Linear Discriminant Analaysis (LDA)

MSBA7002: Business Statistics

Contents

Case Study I: Riding Mowers

A riding-mower manufacturer would like to find a way of classifying families in a city into those likely to purchase a riding mower and those not likely to buy one. A pilot random sample is undertaken of 12 owners and 12 nonowners in the city.

```
mower <- read.csv('RidingMowers.csv')
```

```
str(mower)
```

```
## 'data.frame':    24 obs. of  3 variables:
##  $ Income   : num  60 85.5 64.8 61.5 87 ...
##  $ Lot_Size : num  18.4 16.8 21.6 20.8 23.6 19.2 17.6 22.4 20 20.8 ...
##  $ Ownership: chr  "Owner" "Owner" "Owner" "Owner" ...
```

```
names(mower)
```

```
## [1] "Income"    "Lot_Size"  "Ownership"
```

```
summary(mower)
```

```
##       Income          Lot_Size      Ownership
##  Min.   : 33.00   Min.   :14.00   Length:24
##  1st Qu.: 52.35   1st Qu.:17.50   Class :character
##  Median : 64.80   Median :19.00   Mode  :character
##  Mean   : 68.44   Mean   :18.95
##  3rd Qu.: 83.10   3rd Qu.:20.80
##  Max.   :110.10   Max.   :23.60
```

Change `Ownership` to factor

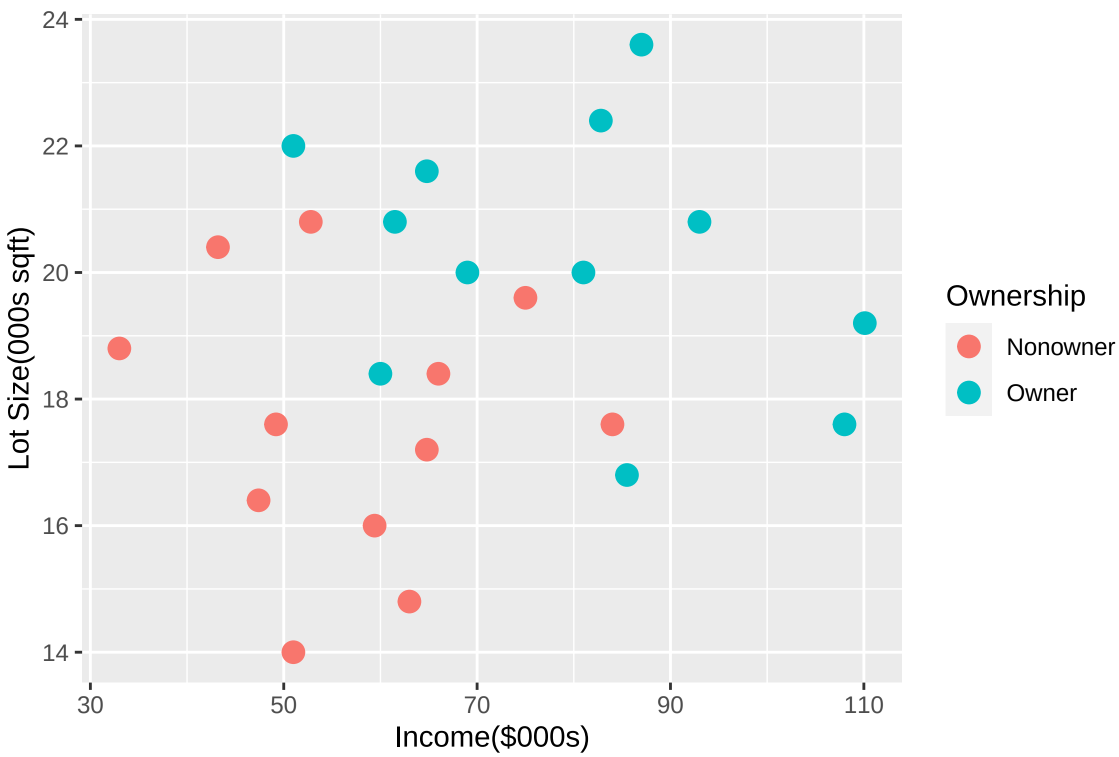
```
mower$Ownership <- factor(mower$Ownership)
```

EDA

Make a scatter plot for this dataset.

We can think of a linear classification rule as a line that separates the two-dimensional region into two parts, with most of the owners in one half-plane and most nonowners in the complementary half-plane.

```
  ggplot(mower) +
  geom_point(aes(x = Income, y = Lot_Size, col = Ownership), size = 3.5) +
  xlab("Income($000s)") +
  ylab("Lot Size(000s sqft)")
```



LDA

Classfication Function We use `DiscriMiner` to do LDA, which gives us Linear Discriminant Analysis function.

It will output the LDA model with classification error rate, confusion table.

```r
da.reg1 <- linDA(mower[,1:2], mower[,3])
#da.reg1 <- linDA(mower[,1:2], mower[,3],prior = c(1/2,1/2))
```

```r
names(da.reg1)
```

```
## [1] "functions"      "confusion"      "scores"         "classification"
## [5] "error_rate"     "specs"
```

```r
da.reg1$functions
```

```
##              Nonowner       Owner
## constant -51.4214500 -73.1602116
## Income     0.3293554   0.4295857
## Lot_Size   4.6815655   5.4667502
```

To classify a family into the class of owners or nonowners, we use the classification functions to compute the family's classification scores.

A family is classified into the class of owners if the owner function score is higher than the nonowner function score, and into nonowners if the reverse is the case.

$$\hat{\delta}(Nonnower|Income, Lot\_Size) = -51.42 + 0.3294 * Income + 4.682 * Lot\_Size$$
$$\hat{\delta}(Owner|Income, Lot\_Size) = -73.16 + 0.4296 * Income + 5.467 * Lot\_Size$$

An alternative way for classifying a record into one of the classes is to compute the probability of belonging to each of the classes and assigning the record to the most likely class.

$$P(Nonowner|Income, Lot\_Size) = \frac{\exp\{\hat{\delta}(Nonowner|Income, Lot\_Size)\}}{\exp\{\hat{\delta}(Owner|Income, Lot\_Size)\} + \exp\{\hat{\delta}(Nonowener|Income, Lot\_Size)\}}$$
$$P(Owner|Income, Lot\_Size) = \frac{\exp\{\hat{\delta}(Owner|Income, Lot\_Size)\}}{\exp\{\hat{\delta}(Owner|Income, Lot\_Size)\} + \exp\{\hat{\delta}(Nonowener|Income, Lot\_Size)\}}$$

```r
propensity.owner <- exp(da.reg1$scores[,2])/(exp(da.reg1$scores[,1])+exp(da.reg1$scores[,2]))
output1 <- data.frame(Actual=mower$Ownership,
           Pred=da.reg1$classification,
           da.reg1$scores,
           propensity.owner=propensity.owner)
output1
```

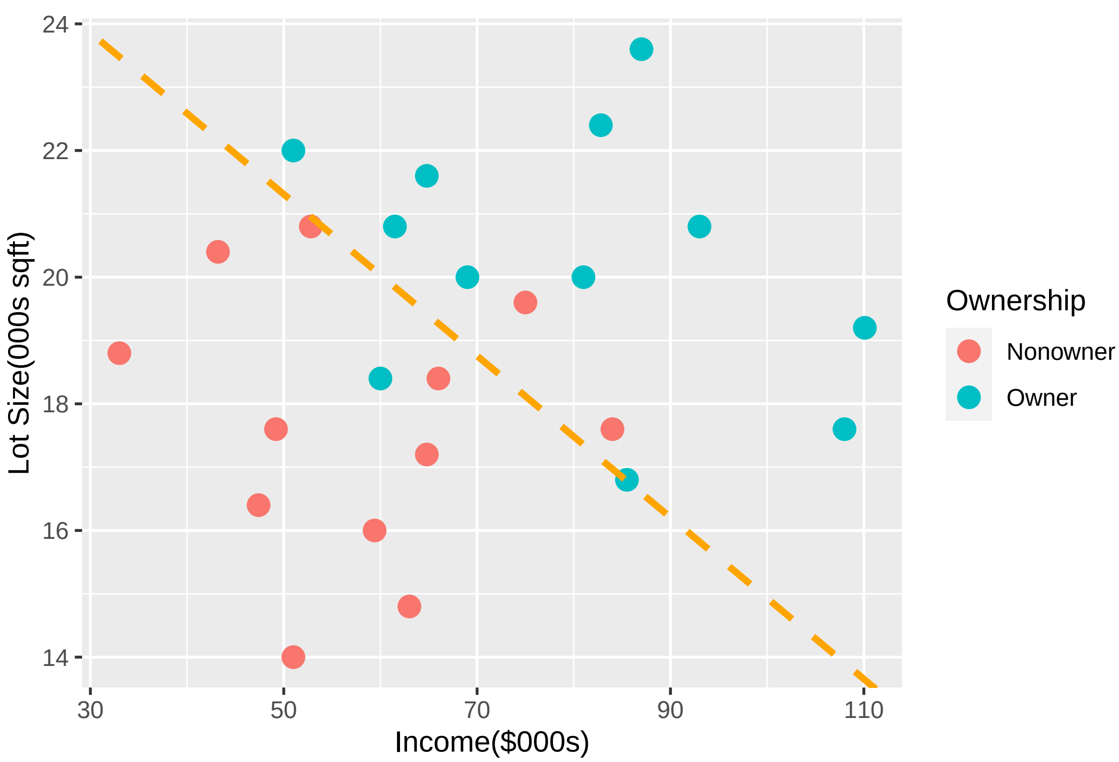
```
##      Actual     Pred Nonowner    Owner propensity.owner
## 1     Owner Nonowner 54.48068 53.20314      0.217968446
## 2     Owner    Owner 55.38874 55.41077      0.505507885
## 3     Owner    Owner 71.04260 72.75875      0.847632493
## 4     Owner    Owner 66.21047 66.96771      0.680755073
## 5     Owner    Owner 87.71742 93.22905      0.995976750
## 6     Owner    Owner 74.72664 79.09878      0.987533203
## 7     Owner    Owner 66.54449 69.44985      0.948110866
## 8     Owner    Owner 80.71625 84.86469      0.984456467
## 9     Owner    Owner 64.93538 65.81621      0.706992840
## 10    Owner    Owner 76.58517 80.49966      0.980439689
## 11    Owner    Owner 68.37012 69.01716      0.656344803
## 12    Owner    Owner 68.88765 70.97124      0.889297671
## 13 Nonowner    Owner 65.03889 66.20702      0.762807097
```

```
## 14 Nonowner Nonowner 63.34508 63.23032      0.471341530
## 15 Nonowner Nonowner 50.44371 48.70505      0.149483096
## 16 Nonowner Nonowner 58.31064 56.91960      0.199241067
## 17 Nonowner    Owner 58.63996 59.13979      0.622420495
## 18 Nonowner Nonowner 47.17839 44.19021      0.047962735
## 19 Nonowner Nonowner 43.04731 39.82518      0.038341510
## 20 Nonowner Nonowner 56.45681 55.78065      0.337118228
## 21 Nonowner Nonowner 40.96767 36.85685      0.016129950
## 22 Nonowner Nonowner 47.46071 43.79102      0.024851077
## 23 Nonowner Nonowner 30.91759 25.28316      0.003559993
## 24 Nonowner Nonowner 38.61511 34.81159      0.021806086
```

Classification Boundary We could also caculate the classification boundary.

$$\begin{aligned}
\hat{\delta}(Nonowner|Income, Lot_Size) &= \hat{\delta}(Owner|Income, Lot_Size) \\
-51.42 + 0.3294 * Income + 4.682 * Lot_Size &= -73.16 + 0.4296 * Income + 5.467 * Lot_Size \\
0.1002 * Income + 0.785 * Lot_Size &= 21.74 \\
Lot_Size &= 27.69 - 0.1276 * Income
\end{aligned}$$

```
ggplot(mower) +
  geom_point(aes(x = Income, y = Lot_Size, col = Ownership), size = 3.5) +
  geom_abline(intercept = 27.69, slope = -0.1276, color = 'orange', linetype= 'dashed', size = 1.2) +
  xlab("Income($000s)") +
  ylab("Lot Size(000s sqft)")
```



Prediction We use `classify` to do a prediction. For instance, the first household has an income of \$60K and a lot size of 18.4K ft^2.

```r
newmower <- mower[1,]
newmower[1] <- 60
newmower[2] <- 18.4
newmower[3] <- 'NA'
newmower
```

```
##   Income Lot_Size Ownership
## 1     60     18.4        NA
```

```r
pred1 <- classify(da.reg1,as.vector(newmower[1:2]))
pred1
```

```
## $scores
##   Nonowner    Owner
## 1 54.48068 53.20314
##
## $pred_class
## [1] Nonowner
## Levels: Nonowner Owner
```

$$\hat{\delta}(Nonowner|Income, Lot\_Size) = -51.42 + 0.3294 * 60 + 4.682 * 18.4 = 54.48$$
$$\hat{\delta}(Owner|Income, Lot\_Size) = -73.16 + 0.4296 * 60 + 5.467 * 18.4 = 53.20$$

Below we discussion some concepts related to classification.

a. Sensitivity

$$Prob(\hat{Y} = 1|Y = 1)$$

Not an error. This is also called `True Positive Rate`: the proportion of corrected positive classification given the status being positive.

b. Specificity

$$Prob(\hat{Y} = 0|Y = 0)$$

`Specificity`: the proportion of corrected negative classification given the status being negative.

c. False Positive

$$1 - Specificity = P(\hat{Y} = 1|Y = 0)$$

`False Positive`: the proportion of wrong classifications among given the status being negative.

d. Misclassification error

Mean value of missclassifications:

$$MCE = \frac{1}{n}\sum\{\hat{y}_i \neq y_i\}.$$

We can get all these quantities through confusion matrix or directly find the misclassification errors.

e. Confusion Matrix

We could use `table` to to create a 2 by 2 table which summarizes the number of mis/agreed labels.

```
table(mower$Ownership, da.reg1$classification)
```

```
##
##            Nonowner Owner
##   Nonowner       10     2
##   Owner           1    11
```

We could use `$confusion` to get the confusion matrix.

```
da.reg1$confusion
```

```
##           predicted
## original   Nonowner Owner
##   Nonowner       10     2
##   Owner           1    11
```

We could also use `confusionMatrix` function from `caret` packages to summarize the number of mis/agreed labels.

```
da.reg1$classification
```

```
##  [1] Nonowner Owner    Owner    Owner    Owner    Owner    Owner    Owner
##  [9] Owner    Owner    Owner    Owner    Owner    Nonowner Nonowner Nonowner
## [17] Owner    Nonowner Nonowner Nonowner Nonowner Nonowner Nonowner Nonowner
## Levels: Nonowner Owner
```

```
confusionMatrix(da.reg1$classification, mower$Ownership, positive = 'Owner')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Nonowner Owner
##   Nonowner       10     1
##   Owner           2    11
##
##                Accuracy : 0.875
##                  95% CI : (0.6764, 0.9734)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0001386
##
##                   Kappa : 0.75
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.9167
##             Specificity : 0.8333
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.9091
##              Prevalence : 0.5000
##          Detection Rate : 0.4583
##    Detection Prevalence : 0.5417
##       Balanced Accuracy : 0.8750
##
##        'Positive' Class : Owner
```

```
##
# first argument is prediction and the second one is reference
```

f. The Roc Curve and AUC

For each model or process, given a threshold, or a classifier, there will be a pair of sensitivity and specificity.

By changing the threshold, we graph all the pairs of False Positive as x-axis and True Positive as y-axis to have a curve: the ROC curve.

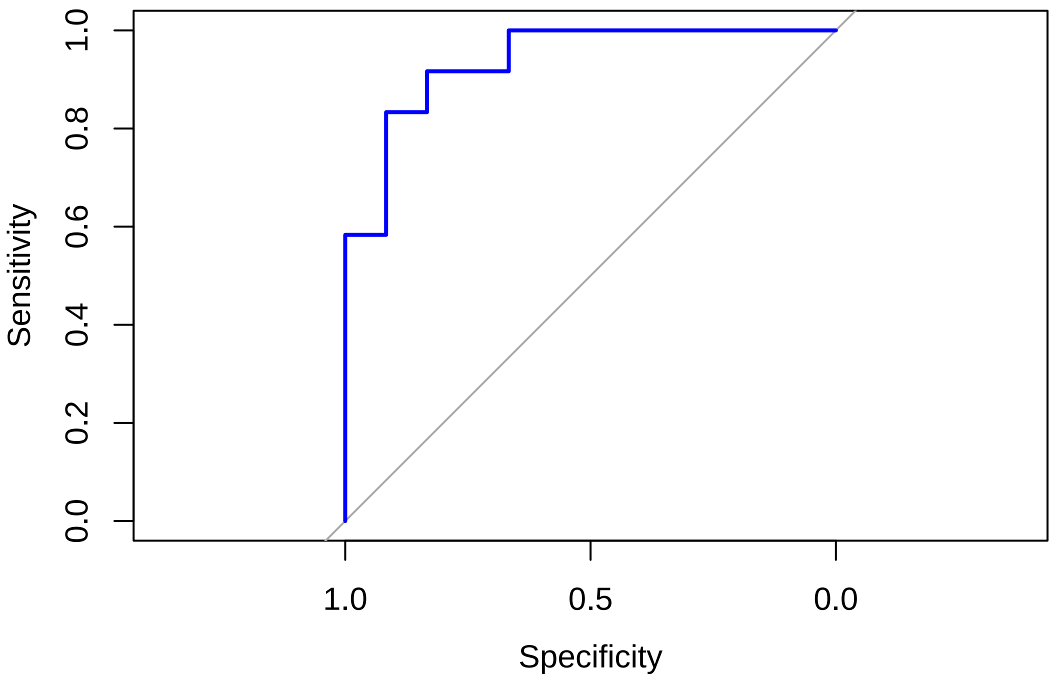
We use the `roc` function from the package `pROC`.

Notice that the ROC curve here is Sensitivity vs Specificity. Most of the ROC is drawn using False Positive rate as x-axis.

```
fit.roc1 <- roc(mower$Ownership, output1$propensity.owner, plot = T, col = 'blue')
```

```
## Setting levels: control = Nonowner, case = Owner
```

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



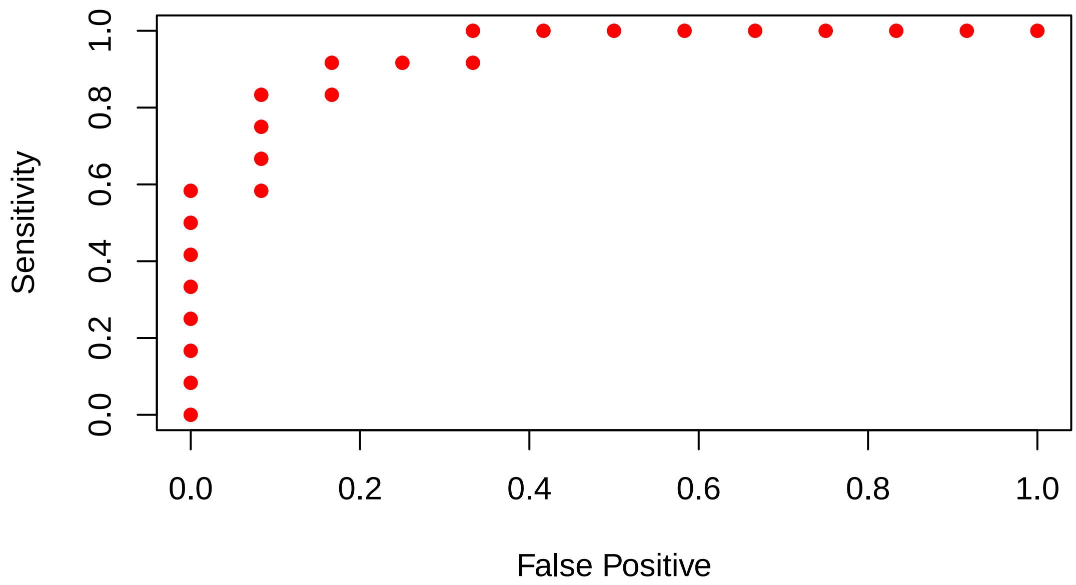

```
fit.roc1$auc
```

```
## Area under the curve: 0.9375
```

```
#auc(fit.roc1)
```

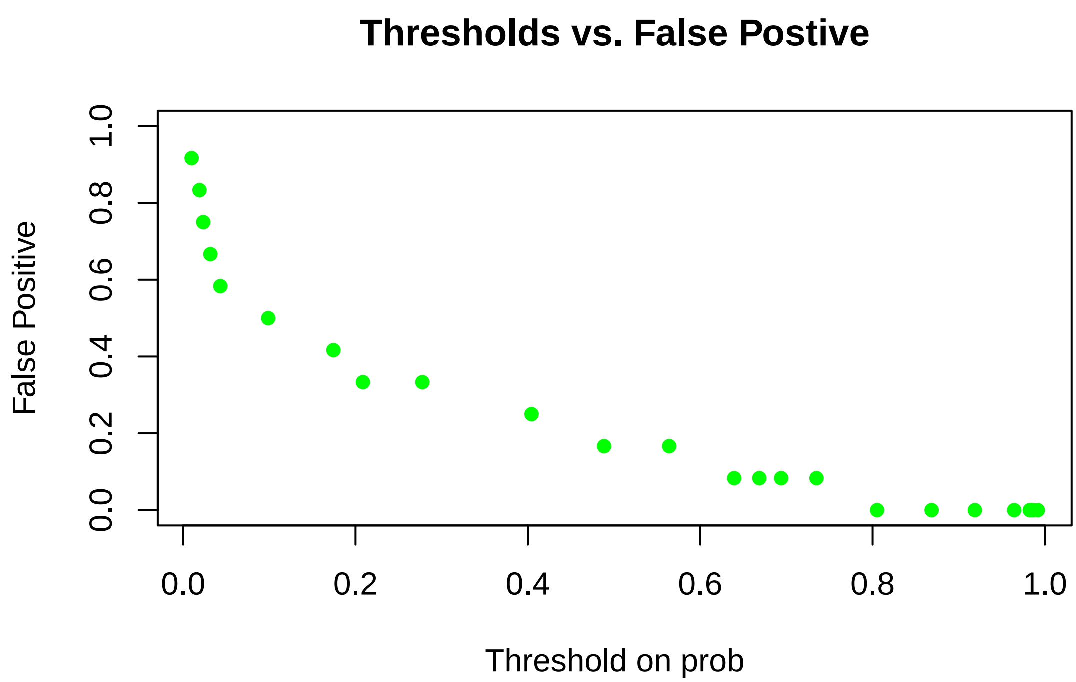
False Positive vs Sensitivity curve is plotted in most ROC curves:

```
plot(1-fit.roc1$specificities, fit.roc1$sensitivities, col="red", pch=16,
     xlab="False Positive",
     ylab="Sensitivity")
```



We can get more from fit.roc1. For example, a curve shows the probability thresholds used and the corresponding False Positive rate.

```
plot(fit.roc1$thresholds, 1-fit.roc1$specificities,  col="green", pch=16,
     xlab="Threshold on prob",
     ylab="False Positive",
     main = "Thresholds vs. False Postive")
```



g. Positive Prediction

Positive Prediction is a measure of the accuracy given the predictions.

Positive Prediction $= P(\text{Positive}|\text{ClassifiedasPositive})$

For da.reg1, recall the confusion matrix being

```r
cm.1 <- table(mower$Ownership, da.reg1$classification)
cm.1
```

```
##
##             Nonowner Owner
##   Nonowner       10     2
##   Owner           1    11
```

```r
positive.pred <- cm.1[2, 2] / (cm.1[1, 2] + cm.1[2, 2])
positive.pred
```

```
## [1] 0.8461538
```

h. Negative Prediction

Negative Prediction $= P(\text{Negative}|\text{ClassifiedasNegative})$

```r
negative.pred <- cm.1[1, 1] / (cm.1[1, 1] + cm.1[2, 1])
negative.pred
```

```
## [1] 0.9090909
```

Case Study II: Personal Loan Acceptance

The riding mowers example is a classic example and is useful in describing the concept and goal of discriminant analysis.

However, in today's business applications, the number of records is much larger, and their separation into classes is much less distinct.

To illustrate this, we consider the Universal Bank example, where the bank's goal is to identify new customers most likely to accept a personal loan.

In this case, we will use `Age`, `Experience`, `Income`, `Family`, `CCAvg`, `Education`, `Mortage`, `Securities.Account`, `CD.Account`, `Online`, `CredictCard` to predict `Personal.Loan`, personal loan acceptance situation.

```r
bank <- read.csv("UniversalBank.csv")
bank <- bank[,-c(1,5)] #Drop ID and zip code columns
bank <- bank %>% mutate(Personal.Loan = as.factor(Personal.Loan),
                        Education = as.factor(Education))
```

```r
str(bank)
```

```
## 'data.frame':    5000 obs. of  12 variables:
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal.Loan     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

```r
names(bank)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Education"
##  [7] "Mortgage"           "Personal.Loan"      "Securities.Account"
## [10] "CD.Account"         "Online"             "CreditCard"
```

```
summary(bank)
```

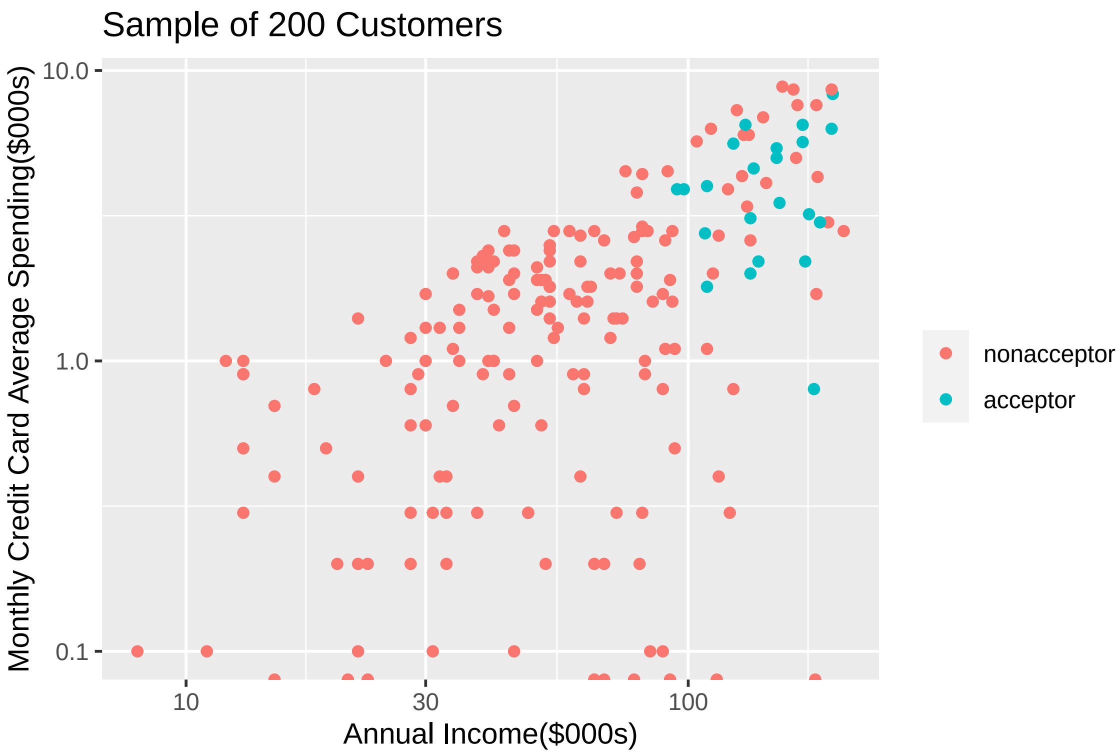
```
##       Age          Experience       Income           Family
##  Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :1.000
##  1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:1.000
##  Median :45.00   Median :20.0   Median : 64.00   Median :2.000
##  Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :2.396
##  3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:3.000
##  Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :4.000
##      CCAvg        Education    Mortgage      Personal.Loan Securities.Account
##  Min.   : 0.000   1:2096    Min.   :  0.0   0:4520        Min.   :0.0000
##  1st Qu.: 0.700   2:1403    1st Qu.:  0.0   1: 480        1st Qu.:0.0000
##  Median : 1.500   3:1501    Median :  0.0                 Median :0.0000
##  Mean   : 1.938             Mean   : 56.5                 Mean   :0.1044
##  3rd Qu.: 2.500             3rd Qu.:101.0                 3rd Qu.:0.0000
##  Max.   :10.000             Max.   :635.0                 Max.   :1.0000
##    CD.Account         Online         CreditCard
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :1.0000   Median :0.000
##  Mean   :0.0604   Mean   :0.5968   Mean   :0.294
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000
```

EDA

For simplicity, we will consider only two predictor variables:

the customer's annual income (Income, in $000s), and the average monthly credit card spending (CCAvg, in $000s).

```
set.seed(1101)
bank[sample(5000,200,replace = FALSE),] %>%
  ggplot(aes(x = Income, y= CCAvg, col=Personal.Loan)) +
  geom_point() +
  scale_colour_hue(name=NULL,labels=c('nonacceptor','acceptor')) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = 'Sample of 200 Customers') +
  xlab("Annual Income($000s)") +
  ylab("Monthly Credit Card Average Spending($000s)")
```



The first figure shows the acceptance of a personal loan by a subset of 200 customers from the bank's database as a function of Income and CCAvg.

We use a logarithmic scale on both axes to enhance visibility because there are many points condensed in the low-income, low-CC spending area. Even for this small subset, the separation is not clear.

```
bank %>%
  ggplot(aes(x = Income, y= CCAvg, col=Personal.Loan)) +
  geom_point() +
  scale_colour_hue(name=NULL,labels=c('nonacceptor','acceptor')) +
  scale_x_log10() +
  scale_y_log10() +
  labs(title = 'All 5000 Customers') +
  xlab("Annual Income($000s)") +
  ylab("Monthly Credit Card Average Spending($000s)")
```

The second figure shows all 5000 customers and the added complexity of dealing with large numbers of records.

Training/Testing Error

In order to evaluate the performance of each procedure, we need to estimate errors using unseen data.

Split the data to two sub-samples. We use Training Data to fit a model and use Testing Data to estimate the performance. Then choose the model with the larger AUC.

```
bank.x <- model.matrix(Personal.Loan~.,bank)[,-1]
bank.y <- bank$Personal.Loan
```

```
set.seed(7002)
index.train <- sample(5000,4000) # Sample 4000 out of 5000 as training dataset
bank.x.train <- bank.x[index.train,]
bank.x.test <- bank.x[-index.train,]
bank.y.train <- bank.y[index.train]
bank.y.test <- bank.y[-index.train]
```

```
dim(bank.x.train)
```

```
## [1] 4000   12
```

```
dim(bank.x.test)
```

```
## [1] 1000   12
```

```
colnames(bank.x.train)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Education2"
##  [7] "Education3"         "Mortgage"           "Securities.Account"
## [10] "CD.Account"         "Online"             "CreditCard"
```

```r
da.reg2.1 <- linDA(bank.x.train[,c(3,5)],bank.y.train)
# Only use Income and CCAvg as our predictors
```

Get `AUC` in test dataset of the model2.1

```r
pred2.1 <- classify(da.reg2.1, bank.x.test[,c(3,5)])
prob2.1 <- exp(pred2.1$scores[,2])/(exp(pred2.1$scores[,1])+exp(pred2.1$scores[,2]))
roc(bank.y.test, prob2.1)$auc
```

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

## Setting direction: controls < cases

## Area under the curve: 0.9347
```

We add another one variable into LDA model, calculate the `AUC` of test dataset for each model.

We only select `Income` and `CCAvg` as our predictors, as adding another variable does not significantly change `AUC`.

```r
cmp <- data.frame(matrix(0, nrow = 1, ncol = 4))
cmp[1,1] <- 'Income'
cmp[1,2] <- 'CCAvg'
cmp[1,3] <- ''
cmp[1,4] <- round(roc(bank.y.test, prob2.1)$auc,4)
```

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

## Setting direction: controls < cases
```

```r
colnames(cmp) <- c('Var1','Var2','Var3','AUC')
for (i in c(1,2,4,6:12)){
  da.reg2.2 <- linDA(bank.x.train[,c(3,5,i)],bank.y.train)
  pred2.2 <- classify(da.reg2.2,bank.x.test[,c(3,5,i)])
  prob2.2 <- exp(pred2.2$scores[,2])/(exp(pred2.2$scores[,1])+exp(pred2.2$scores[,2]))
  cmp <- rbind(cmp,c(colnames(bank.x)[c(3,5,i)],round(roc(bank.y.test, prob2.2)$auc,4)))
}
```

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

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```

```r
cmp
```

```
##      Var1  Var2              Var3    AUC
## 1  Income CCAvg                   0.9347
## 2  Income CCAvg               Age  0.935
## 3  Income CCAvg        Experience  0.935
## 4  Income CCAvg            Family 0.9436
## 5  Income CCAvg        Education2 0.9382
## 6  Income CCAvg        Education3  0.941
## 7  Income CCAvg          Mortgage  0.935
## 8  Income CCAvg Securities.Account 0.9343
## 9  Income CCAvg        CD.Account 0.9361
## 10 Income CCAvg            Online 0.9346
## 11 Income CCAvg        CreditCard 0.9346
```

```r
bank %>%
  ggplot(aes(x = Income, y= CCAvg, col=Personal.Loan)) +
  geom_point() +
  scale_colour_hue(name=NULL,labels=c('nonacceptor','acceptor')) +

  scale_x_log10() +
  scale_y_log10() +
  labs(title = 'All 5000 Customers') +
  xlab("Annual Income($000s)") +
  ylab("Monthly Credit Card Average Spending($000s)")
```

Final Model

```r
# Using all data
da.reg2 <- linDA(bank.x[,c(3,5)],bank.y)
da.reg2$functions
```

```
##                    0           1
## constant -1.49421630 -9.02898339
## Income     0.03947707  0.08458777
## CCAvg      0.09931842  0.28868702
```

```r
confusionMatrix(da.reg2$classification,bank$Personal.Loan)
```

Confussion Matrix

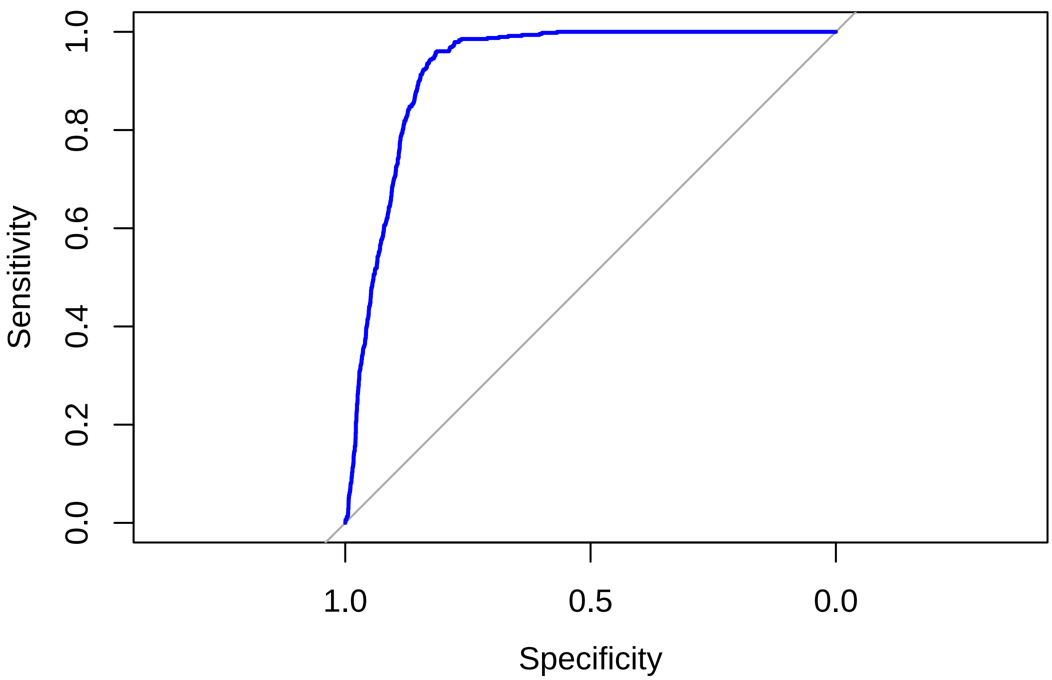
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4284  258
##          1  236  222
##
##                Accuracy : 0.9012
##                  95% CI : (0.8926, 0.9093)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.7579
##
##                   Kappa : 0.4189
##
```

```
##  Mcnemar's Test P-Value : 0.3447
##
##             Sensitivity : 0.9478
##             Specificity : 0.4625
##          Pos Pred Value : 0.9432
##          Neg Pred Value : 0.4847
##              Prevalence : 0.9040
##          Detection Rate : 0.8568
##    Detection Prevalence : 0.9084
##       Balanced Accuracy : 0.7051
##
##        'Positive' Class : 0
##
```

```
prob2 <- exp(da.reg2$scores[,2])/(exp(da.reg2$scores[,1])+exp(da.reg2$scores[,2]))
fit.roc2 <- roc(bank$Personal.Loan, prob2, plot = T, col = 'blue')
```

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

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



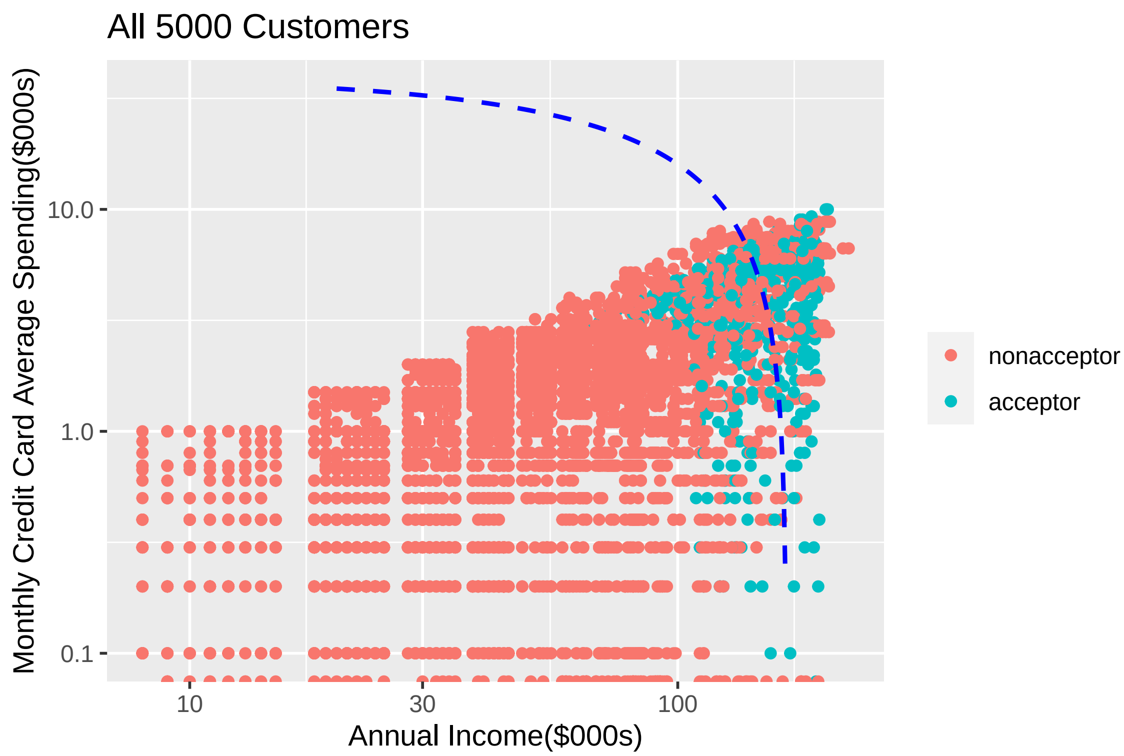

```
fit.roc2$auc
```

```
## Area under the curve: 0.925
```

Classification Boundary

$$\hat{\delta}(Nonacceptor|Income, CCAvg) = \hat{\delta}(Acceptor|Income, CCAvg)$$
$$-1.4942 + 0.03948 * Income + 0.09932 * CCAvg = -9.029 + 0.08459 * Income + 0.2887 * CCAvg$$
$$0.04511 * Income + 0.1894 * CCAvg = 7.535$$
$$CCAvg = 39.78 - 0.2382 * Income$$

```
ggplot() +
geom_point(aes(x = Income, y= CCAvg, col=Personal.Loan), bank) +
geom_line(aes(x = seq(20,166,1), y = (seq(20,166,1)*(-0.2382) + 39.78)),
          linetype= 'dashed', size = 0.8, col = 'blue') +
scale_colour_hue(name=NULL,labels=c('nonacceptor','acceptor')) +
scale_x_log10() +
scale_y_log10() +
labs(title = 'All 5000 Customers') +
xlab("Annual Income($000s)") +
ylab("Monthly Credit Card Average Spending($000s)")
```



Case Study III: IRIS

In IRIS dataset, we try to use Sepal.Length, Sepal.Width, Petal.Length and Petal.Width to predict the Species of IRIS.

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

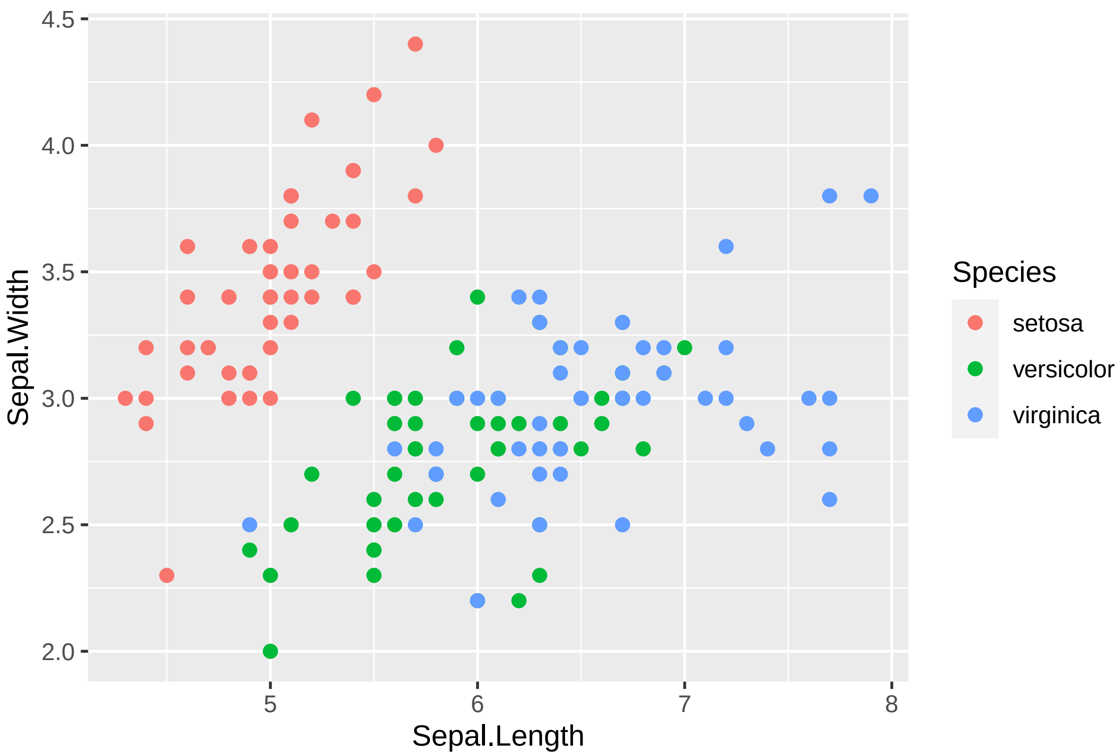
```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
```

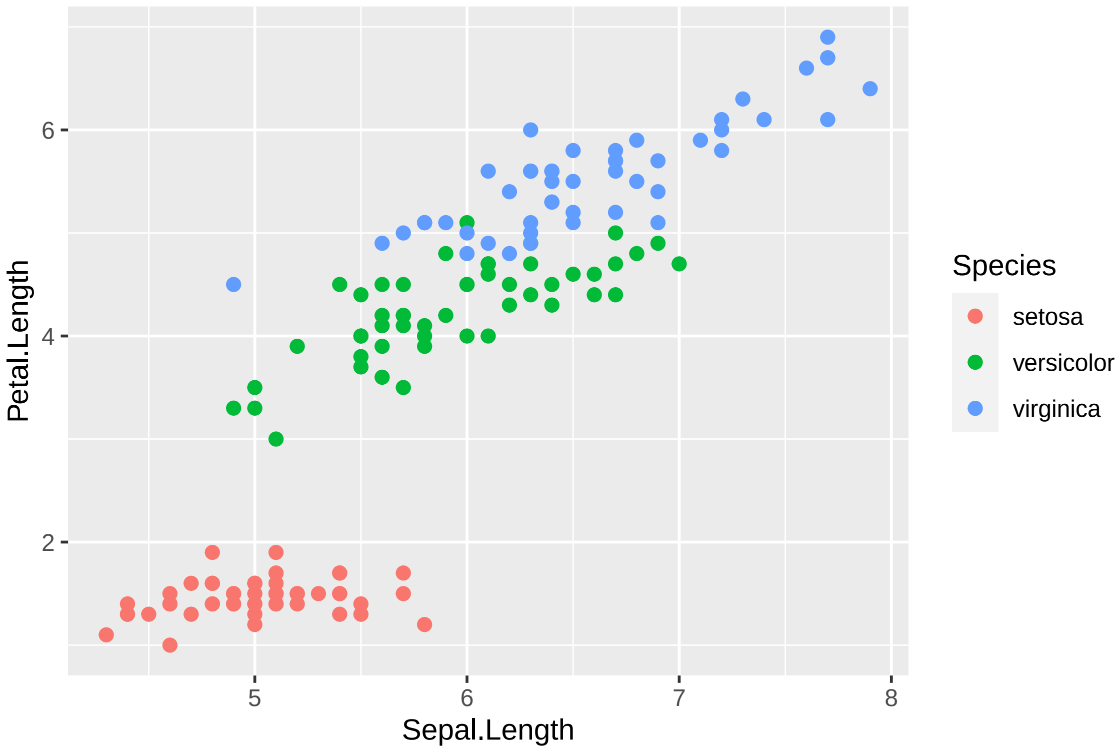
```
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

EDA

```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, col = Species), size = 2) +
  scale_colour_hue(name='Species')
```




```
iris %>%
  ggplot() +
  geom_point(aes(x = Sepal.Length, y = Petal.Length, col = Species), size = 2) +
  scale_colour_hue(name='Species')
```



LDA

```
iris.linda <- linDA(iris[,-5], iris$Species, prior = c(1/3,1/3,1/3))
iris.linda$functions
```

```
##                  setosa versicolor  virginica
## constant     -86.30847 -72.852607 -104.36832
## Sepal.Length  23.54417  15.698209   12.44585
## Sepal.Width   23.58787   7.072510    3.68528
## Petal.Length -16.43064   5.211451   12.76654
## Petal.Width  -17.39841   6.434229   21.07911
```

We could caculate the classification score for each class.

$$\hat{\delta}(setosa|Sepal, Petal) = -86.31 + 23.54 * SL + 23.59 * SW - 16.43 * PL - 17.40 * PW$$

$$\hat{\delta}(versicolor|Sepal, Petal) = -72.85 + 15.70 * SL + 7.073 * SW + 5.211 * PL + 6.434 * PW$$

$$\hat{\delta}(virginica|Sepal, Petal) = -104.4 + 12.45 * SL + 3.685 * SW + 12.77 * PL + 21.08 * PW$$

```
iris.linda$scores[c(1:10),]
```

```
##       setosa versicolor  virginica
## 1   89.84175   40.54492  -5.907026
## 2   73.33898   33.86902 -10.238836
## 3   74.99079   31.62274 -13.267604
## 4   66.99145   30.38796 -12.327408
## 5   89.84612   39.68235  -6.783083
## 6   97.93127   50.93367   7.346627
## 7   73.97104   32.63199 -10.390567
## 8   83.48548   38.78899  -6.243484
## 9   59.20811   25.31267 -16.830288
## 10  75.79455   34.45400 -10.701564
```

```r
confusionMatrix(iris.linda$classification, iris$Species)
```

Confusion Matrix

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         1
##   virginica       0          2        49
##
## Overall Statistics
##
##                Accuracy : 0.98
##                  95% CI : (0.9427, 0.9959)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.97
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9600           0.9800
## Specificity                 1.0000            0.9900           0.9800
## Pos Pred Value              1.0000            0.9796           0.9608
## Neg Pred Value              1.0000            0.9802           0.9899
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3200           0.3267
## Detection Prevalence        0.3333            0.3267           0.3400
## Balanced Accuracy           1.0000            0.9750           0.9800
```

Discriminant Variables We could also use `lda` in `mass` to get discriminant variables.

When there are 3 classes, linear discriminant analysis can be viewed in 2 dimensional plot.

```r
iris.lda <- lda(iris$Species~., data=iris)
iris.lda
```

```
## Call:
## lda(iris$Species ~ ., data = iris)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
##
```

```
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8293776  0.02410215
## Sepal.Width   1.5344731  2.16452123
## Petal.Length -2.2012117 -0.93192121
## Petal.Width  -2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

```
predict.iris_LDA <- predict(iris.lda)
table(iris$Species, predict.iris_LDA$class)
```

```
##
##               setosa versicolor virginica
##   setosa          50          0         0
##   versicolor       0         48         2
##   virginica        0          1        49
```

```
all(predict.iris_LDA$class == iris.linda$classification)
```

```
## [1] TRUE
```

```
# two LDA functions give the same prediction as expected
```

```
iris_pred <- cbind(iris,
                   data.frame(dv1 = iris.lda$scaling[1,1]*iris[,1] +
                                iris.lda$scaling[2,1]*iris[,2] +
                                iris.lda$scaling[3,1]*iris[,3] +
                                iris.lda$scaling[4,1]*iris[,4],
                              dv2 = iris.lda$scaling[1,2]*iris[,1] +
                                iris.lda$scaling[2,2]*iris[,2] +
                                iris.lda$scaling[3,2]*iris[,3] +
                                iris.lda$scaling[4,2]*iris[,4],
                              pred = predict.iris_LDA$class))
iris_pred[c(1:6),]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species      dv1      dv2
## 1          5.1         3.5          1.4         0.2  setosa 5.956693 6.961893
## 2          4.9         3.0          1.4         0.2  setosa 5.023581 5.874812
## 3          4.7         3.2          1.3         0.2  setosa 5.384722 6.396088
## 4          4.6         3.1          1.5         0.2  setosa 4.708094 5.990841
## 5          5.0         3.6          1.4         0.2  setosa 6.027203 7.175935
## 6          5.4         3.9          1.7         0.4  setosa 5.596840 8.123194
##     pred
## 1 setosa
## 2 setosa
## 3 setosa
## 4 setosa
## 5 setosa
## 6 setosa
```

```
iris_pred %>%
  ggplot() +
  geom_point(aes(x = dv1, y = dv2, col = Species), size = 2) +
  xlab("Discriminant Variable 1") +
```

```
ylab("Discriminant Variable 2") +
scale_colour_hue(name='Species')
```