HW3

Yaqi Li

4/22/2019

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ tibble  2.0.1     ✔ purrr   0.3.0
## ✔ tidyr   0.8.3     ✔ dplyr   0.8.0.1
## ✔ readr   1.3.1     ✔ stringr 1.3.1
## ✔ tibble  2.0.1     ✔ forcats 0.3.0
```

```
## ── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

1. NBA predictions revisited

```
nba = read.csv('nbaspread.csv')
attach(nba)
str(nba)
```

```
## 'data.frame':    553 obs. of  7 variables:
##  $ favwin : int  1 1 1 0 0 1 1 1 0 1 ...
##  $ favscr : int  72 82 87 69 77 91 95 90 79 103 ...
##  $ undscr : int  61 74 57 70 79 65 88 67 80 68 ...
##  $ spread : num  7 7 17 9 2.5 9 10 18 7.5 8 ...
##  $ favhome: int  0 1 1 1 0 0 1 1 0 0 ...
##  $ fregion: int  3 3 3 3 2 3 3 4 3 2 ...
##  $ uregion: int  4 1 3 3 3 4 3 4 3 2 ...
```

We can fit the logistic regression model with several different variables and compare their AICs to choose the best combination.

```
nbaLr1 = glm(favwin ~ favscr, family = binomial, data = nba)
AIC(nbaLr1)
```

```
## [1] 485.9228
```

```
nbaLr2 = glm(favwin ~ spread-1, family = binomial, data = nba)
AIC(nbaLr2)
```

```
## [1] 529.9716
```

```
nbaLr3 = glm(favwin ~ favhome , family = binomial, data = nba)
AIC(nbaLr3)
```

```
## [1] 599.7383
```

```
nbaLr4 = glm(favwin ~ fregion , family = binomial, data = nba)
AIC(nbaLr4)
```

```
## [1] 609.4997
```

```
nbaLrAll = glm(favwin ~ . - undscr , family = binomial, data = nba)
AIC(nbaLrAll)
```

```
## [1] 444.5751
```

The result shows `favscore` has the most predictive power as a single variable. The best logistic regression model is the one incorporates all the variables but `undscr` . Be noticed that the favwin is deducted directly from `favhome` - `undscr` so we cannot have both variables in the model simultaneously。

```
nbaLr = glm(favwin ~ favhome + favscr + spread + fregion , family = binomial, data
= nba)
AIC(nbaLr)
```

```
## [1] 445.2505
```

It is more accurate when we incorporate all those variables to predict the outcome.

```
nbaTrain <- createDataPartition(nba$favwin, p=0.6, list=FALSE)
nbatraining <- nba[nbaTrain, ]
nbatesting <- nba[-nbaTrain, ]
nbaLogistic = glm(favwin ~ favhome + favscr + spread + fregion , family = binomial
, data = nbatraining)
nbatesting$model_prob <- predict(nbaLogistic, nbatesting, type = "response")
nbatesting = nbatesting %>%
  mutate(model_pred = 1*(model_prob > .53) + 0)
nbatesting <- nbatesting %>% mutate(accurate = 1*(model_pred == favwin))
sum(nbatesting$accurate)/nrow(nbatesting)
```

```
## [1] 0.8235294
```

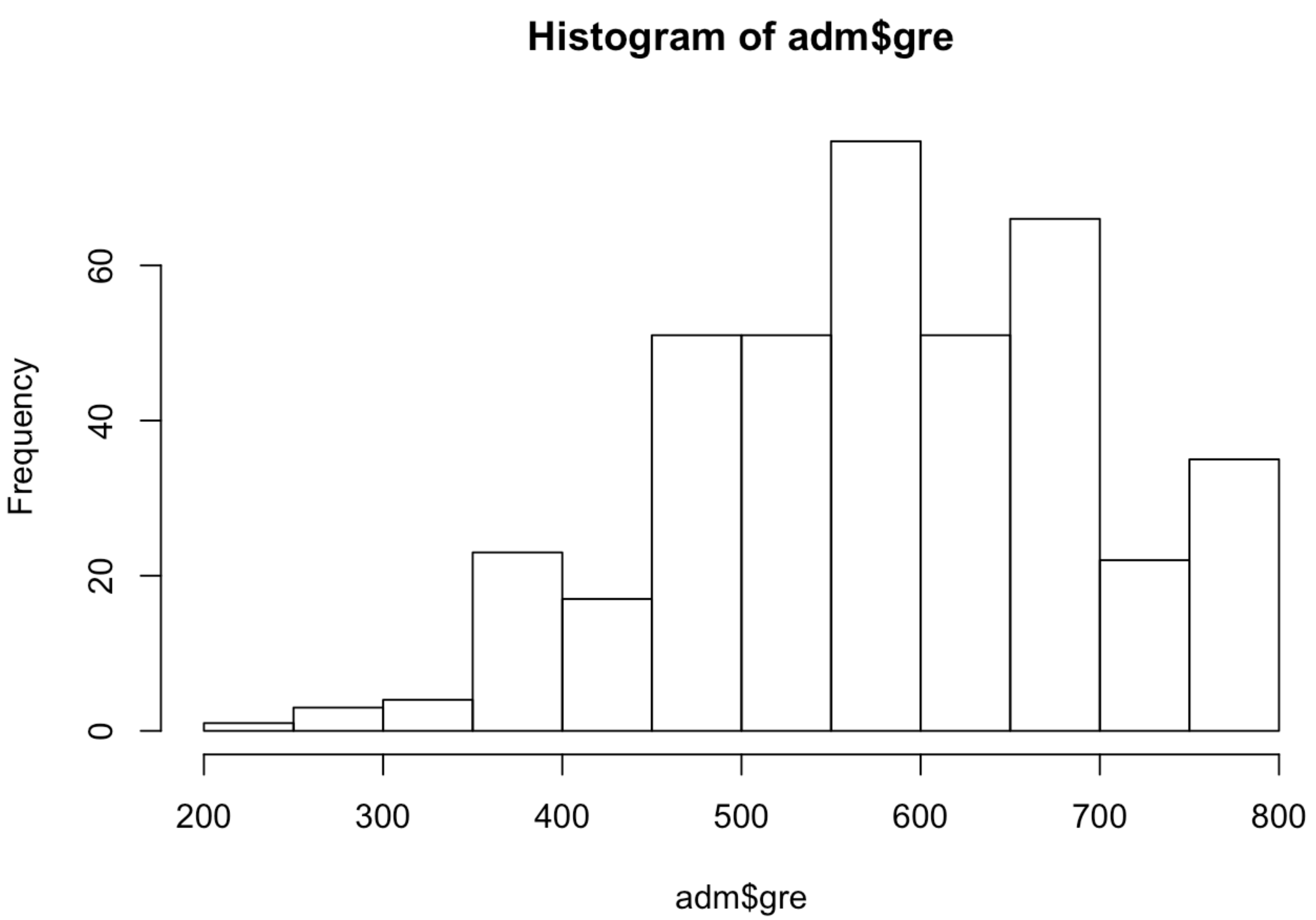
The accuracy of the logistic regression incorporating variables `favhome,favscr, spread, fregion` is 80%. ##2. Graduate school admissions

```
adm = read.csv('admissions.csv')
str(adm)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

We have a dependant binary variable `admit` indicating admitted or not. `gre` and `gpa` are continuous predictors and `rank` is a categorical variable taking on values through 1 to 4. 4 means the lowest prestigious rank.

```
hist(adm$gre)
```




```
mean(adm$gre)
```

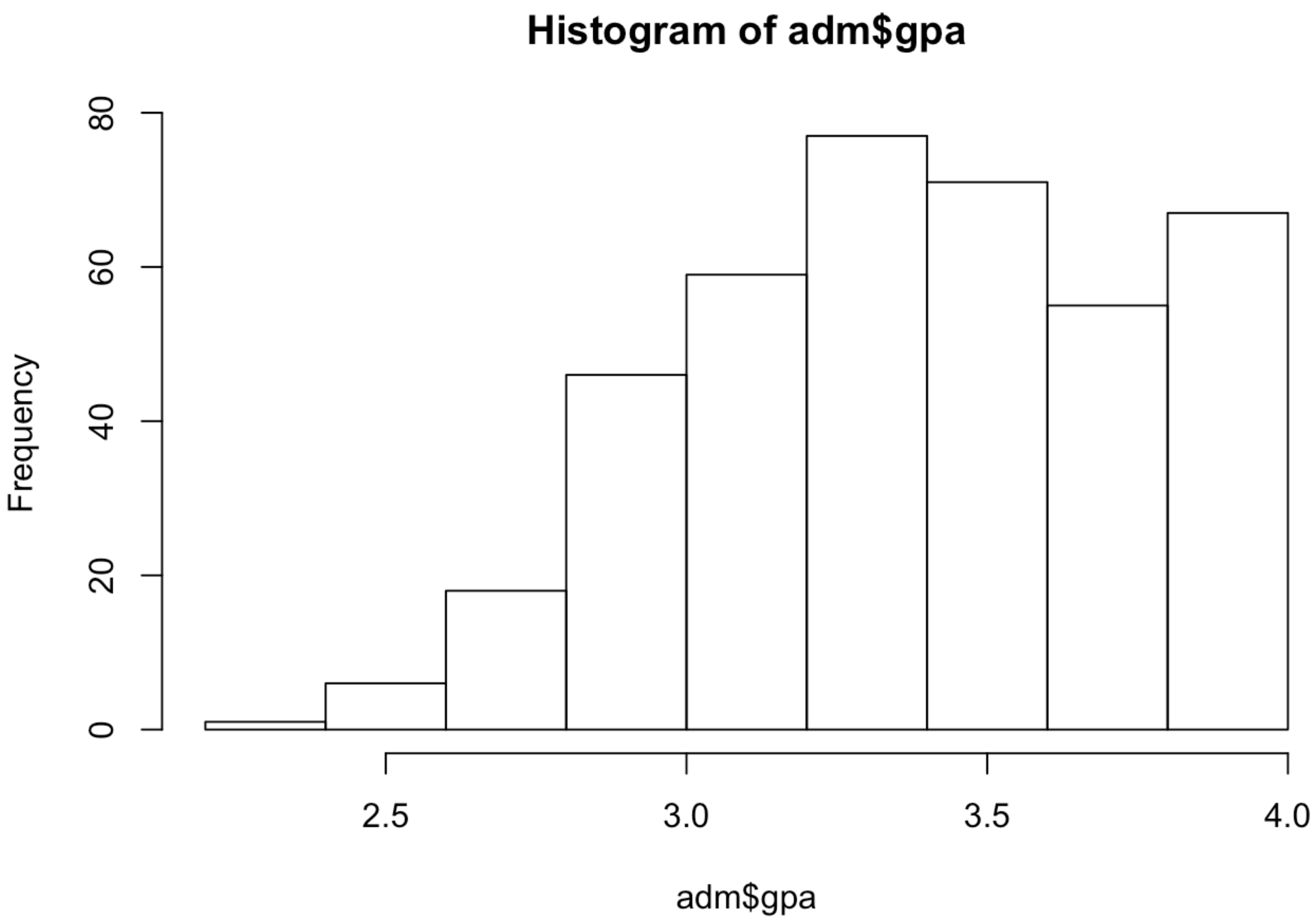
```
## [1] 587.7
```

```
sd(adm$gre)
```

```
## [1] 115.5165
```

Variable `Gre` has a mean of 587.7 with a standard deviation of 115.5.

```
hist(adm$gpa)
```




```
mean(adm$gpa)
```

```
## [1] 3.3899
```

```
sd(adm$gpa)
```

```
## [1] 0.3805668
```

Variable `gpa` has a mean of 3.39 with a standard deviation of 0.38. Both distribution of `gre` and `gpa` are right-skewed. Lets now take a look at the relationship between the two categorical values.

```
xtabs(~admit+rank,data = adm)
```

```
##      rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

This shows us the number of observations in 8 possible scenarios. Now we can predict the admission with these variables. But first we need to transform `rank` into factor.

```
adm$rank = as.factor(adm$rank)
str(adm)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
```

Now we can fit a logistic regression model with `glm`.

```
admLR = glm(admit ~ gre + gpa + rank , family = binomial, data = adm)
summary(admLR)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial, data = adm)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
##
## Number of Fisher Scoring iterations: 4
```

`gre` , `gpa` , and level 2,3, and 4 of `rank` are significant predictors. The coefficient of `rank3` means if the student comes from a university ranked 3rd, the *negative* coefficient would decrease his/her probability of getting admitted.

We can also transform the rank into an ordered factor.

```
adm$rank = as.ordered(adm$rank)
admLR2 = glm(admit ~ gre + gpa + rank , family = binomial, data = adm)
summary(admLR2)
```

```
## 
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial, data = adm)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.6268  -0.8662  -0.6388   1.1490   2.0790  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -4.881757   1.113111  -4.386 1.16e-05 ***
## gre          0.002264   0.001094   2.070   0.0385 *  
## gpa          0.804038   0.331819   2.423   0.0154 *  
## rank.L      -1.189399   0.287159  -4.142 3.44e-05 ***
## rank.Q       0.232092   0.251685   0.922   0.3564    
## rank.C       0.099017   0.212052   0.467   0.6405    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
## 
## Number of Fisher Scoring iterations: 4
```

The AIC are the same for both models however the model treats rank as a combination of linear, quadratic, and cubic formulas and only the first order of rank has shown significant predictive ability in the logistic model.

Now we can calculate the 95% confidence interval for each variable.

```
confint(admLR,level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept) -6.2716202334 -1.792547080
## gre          0.0001375921  0.004435874
## gpa          0.1602959439  1.464142727
## rank2       -1.3008888002 -0.056745722
## rank3       -2.0276713127 -0.670372346
## rank4       -2.4000265384 -0.753542605
```

Now we can predict the different admission probability by changing rank while holding gre and gpa at their mean value.

```
predict(admLR, data.frame(gre = 587, gpa = 3.39 , rank = factor(1)), type = "respo
nse")
```

```
##         1
## 0.5162258
```

```
predict(admLR, data.frame(gre = 587, gpa = 3.39 , rank = factor(2)), type = "respo
nse")
```

```
##         1
## 0.3519413
```

```
predict(admLR, data.frame(gre = 587, gpa = 3.39 , rank = factor(3)), type = "respo
nse")
```

```
##         1
## 0.2183551
```

```
predict(admLR, data.frame(gre = 587, gpa = 3.39 , rank = factor(4)), type = "respo
nse")
```

```
##        1
## 0.184442
```

As expected, the probability of getting admitted decreases as the rank increases, which means less prestigious university background.

We can also calculate the probability directly from the definition of the logistic function.

```
logistic = function(x){
  res = exp(x)/(1+exp(x))
  return(res)
}
# Predict the probability of admission by assigning the sum of linear combination
into logistic function
logistic(sum(admLR$coefficients*c(1,587,3.39,0,0,0)))
```

```
## [1] 0.5162258
```

```
logistic(sum(admLR$coefficients*c(1,587,3.39,1,0,0)))
```

```
## [1] 0.3519413
```

```
logistic(sum(admLR$coefficients*c(1,587,3.39,0,1,0)))
```

```
## [1] 0.2183551
```

```
logistic(sum(admLR$coefficients*c(1,587,3.39,0,0,1)))
```

```
## [1] 0.184442
```

The result is consistent with the `predict` method provided by R.

Conduct Chi-square test to see the effect of coefficients.

Null deviance: 499.98 on 399 degrees of freedom Residual deviance: 458.52 on 394 degrees of freedom

```
1 - pchisq(499.98-458.52,df = 399-394)
```

```
## [1] 7.574756e-08
```

The p-vale is very low, so that we can reject the null hypothesis that there is no difference between the null model and fitted model, which means the model is effective.

Admission probability as gre and rank vary

create a table that shows how admission probabilities vary at each rank level across gre scores from 200 to 800 (increment gre by 10) while holding gpa at its mean.

```
table = data.frame( matrix(nrow = 61))

for (k in c(1,2,3,4)) {
  temp = c()
  for (i in seq(200, 800, by=10)) {
  temp = append(temp,predict(admLR, data.frame(gre = i , gpa = 3.39 , rank = facto
r(k)), type = "response"))
  }
  table = cbind(table,temp)
}
table[1] = NULL
names(table ) = c(1,2,3,4)
row.names(table)  = seq(200, 800, by=10)
table
```

```
##             1         2         3          4
## 200 0.3075908 0.1843951 0.1041808 0.08604819
## 210 0.3124345 0.1878250 0.1063132 0.08784580
## 220 0.3173194 0.1913037 0.1084839 0.08967728
## 230 0.3222450 0.1948315 0.1106934 0.09154310
## 240 0.3272103 0.1984083 0.1129422 0.09344376
## 250 0.3322147 0.2020343 0.1152309 0.09537974
## 260 0.3372572 0.2057095 0.1175597 0.09735153
## 270 0.3423370 0.2094341 0.1199292 0.09935960
## 280 0.3474533 0.2132080 0.1223399 0.10140443
## 290 0.3526050 0.2170313 0.1247921 0.10348652
## 300 0.3577912 0.2209038 0.1272864 0.10560633
## 310 0.3630109 0.2248257 0.1298231 0.10776434
## 320 0.3682631 0.2287967 0.1324027 0.10996103
## 330 0.3735467 0.2328167 0.1350256 0.11219687
## 340 0.3788606 0.2368857 0.1376923 0.11447232
## 350 0.3842038 0.2410035 0.1404030 0.11678785
## 360 0.3895751 0.2451699 0.1431583 0.11914391
## 370 0.3949733 0.2493846 0.1459584 0.12154096
## 380 0.4003973 0.2536474 0.1488038 0.12397946
## 390 0.4058458 0.2579581 0.1516948 0.12645983
## 400 0.4113175 0.2623162 0.1546318 0.12898252
## 410 0.4168113 0.2667216 0.1576151 0.13154796
## 420 0.4223258 0.2711737 0.1606450 0.13415656
## 430 0.4278598 0.2756722 0.1637218 0.13680874
## 440 0.4334118 0.2802166 0.1668458 0.13950491
## 450 0.4389806 0.2848065 0.1700174 0.14224546
## 460 0.4445649 0.2894413 0.1732367 0.14503076
## 470 0.4501631 0.2941206 0.1765039 0.14786121
## 480 0.4557740 0.2988437 0.1798194 0.15073716
## 490 0.4613962 0.3036100 0.1831834 0.15365896
## 500 0.4670282 0.3084188 0.1865959 0.15662694
## 510 0.4726686 0.3132696 0.1900572 0.15964145
## 520 0.4783160 0.3181616 0.1935674 0.16270277
## 530 0.4839690 0.3230940 0.1971267 0.16581122
## 540 0.4896260 0.3280661 0.2007352 0.16896708
## 550 0.4952857 0.3330771 0.2043929 0.17217060
## 560 0.5009467 0.3381261 0.2080999 0.17542203
## 570 0.5066073 0.3432122 0.2118562 0.17872162
## 580 0.5122663 0.3483346 0.2156618 0.18206956
## 590 0.5179222 0.3534923 0.2195168 0.18546605
## 600 0.5235734 0.3586843 0.2234210 0.18891128
## 610 0.5292187 0.3639097 0.2273745 0.19240538
## 620 0.5348564 0.3691673 0.2313771 0.19594850
## 630 0.5404853 0.3744562 0.2354287 0.19954074
## 640 0.5461038 0.3797753 0.2395291 0.20318220
## 650 0.5517107 0.3851234 0.2436781 0.20687293
## 660 0.5573044 0.3904994 0.2478756 0.21061299
## 670 0.5628836 0.3959021 0.2521213 0.21440238
## 680 0.5684470 0.4013303 0.2564149 0.21824111
## 690 0.5739931 0.4067829 0.2607562 0.22212913
## 700 0.5795207 0.4122585 0.2651447 0.22606638
## 710 0.5850285 0.4177559 0.2695802 0.23005279
```

```
## 720 0.5905150 0.4232739 0.2740622 0.23408823
## 730 0.5959791 0.4288110 0.2785903 0.23817256
## 740 0.6014195 0.4343660 0.2831640 0.24230560
## 750 0.6068350 0.4399376 0.2877829 0.24648717
## 760 0.6122243 0.4455243 0.2924464 0.25071701
## 770 0.6175864 0.4511249 0.2971539 0.25499488
## 780 0.6229199 0.4567378 0.3019048 0.25932048
## 790 0.6282238 0.4623618 0.3066986 0.26369349
## 800 0.6334971 0.4679953 0.3115345 0.26811354
```

Credit Card Analysis

Consider a data set of 1000 bank customers. Banks must make decisions regarding whether to approve loans or not. If an applicant is a good credit risk then the cost of not approving a loan is the loss of potentially profitable business to the bank. If the applicant is a bad credit risk, then approving a loan exposes the bank to a significant default risk.

The German Credit Data contains data on 20 variables and the classification whether an ap- plicant is considered a Good or a Bad credit risk for 1000 loan applicants. Your task is to create a predictive model using this data to help make decisions about whether to approve loans to potential borrowers dependent upon the variables in the data set.

```
credit = read.csv("GermanCredit.csv")
##Delete the first index column and add a numeric colomun
credit = credit %>%
  mutate(X = NULL) %>%
  mutate(approval = if_else(Class == "Good",1,0))
```

```
trainIndex <- createDataPartition(credit$approval, p=0.6, list=FALSE)
creditTraining <- credit[trainIndex, ]
creditTesting <-credit[-trainIndex, ]
creditLogistic = glm(approval ~ . - Class , family = binomial, data = creditTraini
ng)
summary(creditLogistic)
```

```
##
## Call:
## glm(formula = approval ~ . - Class, family = binomial, data = creditTraining)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5977  -0.6284   0.3184   0.7070   2.2846
##
## Coefficients: (13 not defined because of singularities)
##                                          Estimate Std. Error z value
## (Intercept)                             9.650e+00  1.969e+00   4.900
## Duration                               -2.786e-02  1.209e-02  -2.305
## Amount                                 -1.761e-04  6.080e-05  -2.896
## InstallmentRatePercentage              -4.672e-01  1.200e-01  -3.895
## ResidenceDuration                       2.437e-01  1.183e-01   2.061
## Age                                     8.843e-03  1.235e-02   0.716
```

```
## NumberExistingCredits                     -3.434e-01  2.635e-01  -1.303
## NumberPeopleMaintenance                   -7.045e-01  3.207e-01  -2.197
## Telephone                                 -5.623e-01  2.836e-01  -1.983
## ForeignWorker                             -7.825e-01  8.864e-01  -0.883
## CheckingAccountStatus.lt.0                -1.489e+00  3.038e-01  -4.899
## CheckingAccountStatus.0.to.200            -9.669e-01  3.013e-01  -3.210
## CheckingAccountStatus.gt.200               2.839e-01  5.257e-01   0.540
## CheckingAccountStatus.none                        NA         NA      NA
## CreditHistory.NoCredit.AllPaid            -1.783e+00  6.212e-01  -2.870
## CreditHistory.ThisBank.AllPaid            -1.824e+00  6.118e-01  -2.981
## CreditHistory.PaidDuly                    -8.757e-01  3.437e-01  -2.548
## CreditHistory.Delay                       -5.603e-01  4.370e-01  -1.282
## CreditHistory.Critical                            NA         NA      NA
## Purpose.NewCar                            -8.020e-01  1.058e+00  -0.758
## Purpose.UsedCar                            8.581e-01  1.132e+00   0.758
## Purpose.Furniture.Equipment               -3.921e-01  1.072e+00  -0.366
## Purpose.Radio.Television                  -9.633e-02  1.060e+00  -0.091
## Purpose.DomesticAppliance                 -1.175e+00  1.517e+00  -0.774
## Purpose.Repairs                           -8.409e-01  1.223e+00  -0.688
## Purpose.Education                         -6.912e-01  1.139e+00  -0.607
## Purpose.Vacation                                  NA         NA      NA
## Purpose.Retraining                         1.267e+01  7.368e+02   0.017
## Purpose.Business                          -1.786e-01  1.089e+00  -0.164
## Purpose.Other                                     NA         NA      NA
## SavingsAccountBonds.lt.100                -1.315e+00  3.564e-01  -3.689
## SavingsAccountBonds.100.to.500            -1.055e+00  4.784e-01  -2.205
## SavingsAccountBonds.500.to.1000           -3.582e-02  6.641e-01  -0.054
## SavingsAccountBonds.gt.1000               -6.104e-01  6.686e-01  -0.913
## SavingsAccountBonds.Unknown                       NA         NA      NA
## EmploymentDuration.lt.1                   -4.201e-01  5.558e-01  -0.756
## EmploymentDuration.1.to.4                 -1.886e-01  5.282e-01  -0.357
## EmploymentDuration.4.to.7                  5.275e-01  5.768e-01   0.915
## EmploymentDuration.gt.7                   -1.797e-01  5.250e-01  -0.342
## EmploymentDuration.Unemployed                     NA         NA      NA
## Personal.Male.Divorced.Seperated          -7.520e-01  6.974e-01  -1.078
## Personal.Female.NotSingle                 -3.508e-01  4.433e-01  -0.791
## Personal.Male.Single                       4.632e-01  4.454e-01   1.040
## Personal.Male.Married.Widowed                     NA         NA      NA
## Personal.Female.Single                            NA         NA      NA
## OtherDebtorsGuarantors.None               -2.225e+00  7.936e-01  -2.804
## OtherDebtorsGuarantors.CoApplicant        -2.066e+00  9.607e-01  -2.151
## OtherDebtorsGuarantors.Guarantor                  NA         NA      NA
## Property.RealEstate                        1.452e+00  5.746e-01   2.527
## Property.Insurance                         1.112e+00  5.619e-01   1.980
## Property.CarOther                          1.059e+00  5.528e-01   1.915
## Property.Unknown                                  NA         NA      NA
## OtherInstallmentPlans.Bank                -1.002e+00  3.313e-01  -3.026
## OtherInstallmentPlans.Stores              -9.069e-01  4.863e-01  -1.865
## OtherInstallmentPlans.None                        NA         NA      NA
## Housing.Rent                              -1.288e+00  6.479e-01  -1.988
## Housing.Own                               -6.333e-01  6.096e-01  -1.039
## Housing.ForFree                                   NA         NA      NA
## Job.UnemployedUnskilled                    1.812e-01  8.381e-01   0.216
## Job.UnskilledResident                     -1.818e-01  4.755e-01  -0.382
```

```
## Job.SkilledEmployee                         1.193e-01  3.847e-01   0.310
## Job.Management.SelfEmp.HighlyQualified             NA         NA      NA
##                                            Pr(>|z|)
## (Intercept)                                9.59e-07 ***
## Duration                                   0.021186 *
## Amount                                     0.003783 **
## InstallmentRatePercentage                  9.82e-05 ***
## ResidenceDuration                          0.039328 *
## Age                                        0.474097
## NumberExistingCredits                      0.192458
## NumberPeopleMaintenance                    0.028029 *
## Telephone                                  0.047419 *
## ForeignWorker                              0.377389
## CheckingAccountStatus.lt.0                 9.61e-07 ***
## CheckingAccountStatus.0.to.200             0.001329 **
## CheckingAccountStatus.gt.200               0.589088
## CheckingAccountStatus.none                       NA
## CreditHistory.NoCredit.AllPaid             0.004108 **
## CreditHistory.ThisBank.AllPaid             0.002872 **
## CreditHistory.PaidDuly                     0.010841 *
## CreditHistory.Delay                        0.199802
## CreditHistory.Critical                           NA
## Purpose.NewCar                             0.448360
## Purpose.UsedCar                            0.448611
## Purpose.Furniture.Equipment                0.714461
## Purpose.Radio.Television                   0.927604
## Purpose.DomesticAppliance                  0.438715
## Purpose.Repairs                            0.491751
## Purpose.Education                          0.543913
## Purpose.Vacation                                 NA
## Purpose.Retraining                         0.986282
## Purpose.Business                           0.869683
## Purpose.Other                                    NA
## SavingsAccountBonds.lt.100                 0.000225 ***
## SavingsAccountBonds.100.to.500             0.027469 *
## SavingsAccountBonds.500.to.1000            0.956987
## SavingsAccountBonds.gt.1000                0.361273
## SavingsAccountBonds.Unknown                      NA
## EmploymentDuration.lt.1                    0.449738
## EmploymentDuration.1.to.4                  0.721018
## EmploymentDuration.4.to.7                  0.360443
## EmploymentDuration.gt.7                    0.732184
## EmploymentDuration.Unemployed                    NA
## Personal.Male.Divorced.Seperated           0.280901
## Personal.Female.NotSingle                  0.428789
## Personal.Male.Single                       0.298435
## Personal.Male.Married.Widowed                    NA
## Personal.Female.Single                           NA
## OtherDebtorsGuarantors.None                0.005049 **
## OtherDebtorsGuarantors.CoApplicant         0.031505 *
## OtherDebtorsGuarantors.Guarantor                 NA
## Property.RealEstate                        0.011509 *
## Property.Insurance                         0.047758 *
## Property.CarOther                          0.055439 .
```

```
## Property.Unknown                                        NA
## OtherInstallmentPlans.Bank                        0.002481 **
## OtherInstallmentPlans.Stores                      0.062164 .
## OtherInstallmentPlans.None                              NA
## Housing.Rent                                      0.046763 *
## Housing.Own                                       0.298918
## Housing.ForFree                                         NA
## Job.UnemployedUnskilled                           0.828849
## Job.UnskilledResident                             0.702191
## Job.SkilledEmployee                               0.756468
## Job.Management.SelfEmp.HighlyQualified                  NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 747.65  on 599  degrees of freedom
## Residual deviance: 511.79  on 551  degrees of freedom
## AIC: 609.79
##
## Number of Fisher Scoring iterations: 14
```

```
creditTesting$model_prob <- predict(creditLogistic, creditTesting, type = "respons
e")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
creditTesting = creditTesting %>%
  mutate(model_pred = 1*(model_prob > .53) + 0) %>%
  mutate(accuracy = 1*(model_pred == approval))
sum(creditTesting$accuracy)/nrow(creditTesting)
```

```
## [1] 0.745
```

Because too many variables are included in the logistic regression model, we need to conduct stepwise to find the best variable combination.

```
library(MASS)
```

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

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

```r
step.creditLogistic <- creditLogistic  %>% stepAIC(trace = FALSE)
AIC(step.creditLogistic)
```

```
## [1] 577.501
```

```r
summary(step.creditLogistic)
```

```
## 
## Call:
## glm(formula = approval ~ Duration + Amount + InstallmentRatePercentage + 
##     ResidenceDuration + NumberPeopleMaintenance + Telephone + 
##     CheckingAccountStatus.lt.0 + CheckingAccountStatus.0.to.200 + 
##     CreditHistory.NoCredit.AllPaid + CreditHistory.ThisBank.AllPaid + 
##     CreditHistory.PaidDuly + Purpose.NewCar + Purpose.UsedCar + 
##     SavingsAccountBonds.lt.100 + SavingsAccountBonds.100.to.500 + 
##     EmploymentDuration.4.to.7 + Personal.Male.Single + OtherDebtorsGuarantors.N
one + 
##     OtherDebtorsGuarantors.CoApplicant + Property.RealEstate + 
##     Property.Insurance + Property.CarOther + OtherInstallmentPlans.Bank + 
##     OtherInstallmentPlans.Stores + Housing.Rent, family = binomial, 
##     data = creditTraining)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.7641  -0.6531   0.3525   0.7005   2.2281  
## 
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         7.708e+00  1.161e+00   6.641 3.12e-11
## Duration                           -2.876e-02  1.150e-02  -2.500 0.012407
## Amount                             -1.750e-04  5.717e-05  -3.061 0.002207
## InstallmentRatePercentage          -4.481e-01  1.136e-01  -3.945 7.98e-05
## ResidenceDuration                   2.525e-01  1.061e-01   2.380 0.017332
## NumberPeopleMaintenance            -7.990e-01  3.034e-01  -2.633 0.008456
## Telephone                          -6.440e-01  2.550e-01  -2.525 0.011561
## CheckingAccountStatus.lt.0         -1.538e+00  2.801e-01  -5.490 4.02e-08
## CheckingAccountStatus.0.to.200     -1.060e+00  2.730e-01  -3.884 0.000103
## CreditHistory.NoCredit.AllPaid     -1.475e+00  5.651e-01  -2.609 0.009074
## CreditHistory.ThisBank.AllPaid     -1.347e+00  5.268e-01  -2.556 0.010577
## CreditHistory.PaidDuly             -4.830e-01  2.443e-01  -1.977 0.048069
## Purpose.NewCar                     -4.306e-01  2.538e-01  -1.697 0.089750
## Purpose.UsedCar                     1.335e+00  4.929e-01   2.709 0.006753
## SavingsAccountBonds.lt.100         -1.221e+00  2.806e-01  -4.352 1.35e-05
## SavingsAccountBonds.100.to.500     -9.830e-01  4.113e-01  -2.390 0.016845
## EmploymentDuration.4.to.7           7.442e-01  3.329e-01   2.236 0.025368
## Personal.Male.Single                8.285e-01  2.594e-01   3.194 0.001404
## OtherDebtorsGuarantors.None        -2.442e+00  7.547e-01  -3.236 0.001214
## OtherDebtorsGuarantors.CoApplicant -2.318e+00  9.011e-01  -2.572 0.010109
## Property.RealEstate                 9.782e-01  3.889e-01   2.515 0.011901
## Property.Insurance                  6.173e-01  3.604e-01   1.713 0.086771
## Property.CarOther                   5.767e-01  3.421e-01   1.686 0.091837
```

```
## OtherInstallmentPlans.Bank          -1.024e+00  3.122e-01  -3.279 0.001042
## OtherInstallmentPlans.Stores        -9.770e-01  4.686e-01  -2.085 0.037070
## Housing.Rent                        -8.053e-01  3.037e-01  -2.652 0.008013
##
## (Intercept)                         ***
## Duration                            *
## Amount                              **
## InstallmentRatePercentage           ***
## ResidenceDuration                   *
## NumberPeopleMaintenance             **
## Telephone                           *
## CheckingAccountStatus.lt.0          ***
## CheckingAccountStatus.0.to.200      ***
## CreditHistory.NoCredit.AllPaid      **
## CreditHistory.ThisBank.AllPaid      *
## CreditHistory.PaidDuly              *
## Purpose.NewCar                      .
## Purpose.UsedCar                     **
## SavingsAccountBonds.lt.100          ***
## SavingsAccountBonds.100.to.500      *
## EmploymentDuration.4.to.7           *
## Personal.Male.Single                **
## OtherDebtorsGuarantors.None         **
## OtherDebtorsGuarantors.CoApplicant *
## Property.RealEstate                 *
## Property.Insurance                  .
## Property.CarOther                   .
## OtherInstallmentPlans.Bank          **
## OtherInstallmentPlans.Stores        *
## Housing.Rent                        **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 747.65  on 599  degrees of freedom
## Residual deviance: 525.50  on 574  degrees of freedom
## AIC: 577.5
##
## Number of Fisher Scoring iterations: 5
```

The AIC of the stepwised model has decreased from 624.52 to 590.38. Now lets find out whether the prediction accuracy has improved.

```
creditTesting$model_probSW <- predict(step.creditLogistic, creditTesting, type = "
response")
creditTesting = creditTesting %>%
  mutate(model_predSW = 1*(model_probSW > .53) + 0) %>%
  mutate(accuracySW = 1*(model_predSW == approval))
sum(creditTesting$accuracySW)/nrow(creditTesting)
```

```
## [1] 0.725
```

The accuracy has increased from 0.77 to 0.7775 after we conducted the step-wise regression.