STA 141 HW 5 Q2

Jiewei Chen (999 494 235)

12/2/2016

2

(i)

Extract the data corresponding to flower types setosa and versicolor, numbering a total of 100 flowers. Set aside the last 10 measurements for each flower type as test data and use the remaining data consisting of 80 measurements as training data.

```r
# load and check IRIS dataset
head(iris)
tail(iris)
dim(iris)
names(iris)
levels(iris$Species)
table(iris$Species)

# Extract the data corresponding to flower types setosa and versicolor
mydata = iris[iris$Species %in% c("setosa", "versicolor"),]
dim(mydata) # 100 5

# Set aside the last 10 measurements for each flower type as test data
mydata.v = mydata[c(41:50,91:100),]
# remove "virginica"
levels(mydata.v$Species) = factor(c("setosa", "versicolor", "versicolor"))

table(mydata.v$Species)

# The remaining data consisting of 80 measurements as training data
mydata.t = mydata[c(1:40,51:90),]
# remove "virginica"
levels(mydata.t$Species) = factor(c("setosa", "versicolor", "versicolor"))

table(mydata.t$Species)
```

(ii) Logistic Regression Model

```r
# Fit logistic model on training data. Use family = binomial for logistic.
log_model = glm(Species ~ Sepal.Length, mydata.t,
                family = binomial)

# Predict for test data. Use type = "response" to get class probabilities.
log_pred = predict(log_model, mydata.v, type = "response")
# Convert predictions to 1 or 2, for category 1 or 2 respectively.
log_pred = (log_pred > 0.5) + 1
# Convert predictions to spam or ham, same category order as original data.
log_pred = levels(mydata.t$Species)[log_pred]

# Make a confusion matrix by tabulating true classes against predicted classes.
log_con = table(true = mydata.v$Species, model = log_pred)

log_model
```

```
##
## Call:  glm(formula = Species ~ Sepal.Length, family = binomial, data = mydata.t)
##
## Coefficients:
##  (Intercept)  Sepal.Length
##      -27.980         5.119
##
## Degrees of Freedom: 79 Total (i.e. Null);  78 Residual
## Null Deviance:       110.9
## Residual Deviance: 49.93     AIC: 53.93
```

The estimates of the model parameters are -27.980 and 5.119. The regression function is

$$log\left(\frac{P(X)}{1-P(X)}\right) = -27.980 + 5.119 * Sepal.Length$$

The confusion matrix obtained by Logistic Regression is

```
##             model
## true          setosa versicolor
##   setosa          10          0
##   versicolor       2          8
```

(iii) LDA Model

```r
library(MASS)

# Fit logistic model on training data. Use family = binomial for logistic.
lda_model = lda(Species ~ Sepal.Length, mydata.t)

# Predict for test data. Use type = "response" to get class probabilities.
lda_pred = predict(lda_model, mydata.v, type = "response")

# Make a confusion matrix by tabulating true classes against predicted classes.
lda_con = table(true = mydata.v$Species, model = lda_pred$class)
```

The confusion matrix obtained by predict function from LDA method is

```
##               model
## true           setosa versicolor
##   setosa           10          0
##   versicolor        3          7
```

The group means of Sepal.Length are 5.0375, and 6.0100, respectively for "setosa" and "versicolor". So the decision boundary for linear discriminant analysis is that 5.52375. When $x > 5.52375$, it belongs to "versicolor". When $x < 5.52375$, it belongs to "setosa".

The confusion matrix obtained by this boundary from LDA method is

```r
mydata.v$Predicted.Species[mydata.v$Sepal.Length < (5.0375 + 6.0100)/2] = "setosa"
mydata.v$Predicted.Species[mydata.v$Sepal.Length > (5.0375 + 6.0100)/2] = "versicolor"
lda_con_2 = table(true = mydata.v$Species, model = mydata.v$Predicted.Species)
lda_con_2
```

```
##               model
## true           setosa versicolor
##   setosa           10          0
##   versicolor        3          7
```

(iv) kNN Model

CASE 1 - k = 3

```
# Use knn() from package class for k-nearest neighbors.
library(class)

# Fit knn (k = 3) model.
knn_pred = knn(
        # Note the use of [ ] rather than $ or [[ ]].
        #
        # The knn() function expects a matrix or data frame for the train and test
        # arguments. Using $ or [[ ]] would get a vector rather than a data frame.
        #
        train = mydata.t["Sepal.Length"], # 1-col data frame
        test  = mydata.v["Sepal.Length"],  # 1-col data frame
        cl    = mydata.t$Species,     # vector
        k     = 3
)

# Confusion matrix.
knn_con = table(true = mydata.v$Species, model = knn_pred)
```

The confusion matrix obtained from kNN-method when k = 3 is

```
##             model
## true          setosa versicolor
##   setosa         10          0
##   versicolor      3          7
```

CASE 2 - k = 5

```
# Use knn() from package class for k-nearest neighbors.
library(class)

# Fit knn (k = 5) model.
knn_pred_2 = knn(
        # Note the use of [ ] rather than $ or [[ ]].
        #
        # The knn() function expects a matrix or data frame for the train and test
        # arguments. Using $ or [[ ]] would get a vector rather than a data frame.
        #
        train = mydata.t["Sepal.Length"], # 1-col data frame
        test  = mydata.v["Sepal.Length"],  # 1-col data frame
        cl    = mydata.t$Species,     # vector
        k     = 5
)

# Confusion matrix.
knn_con_2 = table(true = mydata.v$Species, model = knn_pred_2)
```

The confusion matrix obtained from kNN-method when k = 5 is

```
##             model
## true         setosa versicolor
##   setosa         10          0
##   versicolor      2          8
```

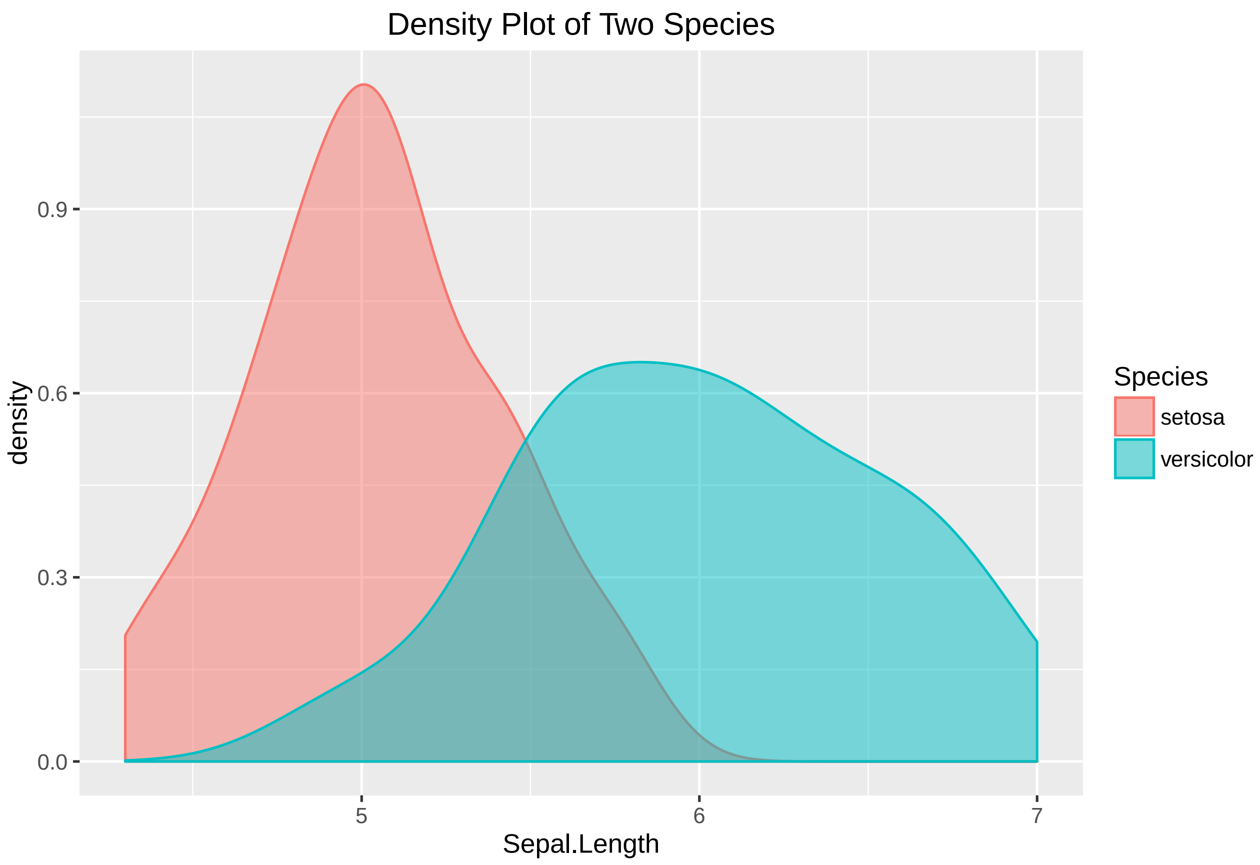
(v) A Brief Summary

It can be found that

1) For the category **setosa**, all three methods can successfully predict the classification of the testing dataset.
2) For the category **versicolor**, all three methods cannot fully predict the classification of the testing dataset. Comparing all three methods, the **Logistic Regression Model** and **kNN classification method with k = 5** will give us better result with eight out of ten cases are correctly predicted. However, the **Linear Discriminate Analysis** and **kNN classification method with k = 3** can only get seven out of ten right.

Below is the distribution of **Sepal.Length** of two different **Species**.

```r
ggplot(data = mydata.t, aes(Sepal.Length, color = Species, fill = Species,
                             group = Species)) +
       geom_density(alpha = 0.5) + labs(title = "Density Plot of Two Species")
```



It can be found that the distribution of category **setosa** is pretty normally distributed. However the distribution of category **versicolor** does not follow the normal distribution, and the distribution is right skewed.

We have learned that the **LDA** method is derived based on the assumption that the distribution of each class is (or almost) normal distribution. Therefore, the **LDA** method can successfully predict the classification of

setosa since it follows the normal distribution. However, in the case of predicting **versicolor**, due to the violation of normality assumption, the misclassification rate of **versicolor** is higher.

The **Logistic Regression Method**, instead of making assumption on the distribution, is built on predicting *Logit Probability* by using a linear model as the equation below.

$$log\left(\frac{P_i}{1-P_i}\right) = \beta_0 + \beta_1 * X_i$$

So this **Logistic Regression Method** is more flexible and will give better prediction when the normality assumption is violated. So it gave us better prediction in the classification **versicolor**.

The quality of **k-NN method** in predicting the classification is dependent on the number of nearest neighbors being chosen. In this example here, when k is small (eg. k = 3), the misclassification rate will be higher, because there is not enough data to use. In the case where k is larger (eg. k = 5), the prediction of classification will be better. However, k can not be too large, because when k is too large, you basically just consider the whole dataset. So it won't be useful for the prediction.

Other possible reason for misclassification may be that the size of the testing dataset is too small.