Classification Example with 3 Groups: Iris Dataset

28 March 2019

The following example is mainly taken from https://daviddalpiaz.github.io/r4sl/generative-models.html

Example of LDA With 3 Classes

There is "famous" dataset in R called the "Iris".

There are three iris flowers from three different species.

1. Iris-setosa
2. Iris-versicolor
3. Iris-virginica

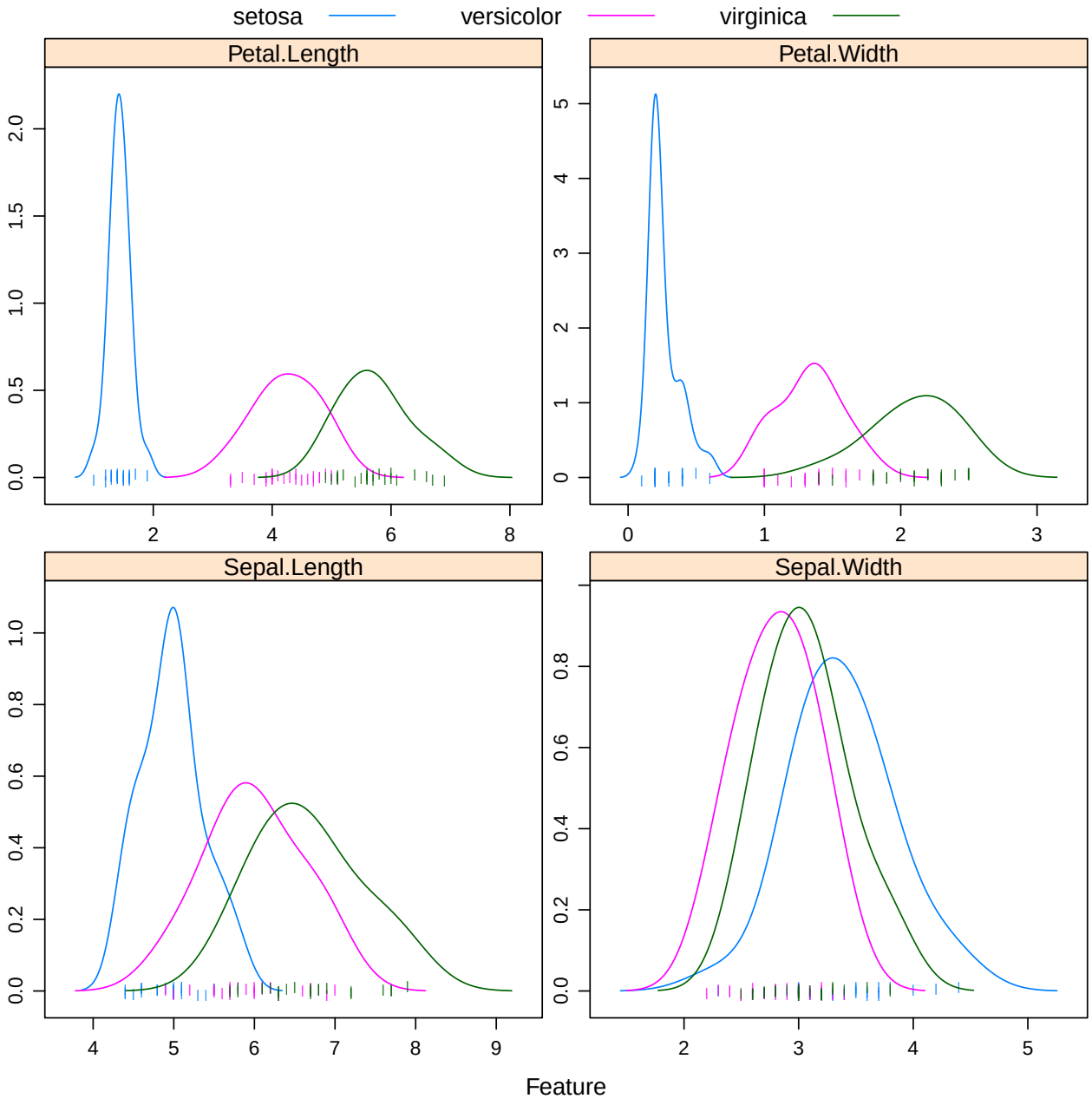
There are four features for predicting the species of flower.

1. Sepal length (cm)
2. Sepal width (cm)
3. Petal length (cm)
4. Petal width (cm)

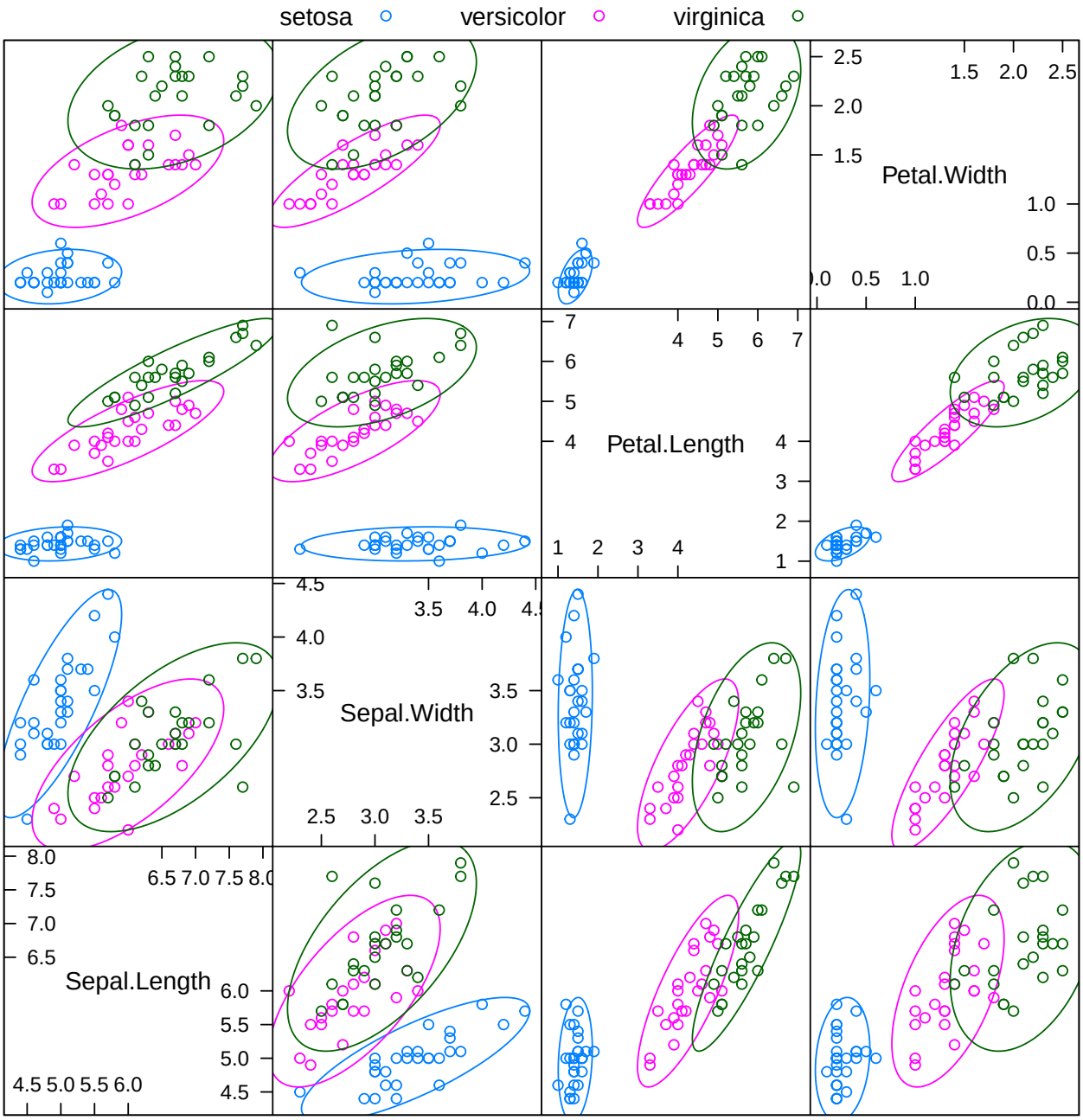
Investigating the Data

```
set.seed(430)
iris_obs = nrow(iris)
iris_idx = sample(iris_obs, size = trunc(0.50 * iris_obs))
# iris_index = sample(iris_obs, size = trunc(0.10 * iris_obs))
iris_trn = iris[iris_idx, ]
iris_tst = iris[-iris_idx, ]
```

```
caret::featurePlot(x = iris_trn[, c("Sepal.Length", "Sepal.Width",
                                    "Petal.Length", "Petal.Width")],
                   y = iris_trn$Species,
                   plot = "density",
                   scales = list(x = list(relation = "free"),
                                 y = list(relation = "free")),
                   adjust = 1.5,
                   pch = "|",
                   layout = c(2, 2),
                   auto.key = list(columns = 3))
```

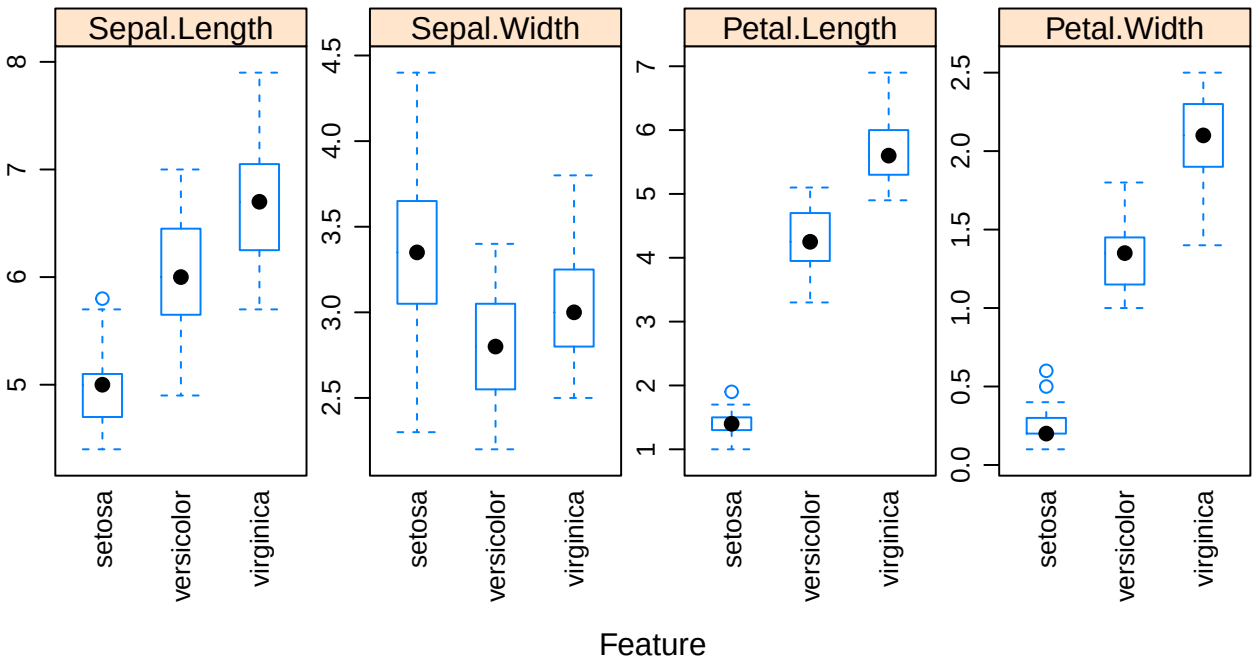



```
caret::featurePlot(x = iris_trn[, 1:4],
            y = iris_trn$Species,
            plot = "ellipse",
            ## Add a key at the top
            auto.key = list(columns = 3))
```



Scatter Plot Matrix

```
caret::featurePlot(x = iris_trn[, c("Sepal.Length", "Sepal.Width",
                                    "Petal.Length", "Petal.Width")],
                   y = iris_trn$Species,
                   plot = "box",
                   scales = list(y = list(relation = "free"),
                                 x = list(rot = 90)),
                   layout = c(4, 1))
```



What is a good classifier?

Using LDA

```
library(MASS)
iris_lda = lda(Species ~ ., data = iris_trn)
iris_lda
```

```
## Call:
## lda(Species ~ ., data = iris_trn)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3733333  0.3200000  0.3066667
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.978571    3.378571     1.432143   0.2607143
## versicolor     5.995833    2.808333     4.254167   1.3333333
## virginica      6.669565    3.065217     5.717391   2.0956522
##
## Coefficients of linear discriminants:
##                     LD1        LD2
## Sepal.Length  0.7100013 -0.8446128
## Sepal.Width   1.2435532  2.4773120
## Petal.Length -2.3419418 -0.4065865
## Petal.Width  -1.8502355  2.3234441
##
## Proportion of trace:
##    LD1    LD2
## 0.9908 0.0092
```

Here we see the estimated $\hat{\pi}_k$ and $\hat{\mu}_k$ for each class.

```
is.list(predict(iris_lda, iris_trn))
```

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

```
names(predict(iris_lda, iris_trn))
```

```
## [1] "class"     "posterior" "x"
```

```
head(predict(iris_lda, iris_trn)$class, n = 10)
```

```
##  [1] setosa     virginica  setosa     setosa     virginica  setosa
##  [7] virginica  setosa     versicolor setosa
## Levels: setosa versicolor virginica
```

```
head(predict(iris_lda, iris_trn)$posterior, n = 10)
```

```
##           setosa   versicolor    virginica
## 23  1.000000e+00 1.517145e-21 1.717663e-41
## 106 2.894733e-43 1.643603e-06 9.999984e-01
## 37  1.000000e+00 2.169066e-20 1.287216e-40
## 40  1.000000e+00 3.979954e-17 8.243133e-36
## 145 1.303566e-37 4.335258e-06 9.999957e-01
## 36  1.000000e+00 1.947567e-18 5.996917e-38
## 119 2.220147e-51 9.587514e-09 1.000000e+00
## 16  1.000000e+00 5.981936e-23 1.344538e-42
## 94  1.599359e-11 9.999999e-01 1.035129e-07
## 27  1.000000e+00 8.154612e-15 4.862249e-32
```

Getting Predictions and Error Rates

```
iris_lda_trn_pred = predict(iris_lda, iris_trn)$class
iris_lda_tst_pred = predict(iris_lda, iris_tst)$class
```

We store the predictions made on the train and test sets.

```
calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}
```

```
calc_class_err(predicted = iris_lda_trn_pred, actual = iris_trn$Species)
```

```
## [1] 0.04
```

```
calc_class_err(predicted = iris_lda_tst_pred, actual = iris_tst$Species)
```

```
## [1] 0.01333333
```

As expected, LDA performs well on both the train and test data.

```
table(predicted = iris_lda_tst_pred, actual = iris_tst$Species)
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         26         1
##   virginica       0          0        26
```

QDA

Guess what?! There is a `qda` function. Yay!

```
iris_qda = qda(Species ~ ., data = iris_trn)
iris_qda
```

```
## Call:
## qda(Species ~ ., data = iris_trn)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3733333  0.3200000  0.3066667
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa         4.978571    3.378571     1.432143   0.2607143
## versicolor     5.995833    2.808333     4.254167   1.3333333
## virginica      6.669565    3.065217     5.717391   2.0956522
```

```
iris_qda_trn_pred = predict(iris_qda, iris_trn)$class
iris_qda_tst_pred = predict(iris_qda, iris_tst)$class
```

The `predict()` function operates the same as the `predict()` function for LDA.

```
calc_class_err(predicted = iris_qda_trn_pred, actual = iris_trn$Species)
```

```
## [1] 0.01333333
```

```
calc_class_err(predicted = iris_qda_tst_pred, actual = iris_tst$Species)
```

```
## [1] 0.04
```

```
table(predicted = iris_qda_tst_pred, actual = iris_tst$Species)
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         23         0
##   virginica       0          3        27
```

Logistic Regression

In lecture, you were told that Logistic Regression does not get used for more than 2 groups all that much. Let's see how it performs anyway.

In this case, the `glm` function cannot be used. Instead a function from the `nnet` packages will be used.

```
library(nnet)

iris_log=multinom(Species~., data=iris_trn)
```

```
## # weights:  18 (10 variable)
## initial  value 82.395922
## iter  10 value 6.164184
## iter  20 value 3.952967
## iter  30 value 3.887791
## iter  40 value 3.826736
```

```
## iter  50 value 3.766389
## iter  60 value 3.755861
## iter  70 value 3.747011
## iter  80 value 3.746702
## iter  90 value 3.745246
## final  value 3.745179
## converged
```

```
summary(iris_log)
```

```
## Call:
## multinom(formula = Species ~ ., data = iris_trn)
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    26.81602    -6.983313   -16.24574     20.35750    3.218787
## virginica    -34.24228    -8.398869   -17.03985     31.94659   11.594518
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor     74.3553     226.5881    115.8998     67.37825    16.14625
## virginica      76.2057     226.6131    115.6749     66.96215    16.04269
##
## Residual Deviance: 7.490358
## AIC: 27.49036
```

```
iris_log_trn_pred = predict(iris_log, iris_trn, "class")
iris_log_tst_pred = predict(iris_log, iris_tst, "class")
```

The `predict()` function operates the same as the `predict()` function for LDA.

```
calc_class_err(predicted = iris_log_trn_pred, actual = iris_trn$Species)
```

```
## [1] 0.04
```

```
calc_class_err(predicted = iris_log_tst_pred, actual = iris_tst$Species)
```

```
## [1] 0.06666667
```

```
table(predicted = iris_log_tst_pred, actual = iris_tst$Species)
```

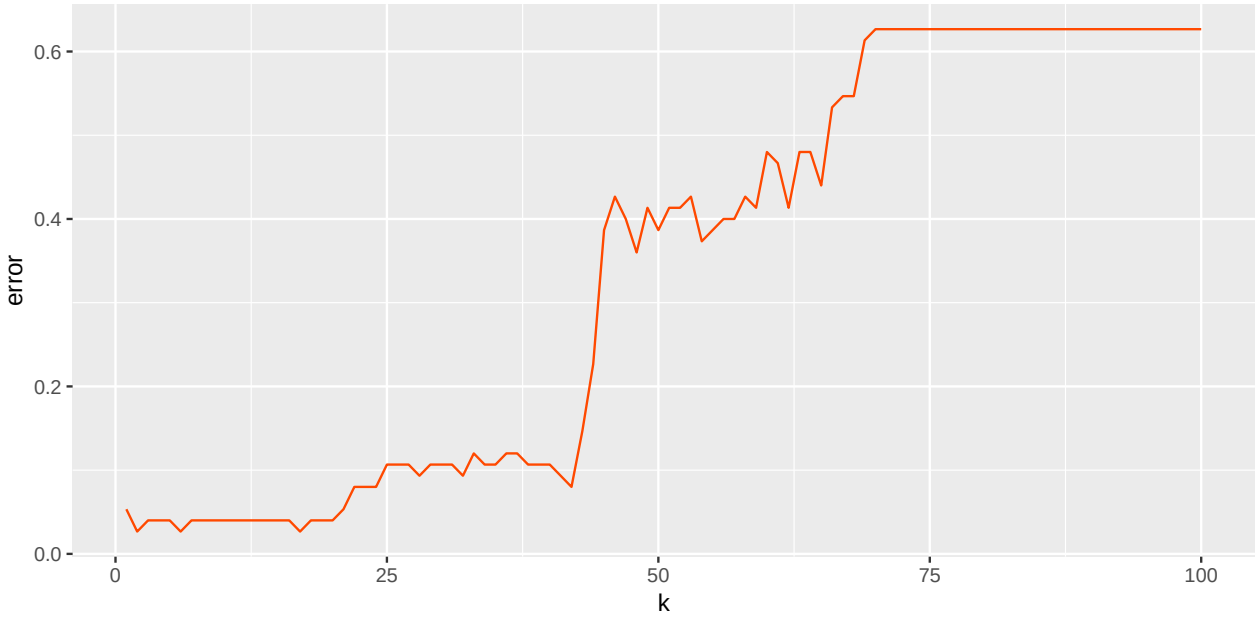
```
##             actual
## predicted    setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         26         5
##   virginica       0          0        22
```

KNN

```
library(class)

kmax = 100
err = double(kmax)
for(ii in 1:kmax){
  pk = knn.cv(iris_trn[,-5],iris_trn$Species, k=ii) # does leave one out CV
  err[ii] = mean(pk != iris_trn$Species)
```

```
}
ggplot(data.frame(k=1:kmax,error=err), aes(k,error))  +
  geom_line(color=red)
```




```
best.k <- max(which(err == min(err)))

best.k
```

```
## [1] 17
```

```
iris_knn_trn_pred = knn(iris_trn[,-5], iris_trn[,-5], iris_trn$Species, k = best.k)
iris_knn_tst_pred = knn(iris_trn[,-5], iris_tst[,-5], iris_trn$Species, k = best.k)
```

```
calc_class_err(predicted = iris_knn_trn_pred, actual = iris_trn$Species)
```

```
## [1] 0.02666667
```

```
calc_class_err(predicted = iris_knn_tst_pred, actual = iris_tst$Species)
```

```
## [1] 0.06666667
```