Consider a remote sensing study[1] that involved detecting diseased trees in Quickbird[2] satellite imagery. Aim: find the `class` of trees (wilted/not wilted) based on the color of emitted light, as viewed from orbit. Load the training and testing datasets which can be found in the `msc` folder online.

```
> D=read.csv("wilt_training.csv")
> D2=read.csv("wilt_testing.csv")
> str(D)
```

Find the following columns in both the training and test datasets

`class` — Factor w/ 2 levels "n","w": wilted or not

`GLCM` — num: GLCM mean texture (panchromatic band)

`Green` — num: mean in green band (520–600 nm)

`Red` — num: mean in red band (630–690 nm)

`NIR` — num: mean in near IR band (760–890 nm)

`SD` — num: standard deviation in panchromatic band

This data has what should be an unusual problem: the wilted fraction in the training set is much less than the wilted fraction in the testing set. In the homework version of this problem you will work to develop fewer and more discriminating features than those in the raw file and to 'push' the KNN analysis to sense more wilted trees without harming too much the overall accuracy. This is the final result:

```
predict2   n   w
   FALSE 276  62
   TRUE   37 125
> (276+125)/500
[1] 0.802
> 125/187
[1] 0.6684492
```

an overall accuracy of 80.2% and a power of 66.8%.

Today we will work to better those numbers using Linear Discriminant Analysis and Quadratic Discriminant Analysis. Following the process outlined in the homework the training set has become:

```
> str(dscaled)
'data.frame': 4339 obs. of  4 variables:
 $ NIR  : num  -0.76069 -1.1636 -0.36407 -1.65462 -0.00746 ...
 $ SD   : num  -0.386 -0.746 -0.221 -0.904 -0.665 ...
 $ diff : num  3 2.84 3.32 2.68 3.12 ...
 $ class: Factor w/ 2 levels "n","w": 2 2 2 2 2 2 2 2 2 2 ...
```

and the testing set is called `d2scaled`. Functions `lda` and `qda` are in the `MASS` library[3]; they use a formula argument.

[1]Johnson, B., Tateishi, R., Hoan, N., 2013. International Journal of Remote Sensing, 34 (20), 6969–6982.
[2]`https://en.wikipedia.org/wiki/QuickBird`
[3]The name comes from the textbook: "Modern Applied Statistics with S" (Springer, 4th, 2003)

```
> library(MASS)
> lda1=lda(class~diff+SD+NIR,data=dscaled)
> lda1

Call:
lda(class ~ diff + SD + NIR, data = dscaled)

Prior probabilities of groups:
         n          w
0.98294538 0.01705462

Group means:
         diff           SD         NIR
n -0.05236526  0.007236775  0.01307415
w  3.01807860 -0.417092500 -0.75353053

Coefficients of linear discriminants:
            LD1
diff  1.1865514
SD   -0.1661454
NIR   0.2888952

> lda.predict=predict(lda1,d2scaled)
> table(lda.predict$class,d2scaled$class])

      n   w
  n 303  95
  w  10  92

> (303+92)/500
[1] 0.79
> (92)/187
[1] 0.4919786
```

SO with the initial attempt the accuracy is comparable to KNN but the power is worse. LDA produces a parameter `x` which is essentially the distance from the plane that divides the cases. (Note that we are about to look at the test dataset `class` and essentially train on it, something that we should avoid in real life.) You can see that there is good separation with `predict` drawing the separating line at about 2.85.

```
> hist(lda.predict$x[d2scaled$class=="n"],breaks=seq(-8,6,.5),col=rgb(0,0,1,.25))
> hist(lda.predict$x[d2scaled$class=="w"],breaks=seq(-8,6,.5),col=rgb(1,0,0,.25),add=T)
```

It should be clear that reducing that dividing line will add more actual `w` (red) than `n` (blue) into the predicted `w` class.

```
> table(lda.predict$x>2.5,d2scaled$class)

          n   w
  FALSE 298  71
  TRUE   15 116
> (298+116)/500
```

```
[1] 0.828
> 116/187
[1] 0.6203209

> table(lda.predict$x>2.25,d2scaled$class)

          n   w
  FALSE 293  53
  TRUE   20 134
> (293+134)/500
[1] 0.854
> 134/187
[1] 0.7165775

> table(lda.predict$x>2.0,d2scaled$class)

          n   w
  FALSE 281  29
  TRUE   32 158

> (281+158)/500
[1] 0.878
> 158/187
[1] 0.8449198

> table(lda.predict$x>1.75,d2scaled$class)

          n   w
  FALSE 270  16
  TRUE   43 171

> (270+171)/500
[1] 0.882
> 171/187
[1] 0.9144385

> table(lda.predict$x>1.5,d2scaled$class)

          n   w
  FALSE 251   9
  TRUE   62 178

 > (251+178)/500
[1] 0.858
> 178/187
[1] 0.9518717
```

The trade-off between true positives and false positives as the dividing line is moved is graphically displayed in the ROC (receiver operating characteristics) curve discussed in the textbook (ISLR p. 147). All but the first case beat KNN. Note that in the last case the overall accuracy has been reduced.

Now we try Quadratic Discriminant Analysis.

```
> qda1=qda(class~diff+SD+NIR,data=dscaled)
> qda1

Call:
qda(class ~ diff + SD + NIR, data = dscaled)

Prior probabilities of groups:
         n          w
0.98294538 0.01705462

Group means:
         diff           SD         NIR
n -0.05236526  0.007236775  0.01307415
w  3.01807860 -0.417092500 -0.75353054

> qda.predict=predict(qda1,d2scaled)
> table(qda.predict$class,d2scaled$class)
      n   w
  n 297  88
  w  16  99
> (297+99)/500
[1] 0.792
> 99/187
[1] 0.5294118

> names(qda1)
 [1] "prior"   "counts"  "means"   "scaling" "ldet"    "lev"     "N"
 [8] "call"    "terms"   "xlevels"
> names(qda.predict)
[1] "class"     "posterior"
> summary(qda.predict$posterior)
       n                  w
 Min.   :0.0000692   Min.   :0.0000000
 1st Qu.:0.5972171   1st Qu.:0.0000061
 Median :0.9931003   Median :0.0068997
 Mean   :0.7671353   Mean   :0.2328647
 3rd Qu.:0.9999939   3rd Qu.:0.4027829
 Max.   :1.0000000   Max.   :0.9999308
```

Note that raw LDA beat raw QDA. We can push QDA as we did LDA, but its a bit more complex. `qda.predict$posterior` gives separate p values for the two classes, and selects the largest as the winner. We can bias the result by requiring a difference greater than zero.

```
> table(qda.predict$posterior[,1]-qda.predict$posterior[,2]<.9,d2scaled$class)

          n   w
  FALSE 271  32
  TRUE   42 155
> (271+155)/500
[1] 0.852
> 155/187
[1] 0.828877
```

But this still does not beat LDA.

While I could proceed as above: train using the unbalanced data and test and push on balanced data, instead, for logistic regression, I will reverse: train on `d2scaled` and test on `dscaled`. We use `glm` (generalized linear model) much as we did `lm`, but with logistic family selected `binomial(link='logit')`.

```
> log1=glm(class~.,family=binomial(link='logit'),data=d2scaled)
> out=predict(log1,type="response")
> table(out>.5,d2scaled$class)

         n   w
  FALSE 284  26
  TRUE   29 161

> (284+161)/500
[1] 0.89
> 161/187
[1] 0.8609626
```

The above accuracy and power are pretty good, but they are for the training set this time. The output of `predict` is a vector of probabilities (by default for the training set). We can gain some insight into an appropriate cut-off from the ROC.

```
> library(ROCR)
> pred <- prediction(out,d2scaled$class)
> str(pred)
```

`pred` has a complex structure—its loaded for many different plots— but we don't have to understand it to use it.

```
> roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
> plot(roc.perf)
> abline(a=0, b= 1)
> acc.perf = performance(pred, measure = "acc")
> plot(acc.perf)
> auc.perf = performance(pred, measure = "auc")
> auc.perf@y.values
[[1]]
[1] 0.9528797
```

If we reduce the cut-off, we increase the power and the false detection rate and reduce the accuracy. We now test using `dscale` (unbalanced data)

```
> out2=predict(log1,newdata=dscaled,type="response")
> table(out2>.25,dscaled$class)

          n    w
  FALSE 3912    1
  TRUE   353   73

> pred <- prediction(out2,dscaled$class)
> roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
> plot(roc.perf)
> acc.perf = performance(pred, measure = "acc")
> plot(acc.perf)
```