 662 ...
##  $ days.with.cr.line: num  6240 2701 4560 5376 3010 ...
##  $ revol.bal        : int  56411 33074 34841 590 3273 0 4980 15977 16473 22783 ...
##  $ revol.util       : num  75.3 68.8 89.6 84.3 69.6 4.5 55.3 83.6 94.1 93.7 ...
##  $ inq.last.6mths   : int  0 2 0 0 1 1 1 0 2 3 ...
##  $ delinq.2yrs      : int  0 0 0 0 0 0 0 0 2 1 ...
##  $ pub.rec          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ not.fully.paid   : int  1 0 0 0 1 0 0 0 0 0 ...
##  $ predicted_risk   : num  0.164 0.169 0.158 0.162 0.147 ...
##  $ profit           : num  -1 0.584 0.604 0.61 -1 ...
```

What is the profit of the investor, who invested 1 dollar in each of these 100 loans (do not include the $ sign in your answer)?

```
selected_loans$profit = exp(selected_loans$int.rate*3) - 1

selected_loans$profit[selected_loans$not.fully.paid == 1] = -1

sum(selected_loans$profit)
```

```
## [1] 32.87361
```

How many of 100 selected loans were not paid back in full?

```r
table(selected_loans$not.fully.paid)
```

```
##
##  0  1
## 82 18
```

We have now seen how analytics can be used to select a subset of the high-interest loans that were paid back at only a slightly lower rate than average, resulting in a significant increase in the profit from our investor's $100 investment. Although the logistic regression models developed in this problem did not have large AUC values, we see that they still provided the edge needed to improve the profitability of an investment portfolio.

We conclude with a note of warning. Throughout this analysis we assume that the loans we invest in will perform in the same way as the loans we used to train our model, even though our training set covers a relatively short period of time. If there is an economic shock like a large financial downturn, default rates might be significantly higher than those observed in the training set and we might end up losing money instead of profiting. Investors must pay careful attention to such risk when making investment decisions.

Explanation

selectedLoans can be constructed with the following code:

selectedLoans = subset(highInterest, predicted.risk <= cutoff)

You can check the number of elements with nrow(selectedLoans). The profit variable contains the profit for the $1 investment into each of the loans, so the following code computes the profit for all 100 loans:

sum(selectedLoans$profit)

The breakdown of whether each of the selected loans was fully paid can be computed with

table(selectedLoans$not.fully.paid)

We have now seen how analytics can be used to select a subset of the high-interest loans that were paid back at only a slightly lower rate than average, resulting in a significant increase in the profit from our investor's $100 investment. Although the logistic regression models developed in this problem did not have large AUC values, we see that they still provided the edge needed to improve the profitability of an investment portfolio.

We conclude with a note of warning. Throughout this analysis we assume that the loans we invest in will perform in the same way as the loans we used to train our model, even though our training set covers a relatively short period of time. If there is an economic shock like a large financial downturn, default rates might be significantly higher than those observed in the training set and we might end up losing money instead of profiting. Investors must pay careful attention to such risk when making investment decisions.