Case III: Logistic Regression, Prediction and ROC

The objective of this case is to get you understand logistic regression (binary classification) and some important ideas such as cross validation, ROC curve, cut-off probability. Code in this case is built upon lecture slides and Shaonan Tian's sample code.

Input and sample data

First load the credit scoring data. It is easy to load comma-separated values (CSV).

```
credit.data <- read.csv
("http://homepages.uc.edu/~maifg/7040/credit0.csv", header =
T)
```

Now split the data 90/10 as training/testing datasets:

```
subset <- sample(nrow(credit.data), nrow(credit.data) * 0.9)
credit.train = credit.data[subset, ]
credit.test = credit.data[-subset, ]
```

The training dataset has 63 variables, 4500 obs.

```
colnames(credit.train)
```

```
##  [1] "id"     "Y"      "X2"     "X3"     "X4"     "X5"
"X6"
##  [8] "X7"     "X8"     "X9"     "X10_2"  "X11_2"  "X12_2"
"X13_2"
## [15] "X14_2"  "X15_2"  "X15_3"  "X15_4"  "X15_5"  "X15_6"
"X16_2"
## [22] "X16_3"  "X16_4"  "X16_5"  "X16_6"  "X17_2"  "X17_3"
"X17_4"
## [29] "X17_5"  "X17_6"  "X18_2"  "X18_3"  "X18_4"  "X18_5"
"X18_6"
## [36] "X18_7"  "X19_2"  "X19_3"  "X19_4"  "X19_5"  "X19_6"
"X19_7"
## [43] "X19_8"  "X19_9"  "X19_10" "X20_2"  "X20_3"  "X20_4"
"X21_2"
## [50] "X21_3"  "X22_2"  "X22_3"  "X22_4"  "X22_5"  "X22_6"
"X22_7"
## [57] "X22_8"  "X22_9"  "X22_10" "X22_11" "X23_2"  "X23_3"
"X24_2"
```

Logistic Regression

Let's build a logistic regression model based on all X variables. Note id is excluded from the model.

```
credit.glm0 <- glm(Y ~ . - id, family = binomial,
credit.train)
```

You can view the result of the estimation:

```
summary(credit.glm0)
```

Note that there might be a warning message "*Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred*". This happens because of a problem called quasi-complete separation. You can learn more here or here. The offending variable is *X9*. We will choose to ignore the warning now. You can try fitting your model without *X9*.

The usual stepwise variable selection still works for logistic regression. **caution: this will take a very long time**.

```
credit.glm.step <- step(credit.glm0)
```

Or you can try model selection with BIC:

```
credit.glm.step <- step(credit.glm0, k = log(nrow
(credit.train)))
```

Prediction and Cross Validation Using Logistic Regression

Now suppose there are 2 models we want to test, one with all X variables(credit.glm0), and one with X3, X8 and X11_2(credit.glm1).

```
credit.glm1 <- glm(Y ~ X3 + X8 + X11_2, family = binomial,
credit.train)
AIC(credit.glm0)
```

```
## [1] 1696
```

```
AIC(credit.glm1)
```

```
## [1] 1863
```

```
BIC(credit.glm0)
```

```
## [1] 2093
```

```
BIC(credit.glm1)
```

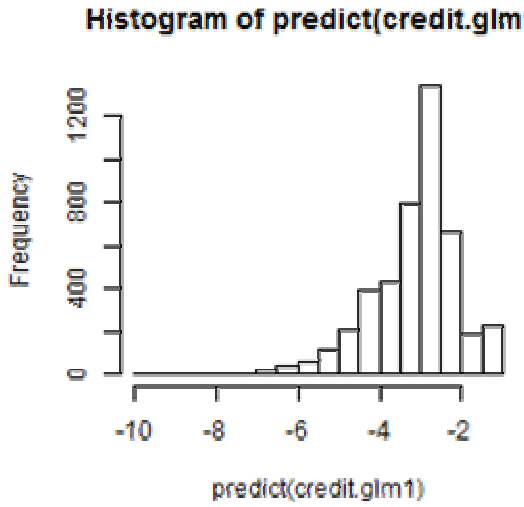
```
## [1] 1888
```

Understanding classification decision making using logistic regression

To get prediction from a logistic regression model, there are several steps you need to understand. Refer to textbook/slides for detailed math.

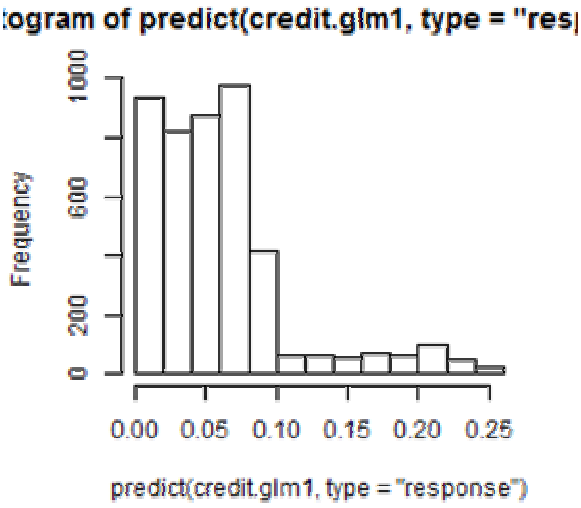
1.The fitted model $\hat{\eta} = b_0 + b_1 x_1 + b_2 x_2 + \ldots$ gives you the estimated value before the inverse of link (logit in case of logistic regression). In logistic regression the $\hat{\eta}$ are called **log odds ratio**, which is $\log(P(y=1)/(1-P(y=1)))$. In R you use the *predict()* function to get a vector of all in-sample \hat{\eta} (for each training ob).

```
hist(predict(credit.glm1))
```



2.For each $\hat{\eta}$, in order to get the P(y=1), we can apply the inverse of the link function (logit here) to $\hat{\eta}$. The equation is $P(y=1) = 1/(1+exp(-\hat{\eta}))$. In R you use the *fitted()* function or *predict(,type="response") to get the **predicted probability** for each training ob.

```
hist(predict(credit.glm1, type = "response"))
```



3.Last but not least, you want a binary classification decision rule. The default rule is if the fitted $P(y=1) > 0.5$ then $y = 1$. The value 0.5 is called **cut-off probability**. You can choose the cut-off probability based on mis-classification rate, cost function, etc. In this case, the cost function can indicate the trade off between the risk of giving loan to someone who cannot pay (predict 0, truth 1), and risk of rejecting someone who qualifys (predict 1, truth 0).

These tables illustrate the impact of choosing different cut-off probability. Choosing a large cut-off probability will result in few cases being predicted as 1, and chossing a small cut-off probability will result in many cases being predicted as 1.

```
table(predict(credit.glm1, type = "response") > 0.5)
```

```
##
## FALSE
##  4500
```

```
table(predict(credit.glm1, type = "response") > 0.2)
```

```
##
## FALSE  TRUE
##  4329   171
```

```
table(predict(credit.glm1, type = "response") > 1e-04)
```

```
##
## FALSE  TRUE
##     2  4498
```

In-sample and out-of-sample prediction

In-sample (performance on training set)

Suppose the cut-off probability is choosen as 0.2. The 2nd statement generates a logical vector (TRUE or FALSE) of whether each ob in training set has a fitted probability greater than 0.2. The 3rd statement transforms the logical vector to numeric (0 or 1).

```
prob.glm1.insample <- predict(credit.glm1, type = "response")
predicted.glm1.insample <- prob.glm1.insample > 0.2
predicted.glm1.insample <- as.numeric(predicted.glm1.insample)
```

Next we look at the confusion matrix, *dnn* is used to label the column and row:

```
table(credit.train$Y, predicted.glm1.insample, dnn = c
("Truth", "Predicted"))
```

```
##       Predicted
## Truth    0    1
##     0 4102  131
##     1  227   40
```

There are many ways to calculate the error rate. The following is one way. The *ifelse* function returns a vector, the elements are 1 if actual != predicted, 0 otherwise. *mean* gives you the percentage of 1s in the vector.

```
mean(ifelse(credit.train$Y != predicted.glm1.insample, 1, 0))
```

```
## [1] 0.07956
```

Out-of-sample (performance on testing set)

To do out-of-sample prediction you need to add the testing set as a second argument after the glm object. Remember to add type = “response”, otherwise you will get the log odds and not the probability.

```
prob.glm1.outsample <- predict(credit.glm1, credit.test, type
= "response")
predicted.glm1.outsample <- prob.glm1.outsample > 0.2
predicted.glm1.outsample <- as.numeric
(predicted.glm1.outsample)
table(credit.test$Y, predicted.glm1.outsample, dnn = c
("Truth", "Predicted"))
```

```
##       Predicted
## Truth   0   1
##     0 456  11
##     1  29   4
```

```
mean(ifelse(credit.test$Y != predicted.glm1.outsample, 1, 0))
```

```
## [1] 0.08
```

It is the same as:

```
glm1.outsample.logodds <- predict(credit.glm1, credit.test)
predicted.glm1.outsample <- exp(glm1.outsample.logodds)/(1 +
exp(glm1.outsample.logodds)) >
    0.2
predicted.glm1.outsample <- as.numeric
(predicted.glm1.outsample)
table(credit.test$Y, predicted.glm1.outsample, dnn = c
("Truth", "Predicted"))
mean(ifelse(credit.test$Y != predicted.glm1.outsample, 1, 0))
```

It is usually the case that your out-of-sample prediction error rate is higher than in-sample error rate.

ROC Curve

To get the ROC curve you need to install the verification library.

```
install.packages("verification")
```

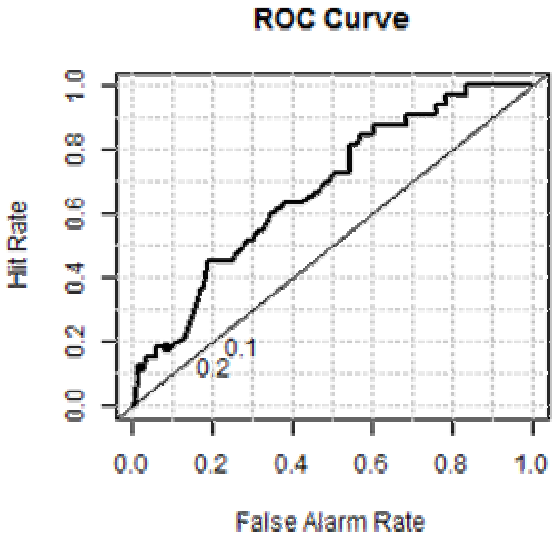
To plot the ROC curve, the first argument of roc.plot is the vector with actual values “A binary observation (coded {0, 1 })”. The second argument is the vector with predicted probability.

```
library("verification")
```

```
roc.plot(credit.test$Y == "1", prob.glm1.outsample)
```

To get the area under the ROC curve:

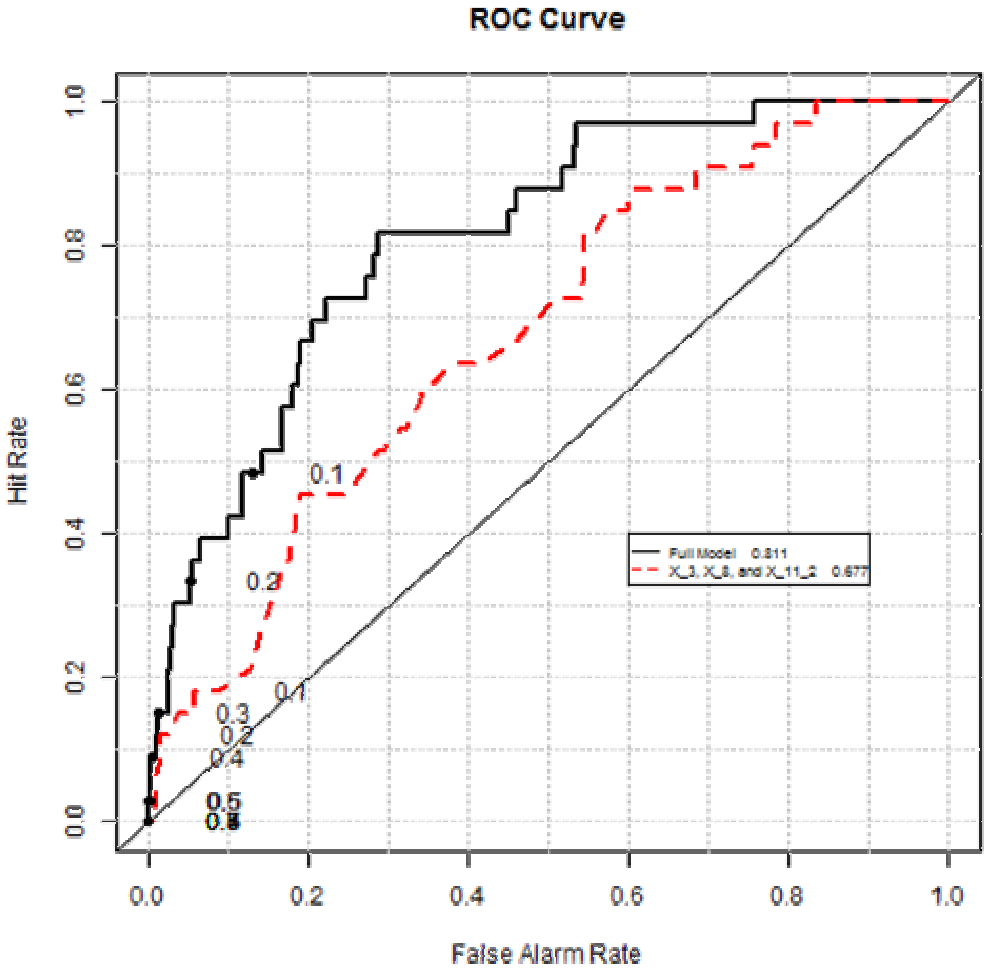
```
roc.plot(credit.test$Y == "1", prob.glm1.outsample)$roc.vol
```




```
##      Model   Area   p.value binorm.area
## 1 Model  1 0.6769 0.0003389          NA
```

We can also compare the glm0 and glm1 on the same graph:

```
prob.glm0.outsample <- predict(credit.glm0, credit.test, type
= "response")
roc.plot(x = credit.test$Y == "1", pred = cbind
(prob.glm0.outsample, prob.glm1.outsample),
    legend = TRUE, leg.text = c("Full Model", "X_3, X_8, and
X_11_2"))$roc.vol
```

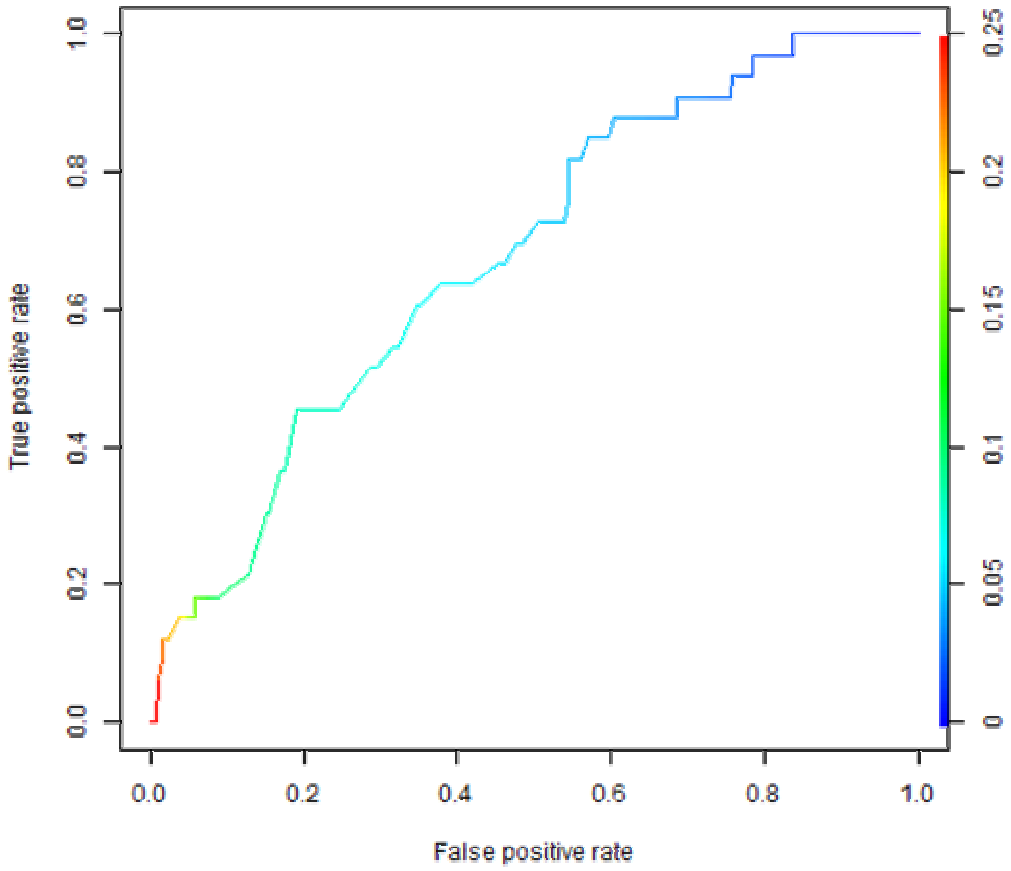



```
##      Model   Area   p.value binorm.area
## 1 Model  1 0.8113 1.112e-09          NA
## 2 Model  2 0.6769 3.389e-04          NA
```

Another library ROCR can also generate ROC curves for you.

```
install.packages("ROCR")
```

```
library(ROCR)
pred <- prediction(prob.glm1.outsample, credit.test$Y)
perf <- performance(pred, "tpr", "fpr")
plot(perf, colorize = TRUE)
```



Cross validation and cost function

Refer to lecture slides and Elements of Statistical Learning book (section 7.10) for more advice on cross validation. You should read the cv.glm help to understand how it works. First we define the cut-off probability and the define the cost function. *cost1* is a symmetric cost function, i.e. getting 1 wrong cost the same as getting 0 wrong. *cost2* is asymmetric cost function (we are double penalizing getting 1 wrong).

```
pcut = 0.2
# Symmetric cost
cost1 <- function(r, pi) {
    mean(((r == 0) & (pi > pcut)) | ((r == 1) & (pi < pcut)))
}
# Asymmetric cost
cost2 <- function(r, pi) {
    weight1 = 2
    weight0 = 1
    c1 = (r == 1) & (pi < pcut)  #logical vector - true if
actual 1 but predict 0
    c0 = (r == 0) & (pi > pcut)  #logical vecotr - true if
actual 0 but predict 1
    return(mean(weight1 * c1 + weight0 * c0))
}
```

10-fold cross validation, note we are using the **full data** to train and evaluate the model.

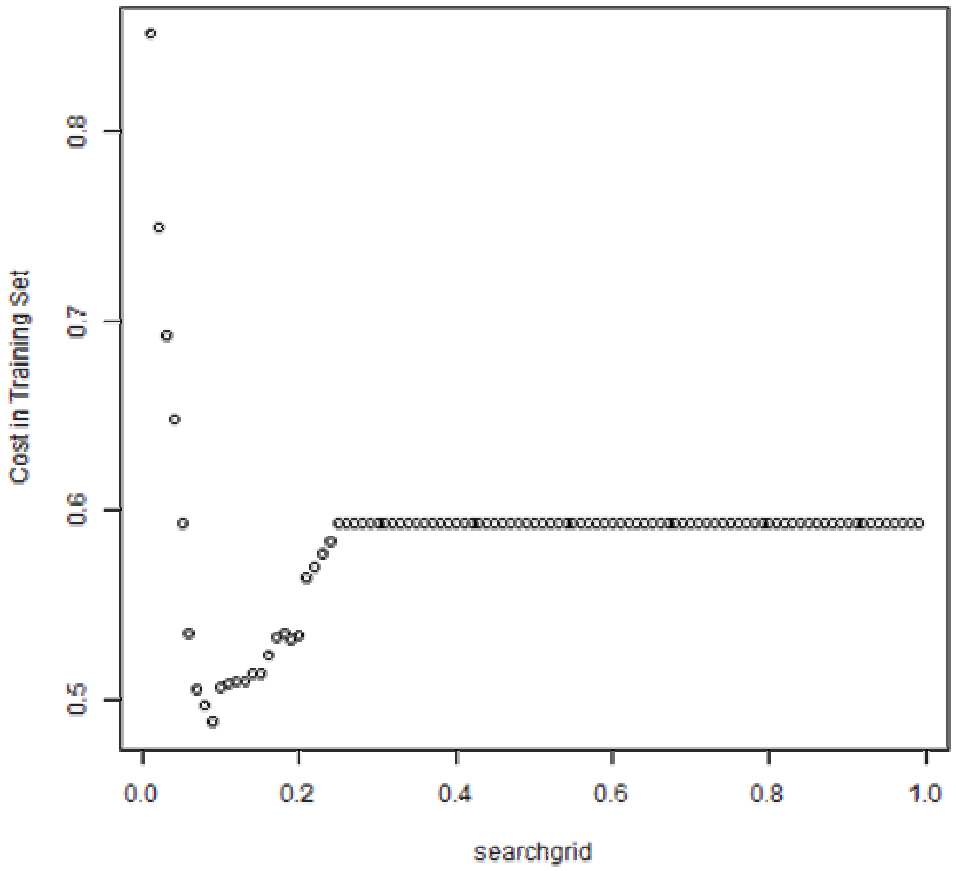
```
library(boot)
credit.glm3 <- glm(Y ~ X3 + X8 + X11_2, family = binomial,
credit.data)
cv.result = cv.glm(credit.data, credit.glm3, cost1, 10)
cv.result$delta
```

```
## [1] 0.07820 0.07814
```

Search for optimal cut-off probability

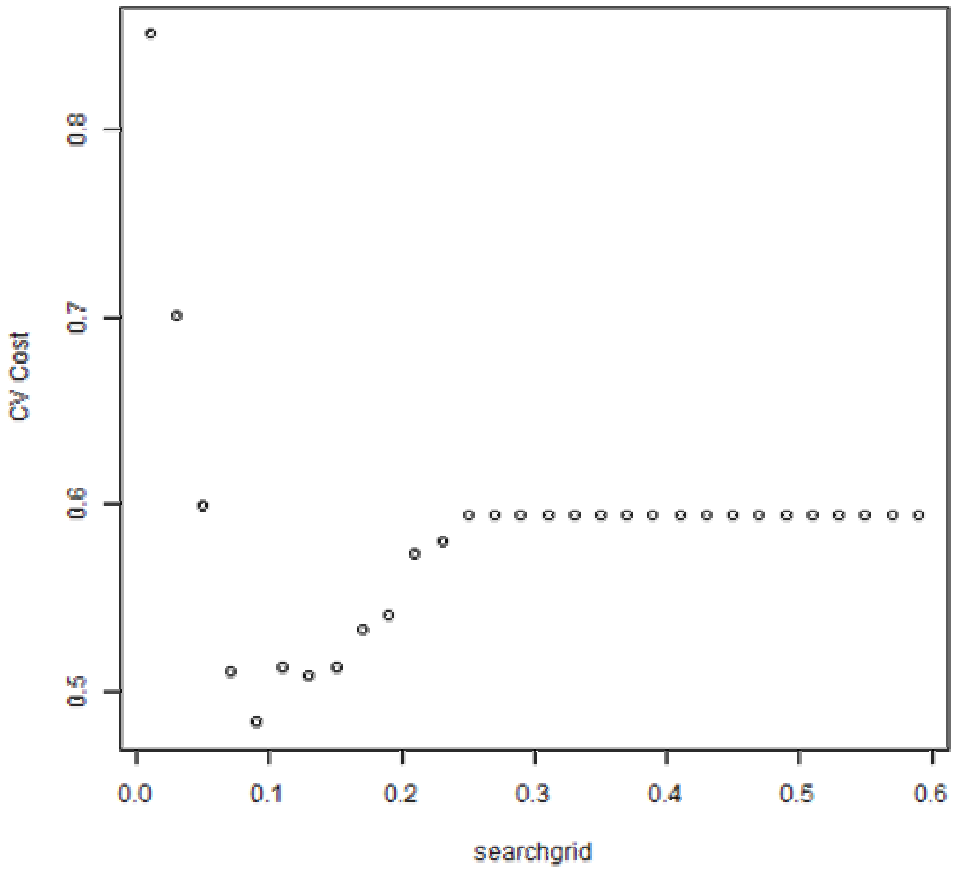
The following code does a grid search from pcut = 0.01 to pcut = 0.99 with the objective of minimizing overall cost in the training set. I am using an asymmetric cost function by assuming that giving out a bad loan cost 10 time as much as rejecting application from someone who can pay.

```
# define the searc grid from 0.01 to 0.99
searchgrid = seq(0.01, 0.99, 0.01)
# result is a 99x2 matrix, the 1st col stores the cut-off p,
the 2nd
# column stores the cost
result = cbind(searchgrid, NA)
# in the cost function, both r and pi are vectors, r=truth,
pi=predicted
# probability
cost1 <- function(r, pi) {
    weight1 = 10
    weight0 = 1
    c1 = (r == 1) & (pi < pcut)  #logical vector - true if
actual 1 but predict 0
    c0 = (r == 0) & (pi > pcut)  #logical vecotr - true if
actual 0 but predict 1
    return(mean(weight1 * c1 + weight0 * c0))
}
credit.glm1 <- glm(Y ~ X3 + X8 + X11_2, family = binomial,
credit.train)
for (i in 1:length(searchgrid)) {
    pcut <- result[i, 1]
    # assign the cost to the 2nd col
    result[i, 2] <- cost1(credit.train$Y, predict
(credit.glm1, type = "response"))
}
plot(result, ylab = "Cost in Training Set")
```



Alternatively, you can use the cross validation to choose the best cut-off probability.

```
searchgrid = seq(0.01, 0.6, 0.02)
result = cbind(searchgrid, NA)
cost1 <- function(r, pi) {
    weight1 = 10
    weight0 = 1
    c1 = (r == 1) & (pi < pcut)  #logical vector - true if
actual 1 but predict 0
    c0 = (r == 0) & (pi > pcut)  #logical vecotr - true if
actual 0 but predict 1
    return(mean(weight1 * c1 + weight0 * c0))
}
credit.glm1 <- glm(Y ~ X3 + X8 + X11_2, family = binomial,
credit.train)
for (i in 1:length(searchgrid)) {
    pcut <- result[i, 1]
    result[i, 2] <- cv.glm(data = credit.train, glmfit =
credit.glm1, cost = cost1,
        K = 3)$delta[2]
}
plot(result, ylab = "CV Cost")
```



Starter code for German credit scoring

Refer to http://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)) for variable description. Notice that “It is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1).” Define your cost function accordingly!

```
german_credit = read.table
("http://archive.ics.uci.edu/ml/machine-learning-
databases/statlog/german/german.data")

colnames(german_credit) = c("chk_acct", "duration",
"credit_his", "purpose",
    "amount", "saving_acct", "present_emp",
"installment_rate", "sex", "other_debtor",
    "present_resid", "property", "age", "other_install",
"housing", "n_credits",
    "job", "n_people", "telephone", "foreign", "response")

# orginal response coding 1= good, 2 = bad we need 0 = good, 1
= bad
german_credit$response = german_credit$response - 1
```