Classification

J. Di Iorio, F. Chiaromonte

2/28/2021

Libraries

We are going to use **tidyverse**, **caret**, **readxl** and **MASS**

```
library(tidyverse) # for data manipulation and visualization
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidyverse 1.3
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

```
library(readxl) # for reading xlsx files
library(caret) # for statistical learning techniques
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(MASS) # for AIC based stepwise regression
```

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

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

```
library(ggplot2) # for plots
library(klaR) # for LDA and QDA partition
```

Data

Today we are going to use the **Titanic** data set. It is made up of 1309 rows and 15 attributes. Each row represents a passenger and the columns describe different attributes about each passenger.

```
df <- read_excel("Titanic.xlsx")
head(df)
```

```
## # A tibble: 6 x 15
##   pclass survived Residence name    age sibsp parch ticket  fare cabin embarked
##    <dbl>    <dbl>     <dbl> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
## 1      3        0         0 Abbi~    42     0     0 C.A. ~  7.55 <NA>  S
## 2      3        0         0 Abbo~    13     0     2 C.A. ~ 20.2  <NA>  S
## 3      3        0         0 Abbo~    16     1     1 C.A. ~ 20.2  <NA>  S
## 4      3        1         0 Abbo~    35     1     1 C.A. ~ 20.2  <NA>  S
## 5      3        1         2 Abel~    16     0     0 348125  7.65 <NA>  S
## 6      3        1         0 Abel~    25     0     0 348122  7.65 F G63 S
## # ... with 4 more variables: boat <chr>, body <dbl>, home.dest <chr>,
## #   Gender <dbl>
```

Let us see how our data are structured.

```
str(df)
```

```
## tibble [1,309 x 15] (S3: tbl_df/tbl/data.frame)
##  $ pclass   : num [1:1309] 3 3 3 3 3 3 2 2 3 3 ...
##  $ survived : num [1:1309] 0 0 0 1 1 1 0 1 1 1 ...
##  $ Residence: num [1:1309] 0 0 0 0 2 0 2 2 2 2 ...
##  $ name     : chr [1:1309] "Abbing, Mr. Anthony" "Abbott, Master. Eugene Joseph" "Abbott, Mr. Rossmo
##  $ age      : num [1:1309] 42 13 16 35 16 25 30 28 20 18 ...
##  $ sibsp    : num [1:1309] 0 0 1 1 0 0 1 1 0 0 ...
##  $ parch    : num [1:1309] 0 2 1 1 0 0 0 0 0 0 ...
##  $ ticket   : chr [1:1309] "C.A. 5547" "C.A. 2673" "C.A. 2673" "C.A. 2673" ...
##  $ fare     : num [1:1309] 7.55 20.25 20.25 20.25 7.65 ...
##  $ cabin    : chr [1:1309] NA NA NA NA ...
##  $ embarked : chr [1:1309] "S" "S" "S" "S" ...
##  $ boat     : chr [1:1309] NA NA NA "A" ...
##  $ body     : num [1:1309] NA NA 190 NA NA NA NA NA NA NA ...
##  $ home.dest: chr [1:1309] NA "East Providence, RI" "East Providence, RI" "East Providence, RI" ...
##  $ Gender   : num [1:1309] 0 0 0 1 1 0 0 1 0 1 ...
```

The type of some column is not the optimal one. We change it.

```
df$pclass <- as.factor(df$pclass)
df$survived <- as.factor(df$survived)
df$Residence <- as.factor(df$Residence)
df$body <- as.factor(df$body)
df$Gender <- as.factor(df$Gender)
```

Let us summarize our data using **summary** which returns information for every column of the data set depending on the column type.

```
summary(df)
```

```
##  pclass  survived Residence     name                age
##  1:323   0:809    0:258     Length:1309        Min.   : 0.1667
##  2:277   1:500    1:302     Class :character   1st Qu.:21.0000
##  3:709            2:749     Mode  :character   Median :28.0000
##                                                Mean   :29.8811
##                                                3rd Qu.:39.0000
##                                                Max.   :80.0000
##                                                NA's   :263
```

```
##      sibsp            parch          ticket               fare
##  Min.   :0.0000   Min.   :0.000   Length:1309        Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:0.000   Class :character   1st Qu.:  7.896
##  Median :0.0000   Median :0.000   Mode  :character   Median : 14.454
##  Mean   :0.4989   Mean   :0.385                      Mean   : 33.295
##  3rd Qu.:1.0000   3rd Qu.:0.000                      3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000                      Max.   :512.329
##                                                      NA's   :1
##     cabin             embarked            boat                body
##  Length:1309        Length:1309        Length:1309        1      :   1
##  Class :character   Class :character   Class :character   4      :   1
##  Mode  :character   Mode  :character   Mode  :character   7      :   1
##                                                           9      :   1
##                                                           14     :   1
##                                                           (Other): 116
##                                                           NA's   :1188
##   home.dest         Gender
##  Length:1309        0:843
##  Class :character   1:466
##  Mode  :character
##
##
##
##
```

We could drop some column: the ones having more than 50 percent of NAs and the ones we don't think could be useful for our analysis. We also can drop all those rows without age info.

```
# dropping columns
df <- df %>% dplyr::select(-c(name,ticket, fare, cabin, embarked, boat, body, home.dest))
# filtering out rows
df <- df %>% filter(!is.na(age))
dim(df)
```

```
## [1] 1046    7
```

Logistic Regression

The data is divided into training and testing set using a 75:25 ratio.

```
set.seed(123)
training_samples <- df$survived %>% createDataPartition(p = 0.75, list = FALSE)
train <- df[training_samples, ]
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
test <- df[-training_samples, ]
```

A Simple Logistic Regression

Using the training set, we build a simple logistic regression model using sex as the only predictor for survival status of the passenger.

```
simple_glm <- glm(survived ~ Gender, data = train, family = 'binomial')
summary(simple_glm)
```

```
##
## Call:
## glm(formula = survived ~ Gender, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6274  -0.6878  -0.6878   0.7864   1.7650
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.3211     0.1109  -11.91   <2e-16 ***
## Gender1       2.3362     0.1719   13.59   <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: 1063.10  on 785  degrees of freedom
## Residual deviance:  847.53  on 784  degrees of freedom
## AIC: 851.53
##
## Number of Fisher Scoring iterations: 4
```

We look at the coefficient in this way:

```
simple_glm$coefficients
```

```
## (Intercept)     Gender1
##   -1.321108    2.336156
```

Our model is $log(\frac{p(x)}{1-p(x)}) = \beta_0 + \beta_1 Gender$. In our case $\beta_0 = Intercept$ and $\beta_1 = Gender1$. Therefore, $log(\frac{p(x)}{1-p(x)}) = Intercept + Gender1 * Gender$.

Let us see how our model performs in terms of accuracy (proportion of correct predictions, both true positives and true negatives, among the total number of cases examined)

```
# Test for accuracy
predict_sex_survived <- predict(simple_glm, newdata = test,type = 'response')
# Since Survived can only be either 1 or 0, write if statement to round up of down the response
predict_sex_survived <- ifelse(predict_sex_survived>0.5,1,0)
accuracy <- mean(predict_sex_survived==test$survived)
accuracy
```

```
## [1] 0.8115385
```

Our result is "pretty good" and it is also consistent with the "women and children first" myth.

A Simple Logistic Regression - Backward selection

Accuracy score is not too bad as a start, however we can further improve it by including more features. Let us start with one model considering all the features.

```
glm_complete <- glm(survived ~ ., data=train, family = 'binomial')
summary(glm_complete)
```

```
##
## Call:
## glm(formula = survived ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7502  -0.6744  -0.4067   0.6634   2.5247
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.638344   0.390539   4.195 2.73e-05 ***
## pclass2     -1.177714   0.291066  -4.046 5.21e-05 ***
## pclass3     -2.372486   0.296672  -7.997 1.27e-15 ***
## Residence1  -0.483365   0.307982  -1.569  0.11654
## Residence2  -0.125636   0.271809  -0.462  0.64392
## age         -0.044028   0.007677  -5.735 9.74e-09 ***
## sibsp       -0.384458   0.118970  -3.232  0.00123 **
## parch        0.145025   0.109662   1.322  0.18601
## Gender1      2.428404   0.199612  12.166  < 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: 1063.10  on 785  degrees of freedom
## Residual deviance:  729.57  on 777  degrees of freedom
## AIC: 747.57
##
## Number of Fisher Scoring iterations: 5
```

Let us select the significant predictors using stepwise regression with AIC as the score. A new model is selected eventually as it produces a lower AIC score.

```
glm_stepwise <- glm_complete %>% stepAIC(direction='both', trace = FALSE)
summary(glm_stepwise)
```

```
##
## Call:
## glm(formula = survived ~ pclass + age + sibsp + Gender, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7215  -0.6664  -0.4103   0.6854   2.5462
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.619044   0.364356   4.444 8.85e-06 ***
```

```
## pclass2     -1.383870   0.263438  -5.253 1.50e-07 ***
## pclass3     -2.469957   0.264871  -9.325  < 2e-16 ***
## age         -0.045191   0.007606  -5.942 2.82e-09 ***
## sibsp       -0.346727   0.111916  -3.098  0.00195 **
## Gender1      2.496307   0.196409  12.710  < 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: 1063.10  on 785  degrees of freedom
## Residual deviance:  734.08  on 780  degrees of freedom
## AIC: 746.08
##
## Number of Fisher Scoring iterations: 4
```

Let us compare the two models (complete and AIC selected) using AIC

```
AIC(glm_complete, glm_stepwise)
```

```
##              df      AIC
## glm_complete  9 747.5722
## glm_stepwise  6 746.0796
```

Using the second model, we can then compute the probability for survival and accuracy of the model.

```
# Test for accuracy
predict_sex_survived <- predict(glm_stepwise, newdata = test,type = 'response')
# Since Survived can only be either 1 or 0, write if statement to round up of down the response
predict_sex_survived <- ifelse(predict_sex_survived>0.5,1,0)
accuracy <- mean(predict_sex_survived==test$survived)
accuracy
```

```
## [1] 0.8038462
```

We can create also a confusion matrix (and statistics) to compare our prediction to the labels in the test set.

```
confusionMatrix(as.factor(predict_sex_survived), test$survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 133  30
##          1  21  76
##
##                Accuracy : 0.8038
##                  95% CI : (0.7503, 0.8503)
##     No Information Rate : 0.5923
##     P-Value [Acc > NIR] : 2.921e-13
##
##                   Kappa : 0.5884
##
##  Mcnemar's Test P-Value : 0.2626
##
##             Sensitivity : 0.8636
##             Specificity : 0.7170
##          Pos Pred Value : 0.8160
```

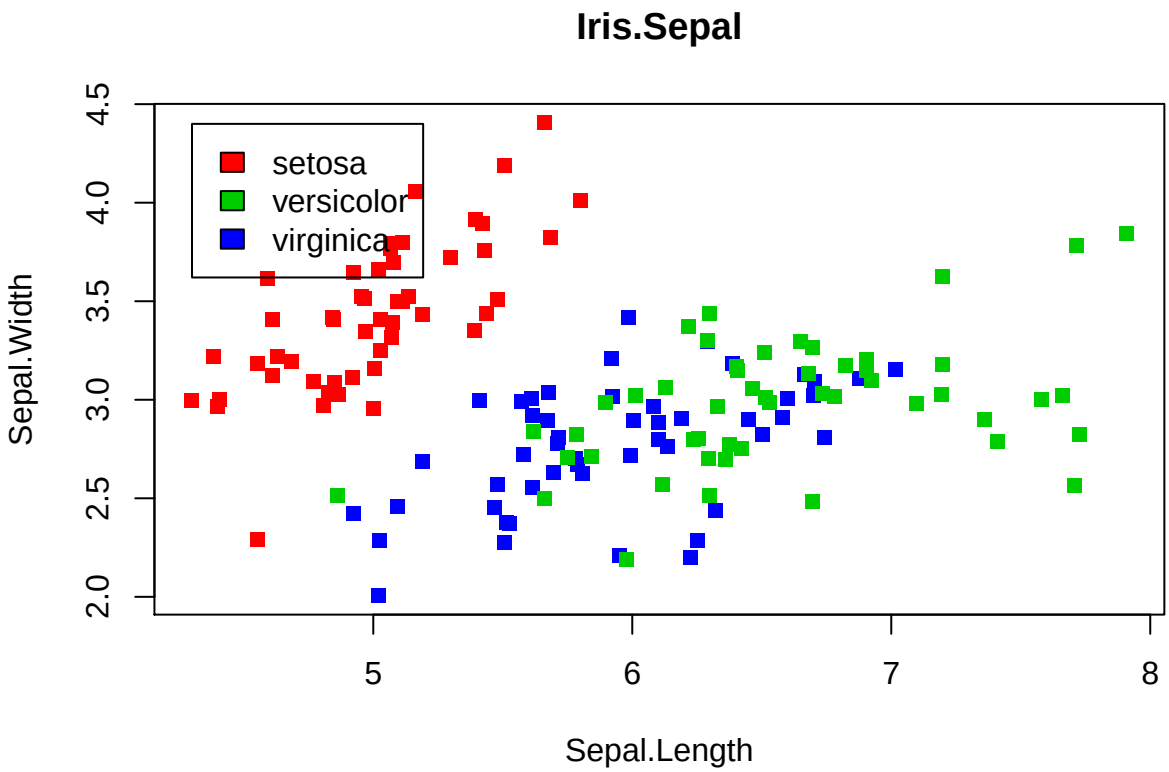
```
##          Neg Pred Value : 0.7835
##              Prevalence : 0.5923
##          Detection Rate : 0.5115
##    Detection Prevalence : 0.6269
##       Balanced Accuracy : 0.7903
##
##        'Positive' Class : 0
##
```

LDA and QDA

Let us apply LDA and QDA to a multi label dataset such as **iris**. We are going to use just the first two columns with a gaussian noise.

```
iris2 <- iris[,c(1,2,5)]
species_name <- iris$Species
iris2[,1] <- iris2[,1] + rnorm(150, sd=0.025)
iris2[,2] <- iris2[,2] + rnorm(150, sd=0.025)

plot(iris2[,1:2], main='Iris.Sepal', xlab='Sepal.Length', ylab='Sepal.Width', pch=15)
points(iris2[1:50,], col=2, pch=15)
points(iris2[51:100,], col=4, pch=15)
points(iris2[101:150,], col=3, pch=15)
legend(min(iris[,1]), max(iris[,2]), legend=levels(species_name), fill=c(2,3,4))
```



Once again we create a train set and a test set.

```
set.seed(123)
training.samples <- species_name %>%
  createDataPartition(p = 0.8, list = FALSE)
```

```
train <- iris2[training.samples, ]
test <- iris2[-training.samples, ]
```

It is generally recommended to standardize/normalize continuous predictor before the analysis.

```
# Estimate preprocessing parameters
preproc.param <- train %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train_transformed <- preproc.param %>% predict(train)
test_transformed <- preproc.param %>% predict(test)
```

LDA

Before performing LDA, consider:

- Inspecting the univariate distributions of each variable and make sure that they are normally distribute. If not, you can transform them using log and root for exponential distributions and Box-Cox for skewed distributions.
- Removing outliers from your data and standardize the variables to make their scale comparable.

```
lda.iris <- lda(factor(Species)~ Sepal.Length + Sepal.Width, data=train_transformed)
lda.iris
```

```
## Call:
## lda(factor(Species) ~ Sepal.Length + Sepal.Width, data = train_transformed)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width
## setosa       -1.0120940  0.77319367
## versicolor    0.1004989 -0.68484625
## virginica     0.9115952 -0.08834742
##
## Coefficients of linear discriminants:
##                     LD1        LD2
## Sepal.Length -1.775634 -0.6120093
## Sepal.Width   1.086270 -0.9378600
##
## Proportion of trace:
##    LD1    LD2
## 0.9493 0.0507
```

The linear discriminant function from the result in above can be identified using the Coefficients of Linear discriminants. The "proportion of trace"" that is printed is the percentage separation achieved by each discriminant function.

We will find the model accuracy for training data.

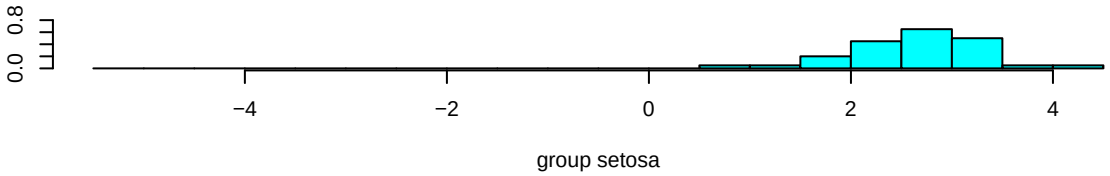
```
predmodel.train.lda = predict(lda.iris, data=train_transformed)
confusionMatrix(as.factor(predmodel.train.lda$class), train_transformed$Species)
```

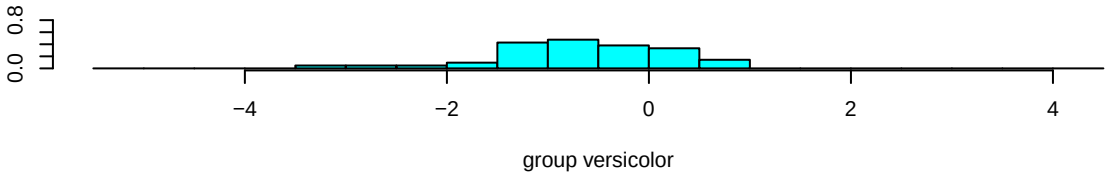
```
## Confusion Matrix and Statistics
```

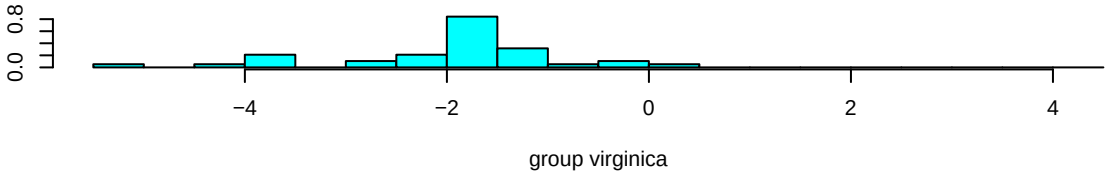
```
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         39          1         0
##   versicolor      1         29        12
##   virginica       0         10        28
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.7172, 0.8675)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 0.9750            0.7250           0.7000
## Specificity                 0.9875            0.8375           0.8750
## Pos Pred Value              0.9750            0.6905           0.7368
## Neg Pred Value              0.9875            0.8590           0.8537
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3250            0.2417           0.2333
## Detection Prevalence        0.3333            0.3500           0.3167
## Balanced Accuracy           0.9812            0.7812           0.7875
```

The below plot shows how the response class has been classified by the LDA classifier. The X-axis shows the value of line defined by the coefficient of linear discriminant for LDA model. The two groups are the groups for response classes.

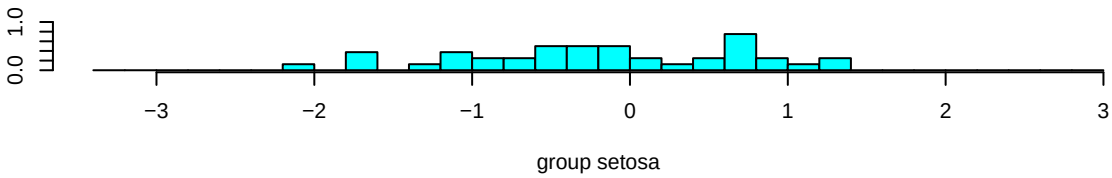
```
# first discriminant
ldahist(predmodel.train.lda$x[,1], g= predmodel.train.lda$class)
```

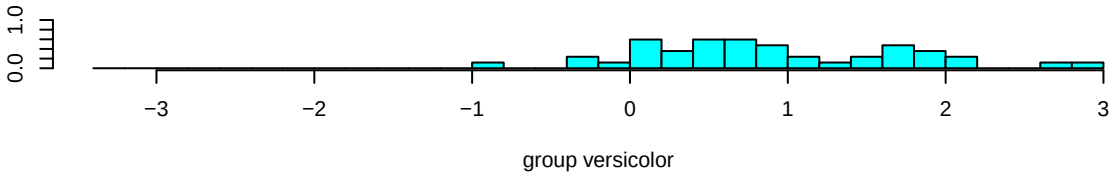


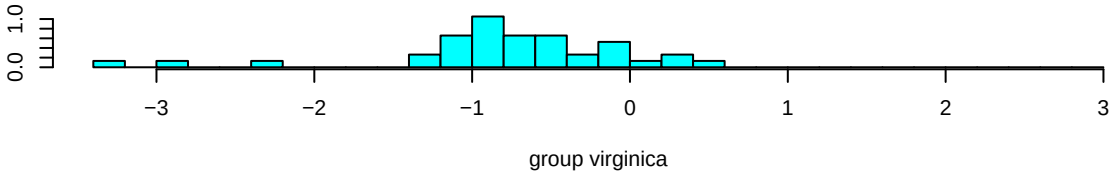





```
# second discriminant
ldahist(predmodel.train.lda$x[,2], g= predmodel.train.lda$class)
```

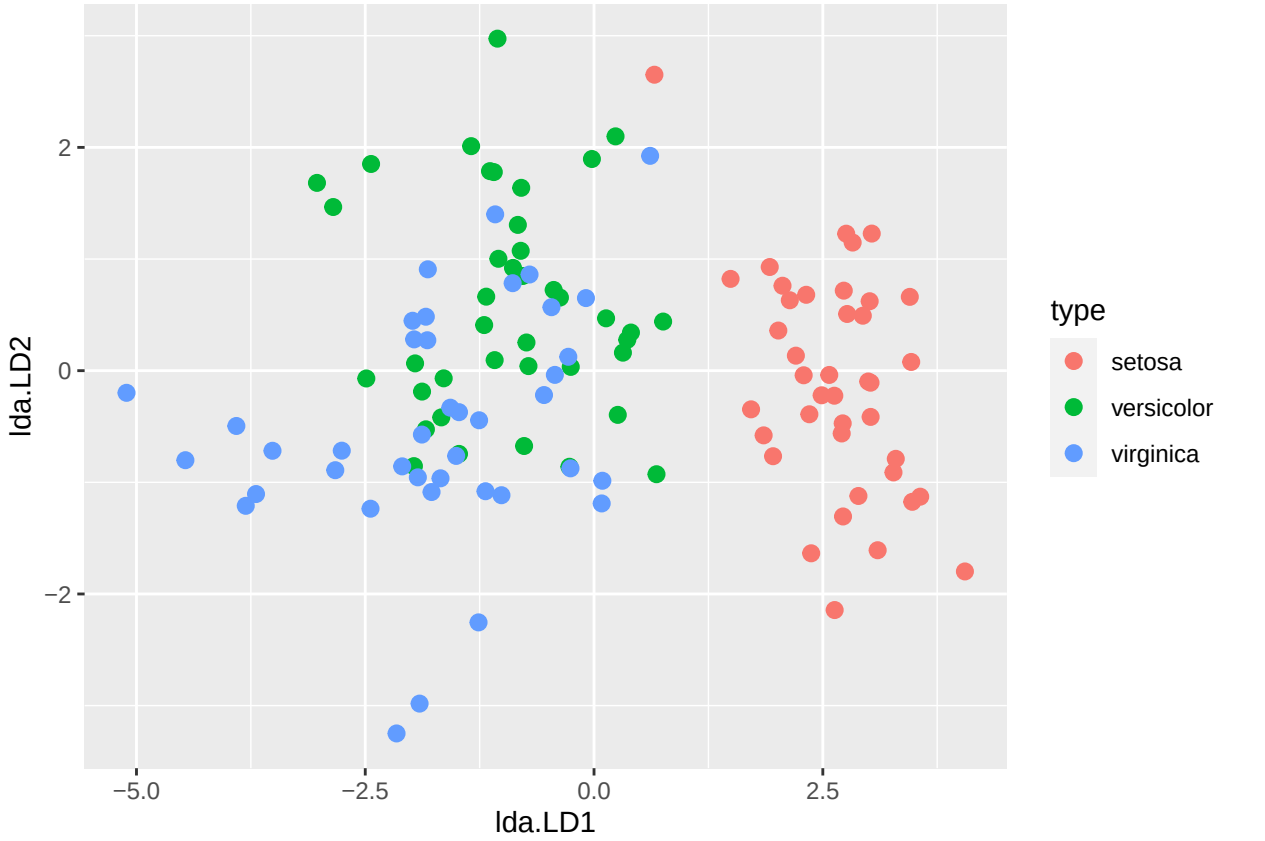






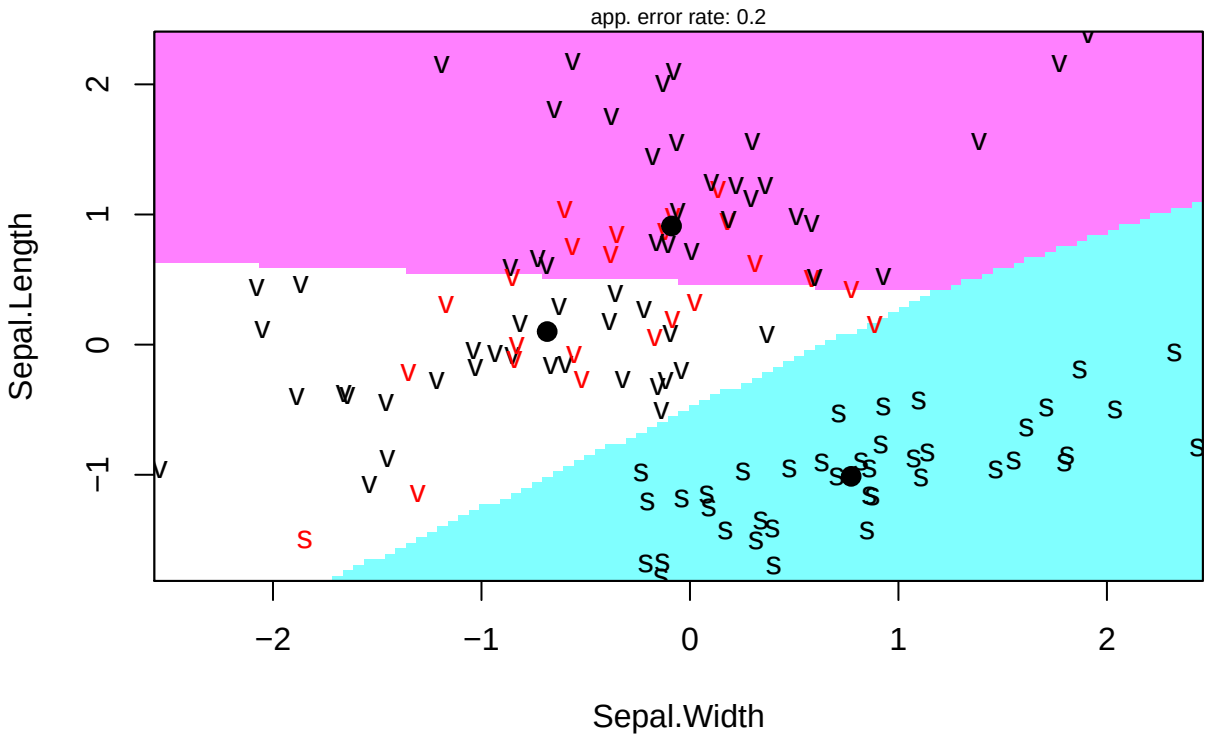
See new x with original labels

```r
#convert to data frame
newdata <- data.frame(type = train_transformed$Species, lda = predmodel.train.lda$x)
library(ggplot2)
ggplot(newdata) + geom_point(aes(lda.LD1, lda.LD2, colour = type), size = 2.5)
```



See geometric division

```r
library(klaR)
partimat(factor(Species)~ Sepal.Length + Sepal.Width, data=train_transformed, method = "lda")
```



Now we will check for model accuracy for test data.

```
predmodel.test.lda = predict(lda.iris, newdata=test_transformed)
confusionMatrix(as.factor(predmodel.test.lda$class), test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          8         4
##   virginica       0          2         6
##
## Overall Statistics
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 2.09e-07
##
##                   Kappa : 0.7
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.8000           0.6000
```

```
## Specificity                 1.0000            0.8000           0.9000
## Pos Pred Value              1.0000            0.6667           0.7500
## Neg Pred Value              1.0000            0.8889           0.8182
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2667           0.2000
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.8000           0.7500
```

QDA

Next we will fit the model to QDA as below. The command is similar to LDA and it outputs the prior probabilities and Group means. Please note that 'prior probability' and 'Group Means' values are same as of LDA.

```
qda.iris <- qda(factor(Species)~ Sepal.Length + Sepal.Width, data=train_transformed)
qda.iris
```

```
## Call:
## qda(factor(Species) ~ Sepal.Length + Sepal.Width, data = train_transformed)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width
## setosa       -1.0120940  0.77319367
## versicolor    0.1004989 -0.68484625
## virginica     0.9115952 -0.08834742
```

We will find the model accuracy for training data.

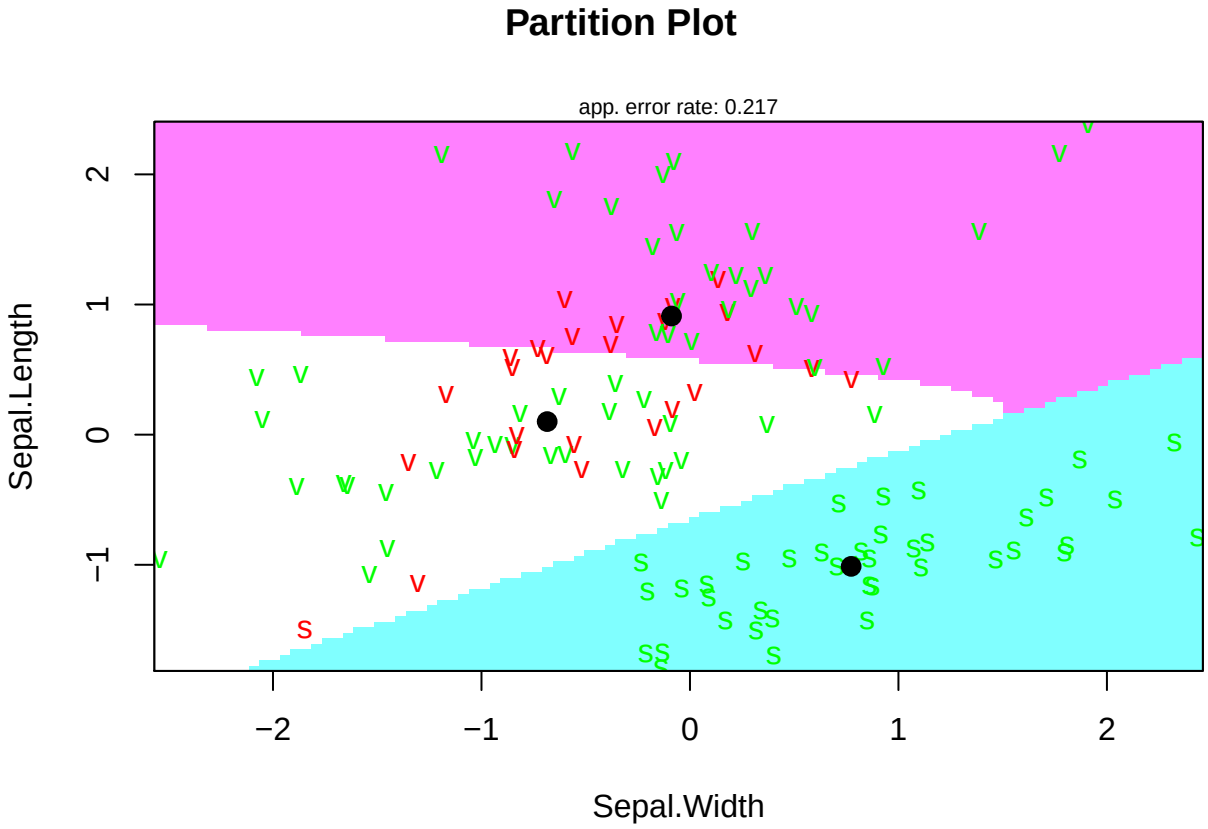
```
predmodel.train.qda = predict(qda.iris, data=train_transformed)
confusionMatrix(as.factor(predmodel.train.qda$class), train_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         39          0         0
##   versicolor      1         30        15
##   virginica       0         10        25
##
## Overall Statistics
##
##                Accuracy : 0.7833
##                  95% CI : (0.6989, 0.8533)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.675
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 0.9750            0.7500           0.6250
## Specificity                 1.0000            0.8000           0.8750
## Pos Pred Value              1.0000            0.6522           0.7143
## Neg Pred Value              0.9877            0.8649           0.8235
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3250            0.2500           0.2083
## Detection Prevalence        0.3250            0.3833           0.2917
## Balanced Accuracy           0.9875            0.7750           0.7500
```

We can see the geometric partition

```
library(klaR)
partimat(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed, method = "qda", col.corre
```



kNN

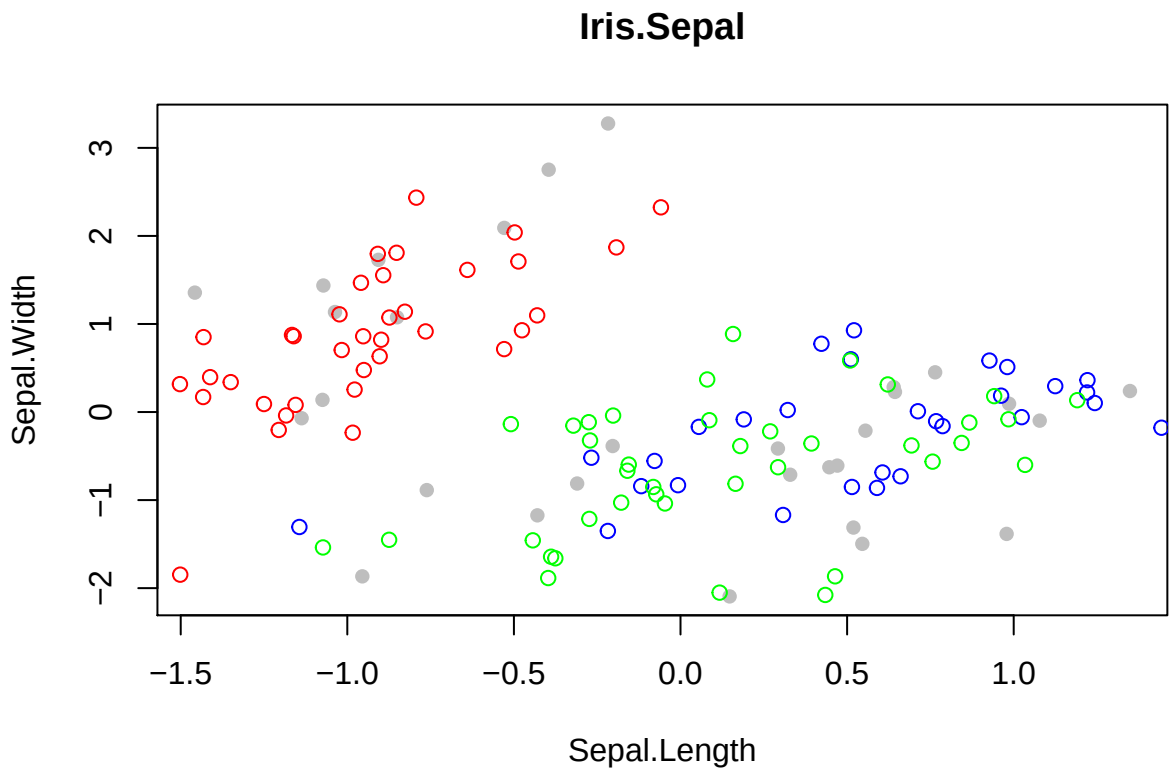
We are going to use knn3Train() function on iris2 dataset. Let us train the knn with $k = 1, 2, 3, 10$

```
knn_iris1 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed, k = 1)
knn_iris2 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed, k = 2)
knn_iris3 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed, k = 3)

knn_iris10 <- knn3(factor(Species) ~ Sepal.Length + Sepal.Width, data=train_transformed, k = 10)
```

Hand-made KNN:

```
plot(test_transformed[,1:2], main='Iris.Sepal', xlab='Sepal.Length', ylab='Sepal.Width',pch=16, col='gr
points(train_transformed[which(train_transformed$Species=="setosa"),1:2], col='red', pch=1)
points(train_transformed[which(train_transformed$Species=="virginica"),1:2], col='blue', pch=1)
points(train_transformed[which(train_transformed$Species=="versicolor"),1:2], col='green', pch=1)
```



And now let us predict new points labels in the test set. Using $k = 1$

```
predict(knn_iris1, test_transformed, type='prob')
```

```
##       setosa versicolor virginica
##  [1,]      1          0         0
##  [2,]      1          0         0
##  [3,]      1          0         0
##  [4,]      1          0         0
##  [5,]      1          0         0
##  [6,]      1          0         0
##  [7,]      1          0         0
##  [8,]      1          0         0
##  [9,]      1          0         0
## [10,]      1          0         0
## [11,]      0          0         1
## [12,]      0          0         1
## [13,]      0          1         0
## [14,]      0          1         0
## [15,]      0          1         0
## [16,]      0          0         1
## [17,]      0          1         0
## [18,]      0          1         0
## [19,]      0          0         1
## [20,]      0          1         0
```

```
## [21,]      0          1         0
## [22,]      0          0         1
## [23,]      0          1         0
## [24,]      0          0         1
## [25,]      0          1         0
## [26,]      0          1         0
## [27,]      0          1         0
## [28,]      0          0         1
## [29,]      0          1         0
## [30,]      0          0         1
```

```
predict_test_knn1 <- predict(knn_iris1, test_transformed, type='class')
confusionMatrix(predict_test_knn1, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          6         6
##   virginica       0          4         4
##
## Overall Statistics
##
##                Accuracy : 0.6667
##                  95% CI : (0.4719, 0.8271)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.0001938
##
##                   Kappa : 0.5
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.6000           0.4000
## Specificity                 1.0000            0.7000           0.8000
## Pos Pred Value              1.0000            0.5000           0.5000
## Neg Pred Value              1.0000            0.7778           0.7273
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2000           0.1333
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.6500           0.6000
```

```
predict(knn_iris2, test_transformed, type='prob')
```

```
##       setosa versicolor virginica
##  [1,]      1        0.0       0.0
##  [2,]      1        0.0       0.0
##  [3,]      1        0.0       0.0
##  [4,]      1        0.0       0.0
##  [5,]      1        0.0       0.0
##  [6,]      1        0.0       0.0
##  [7,]      1        0.0       0.0
```

```
##  [8,]      1        0.0       0.0
##  [9,]      1        0.0       0.0
## [10,]      1        0.0       0.0
## [11,]      0        0.0       1.0
## [12,]      0        0.5       0.5
## [13,]      0        1.0       0.0
## [14,]      0        0.5       0.5
## [15,]      0        1.0       0.0
## [16,]      0        0.5       0.5
## [17,]      0        0.5       0.5
## [18,]      0        1.0       0.0
## [19,]      0        0.5       0.5
## [20,]      0        0.5       0.5
## [21,]      0        1.0       0.0
## [22,]      0        0.0       1.0
## [23,]      0        0.5       0.5
## [24,]      0        0.5       0.5
## [25,]      0        0.5       0.5
## [26,]      0        1.0       0.0
## [27,]      0        0.5       0.5
## [28,]      0        0.5       0.5
## [29,]      0        0.5       0.5
## [30,]      0        0.0       1.0
```

```
predict_test_knn2 <- predict(knn_iris2, test_transformed, type='class')
confusionMatrix(predict_test_knn2, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          7         5
##   virginica       0          3         5
##
## Overall Statistics
##
##                Accuracy : 0.7333
##                  95% CI : (0.5411, 0.8772)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.752e-06
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.7000           0.5000
## Specificity                 1.0000            0.7500           0.8500
## Pos Pred Value              1.0000            0.5833           0.6250
## Neg Pred Value              1.0000            0.8333           0.7727
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2333           0.1667
```

```
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.7250           0.6750
```

```
# k=3
predict(knn_iris3, test_transformed, type='prob')
```

```
##          setosa versicolor virginica
##  [1,] 1.0000000  0.0000000 0.0000000
##  [2,] 1.0000000  0.0000000 0.0000000
##  [3,] 1.0000000  0.0000000 0.0000000
##  [4,] 1.0000000  0.0000000 0.0000000
##  [5,] 1.0000000  0.0000000 0.0000000
##  [6,] 1.0000000  0.0000000 0.0000000
##  [7,] 1.0000000  0.0000000 0.0000000
##  [8,] 1.0000000  0.0000000 0.0000000
##  [9,] 1.0000000  0.0000000 0.0000000
## [10,] 1.0000000  0.0000000 0.0000000
## [11,] 0.0000000  0.0000000 1.0000000
## [12,] 0.0000000  0.6666667 0.3333333
## [13,] 0.0000000  1.0000000 0.0000000
## [14,] 0.0000000  0.6666667 0.3333333
## [15,] 0.0000000  0.6666667 0.3333333
## [16,] 0.0000000  0.3333333 0.6666667
## [17,] 0.0000000  0.6666667 0.3333333
## [18,] 0.3333333  0.6666667 0.0000000
## [19,] 0.0000000  0.6666667 0.3333333
## [20,] 0.0000000  0.3333333 0.6666667
## [21,] 0.0000000  0.6666667 0.3333333
## [22,] 0.0000000  0.3333333 0.6666667
## [23,] 0.0000000  0.3333333 0.6666667
## [24,] 0.0000000  0.6666667 0.3333333
## [25,] 0.0000000  0.6666667 0.3333333
## [26,] 0.0000000  1.0000000 0.0000000
## [27,] 0.0000000  0.3333333 0.6666667
## [28,] 0.0000000  0.3333333 0.6666667
## [29,] 0.0000000  0.6666667 0.3333333
## [30,] 0.0000000  0.0000000 1.0000000
```

```
predict_test_knn3 <- predict(knn_iris3, test_transformed, type='class')
confusionMatrix(predict_test_knn3, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          7         5
##   virginica       0          3         5
##
## Overall Statistics
##
##                Accuracy : 0.7333
##                  95% CI : (0.5411, 0.8772)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.752e-06
```

```
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.7000           0.5000
## Specificity                 1.0000            0.7500           0.8500
## Pos Pred Value              1.0000            0.5833           0.6250
## Neg Pred Value              1.0000            0.8333           0.7727
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2333           0.1667
## Detection Prevalence        0.3333            0.4000           0.2667
## Balanced Accuracy           1.0000            0.7250           0.6750
```

```
# k=10
predict(knn_iris10, test_transformed, type='prob')
```

```
##       setosa versicolor virginica
##  [1,]    1.0        0.0       0.0
##  [2,]    1.0        0.0       0.0
##  [3,]    1.0        0.0       0.0
##  [4,]    1.0        0.0       0.0
##  [5,]    1.0        0.0       0.0
##  [6,]    1.0        0.0       0.0
##  [7,]    1.0        0.0       0.0
##  [8,]    1.0        0.0       0.0
##  [9,]    1.0        0.0       0.0
## [10,]    1.0        0.0       0.0
## [11,]    0.0        0.2       0.8
## [12,]    0.0        0.7       0.3
## [13,]    0.0        0.6       0.4
## [14,]    0.0        0.5       0.5
## [15,]    0.0        0.5       0.5
## [16,]    0.0        0.4       0.6
## [17,]    0.0        0.8       0.2
## [18,]    0.1        0.7       0.2
## [19,]    0.0        0.6       0.4
## [20,]    0.0        0.7       0.3
## [21,]    0.0        0.5       0.5
## [22,]    0.0        0.5       0.5
## [23,]    0.0        0.3       0.7
## [24,]    0.0        0.5       0.5
## [25,]    0.0        0.3       0.7
## [26,]    0.0        0.8       0.2
## [27,]    0.0        0.6       0.4
## [28,]    0.0        0.6       0.4
## [29,]    0.0        0.4       0.6
## [30,]    0.0        0.4       0.6
```

```
predict_test_knn10 <- predict(knn_iris10, test_transformed, type='class')
confusionMatrix(predict_test_knn10, test_transformed$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0          8         5
##   virginica       0          2         5
##
## Overall Statistics
##
##                Accuracy : 0.7667
##                  95% CI : (0.5772, 0.9007)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 1.475e-06
##
##                   Kappa : 0.65
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.8000           0.5000
## Specificity                 1.0000            0.7500           0.9000
## Pos Pred Value              1.0000            0.6154           0.7143
## Neg Pred Value              1.0000            0.8824           0.7826
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.2667           0.1667
## Detection Prevalence        0.3333            0.4333           0.2333
## Balanced Accuracy           1.0000            0.7750           0.7000
```