STOR 565 Fall 2019 Homework 6

Due on 01/31/2018 in Class

Ted Henson

Remark. Credits for **Theoretical Part** and **Computational Part** are in total 100 pt. For **Computational Part**, please complete your answer in the **RMarkdown** file and summit your printed PDF homework created by it.

##Comment If dplyr and MASS are both loaded, you might need to specify `dplyr::select` to specify that you want the dplyr version of the `select` function.

Computational Part

###About the data: Tree leaf images

We will attempt to identify trees based on image data of their leaves. This is a tough problem, though apps such as iNaturalist now do a pretty good job identifying plants from images taken on your phone.

The data set is from here: https://www.kaggle.com/c/leaf-classification/data

Images have been pre-processed, so the dataset inlcudes vectors for margin, shape and texture attributes for each of almost 1000 images. We will focus on the shape attributes, which describe the contours of the leaf in the image.

###A helpful demonstration for SVM

http://uc-r.github.io/svm

###Q1 ###(a) (3 points)

Load the `leaf_train` dataset.

```r
library(readr)
leaf <- read_csv("leaf_train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   species = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

(i) Subset the columns to include only `id`, `species` and the `shape` variables, which is most easily done using the dplyr `select` function and the sub-function `contains`. There should be 66 variables in all.

```r
leaf = dplyr::select(leaf, matches("id|species|shape"))
```

(ii) Then create a new variable `genus` by extracting the first part of the species name. You can use the following code, assuming your data objects are named in a compatible way. You will probably want to load the data with `stringsAsFactors` as false.

```r
leaf$genus = word(leaf$species, start = 1, end = 1, sep = "_")
```

(iii) Lastly, convert the genus variable to a factor.

```r
leaf$genus <- as.factor(leaf$genus)
```

(iv) Display your resulting data frame and the result of `summary(leaf$genus)`, which should give the number of observations of each genus. **Display only the id, species and first two species variables in your output, and only five rows of the data, eg by using the head function.**

```r
head(leaf[, c('id', 'species', 'shape1', 'shape2')])
```

```
## # A tibble: 6 x 4
##      id species                 shape1   shape2
##   <dbl> <chr>                    <dbl>    <dbl>
## 1     1 Acer_Opalus           0.000647 0.000609
## 2     2 Pterocarya_Stenoptera 0.000749 0.000695
## 3     3 Quercus_Hartwissiana  0.000973 0.000910
## 4     5 Tilia_Tomentosa       0.000453 0.000465
## 5     6 Quercus_Variabilis    0.000682 0.000598
## 6     8 Magnolia_Salicifolia  0.000390 0.000347
```

```r
summary(leaf$genus)
```

```
##         Acer        Alnus  Arundinaria       Betula   Callicarpa
##          100           50           10           20           10
##     Castanea       Celtis       Cercis       Cornus      Cotinus
##           10           10           10           30           10
##    Crataegus      Cytisus   Eucalyptus        Fagus       Ginkgo
##           10           10           30           10           10
##         Ilex  Liquidambar Liriodendron  Lithocarpus     Magnolia
##           20           10           10           20           20
##        Morus         Olea  Phildelphus      Populus       Prunus
##           10           10           10           30           20
##   Pterocarya      Quercus Rhododendron        Salix       Sorbus
##           10          380           10           20           10
##        Tilia        Ulmus     Viburnum      Zelkova
##           30           10           20           10
```

(v) Randomly split your data into test and training sets. About 35 percent of the data should be in the test set. Display a summary of genus labels in the training set.

Note: In the rare event that one class in the training data is not represented, you may reduce the test set percentage to 30 percent and resample.

```r
set.seed(400)

#classes were not represented unless it was an even split
smp_size <- floor(0.5 * nrow(leaf))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(leaf)), size = smp_size)

train <- leaf[train_ind, ]
test <- leaf[-train_ind, ]

length(unique(test$genus))
```

```
## [1] 34
```

```r
length(unique(train$genus))
```

```
## [1] 34
```

```r
summary(train$genus)
```

```
##         Acer        Alnus  Arundinaria       Betula   Callicarpa 
##           60           28            3           11            4 
##     Castanea       Celtis       Cercis       Cornus      Cotinus 
##            2            6            7           15            5 
##    Crataegus     Cytisus   Eucalyptus        Fagus       Ginkgo 
##            5            4           19            7            7 
##         Ilex  Liquidambar Liriodendron  Lithocarpus     Magnolia 
##           10            7            3            8            9 
##        Morus         Olea  Phildelphus      Populus       Prunus 
##            5            3            2           14           11 
##   Pterocarya      Quercus Rhododendron        Salix       Sorbus 
##            6          182            4           11            4 
##        Tilia        Ulmus     Viburnum      Zelkova 
##           18            3            8            4
```

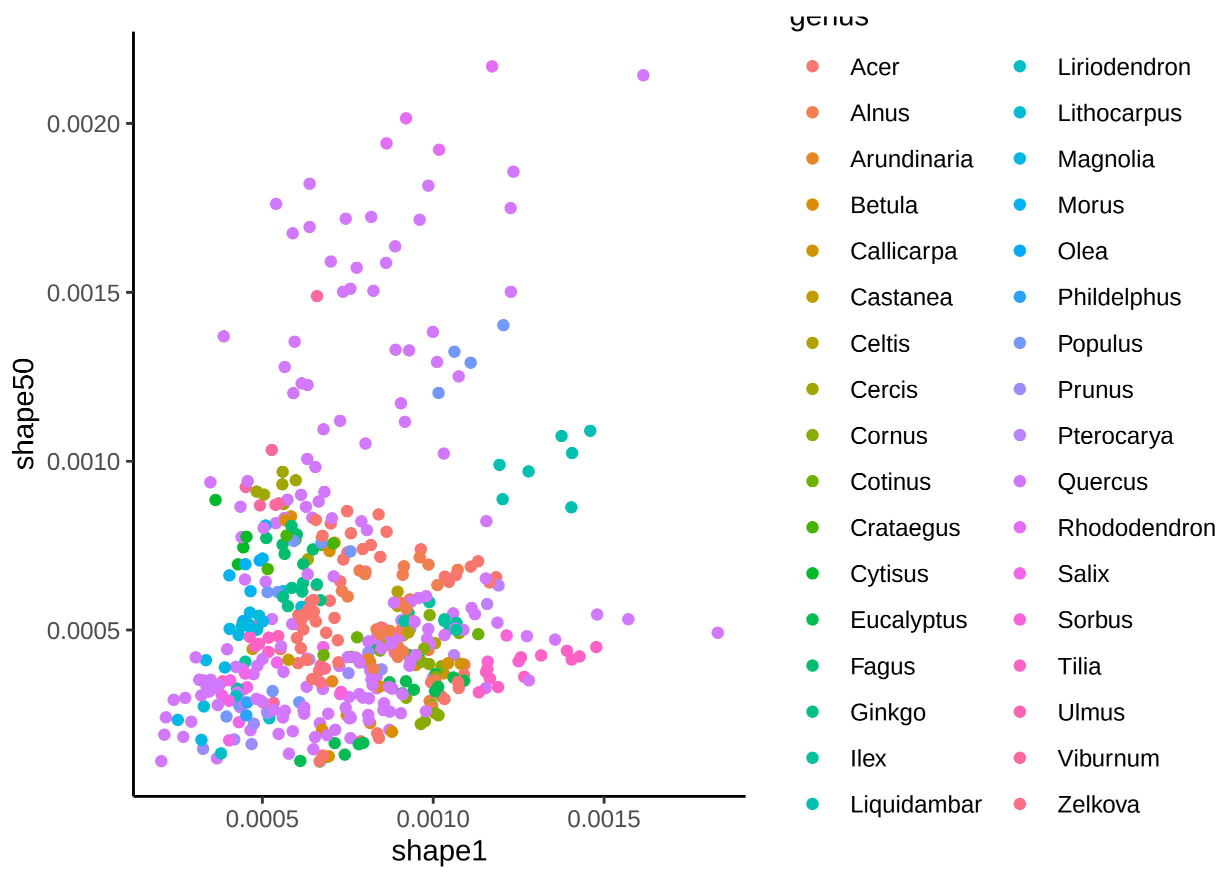
##(b) (2 points)

For the training data:

(i) Make a scatter plot of `shape1` by `shape50`, with some form of genus label. `ggplot2` is probably the best package for this, though you do not need to make the plot fancier than required to display the information above.

(ii) Write two to three sentences discussing some possible implications of this plot for the SVM model.

```r
ggplot(train, aes(shape1, shape50,
                  colour = genus)) + geom_point() + theme_classic()
```



There are many different genera so our svm model may have trouble separating them out; however, the Quercus genus is the clear dominant genus. We could maybe do a better job separating them if we created a new binary class: Quercus and 'Not Quercus'. Additionally, the relationship between shape 1 and shape 50 (and the corresponding genus) may not be linear. We may want to use polynomial svms (and the kernel trick) to separate out the genera better.

##(c) (15 points)

For the training data:

(i) Write a function, or use an available one, to choose the cost parameter for the SVM model on this training data with **linear kernel.** Use **shape variables as predictors only, genus as response**.

Use **5-fold cross validation.** Use the array of costs provided in the code below.

If you use a built-in function, you must state specifically how the best parameter value is chosen, for example by giving the error function minimized. Simply stating `classification error` is insufficient and will receive no points. You must state what that means. If using your own function, you may use any error function you like that is justified for classification problems.

See the demo linked above for help.

This might take some time to run. Do not knit your file at the last minute before the assignment is due.

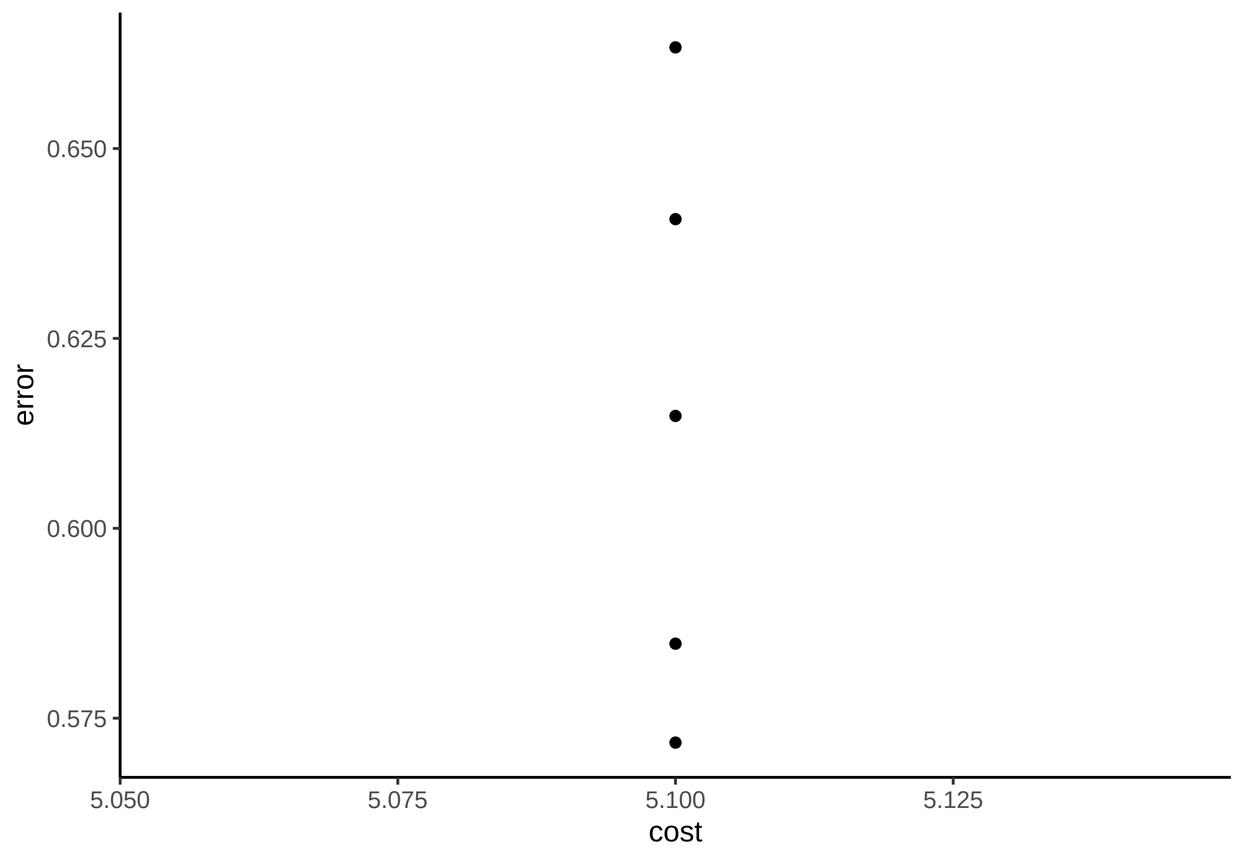
(ii) Report the best value of cost chosen, and plot the errors by the cost values.

(iii) Write two or three sentences discussing some basic implications of your answer in (ii), using the concepts from class. Lecture 7 will be helpful.

```r
cost_out <- seq(from = 0.1, to =5.1, by = 1)
library(caret)
```

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

```r
  train.data = train %>% dplyr::select(-id, -species)

svm.cv.func = function(cost = cost_out, train = train.data){


  #get rid of auto correlated variables


  flds <- createFolds(train$genus, k = 5)


  total.results = data.frame(fld = NULL,
                             cost = NULL,
                             accuracy = NULL)
  for(i in 1:length(flds)){
    fld.results = data.frame(fld = c(0),
                          cost = c(0),
                           accuracy = c(0))

    for(j in 1:length(cost_out)){


  svm.mod = svm(genus ~ ., data = train[flds[[i]],],
                kernel = 'linear',
                cost = cost[j],
               type = "C-classification")

  predictions = predict(svm.mod, train[-c(flds[[i]]),])
  fld.results[i, 'fld'] = i
    fld.results[i, 'cost'] = cost[j]
  fld.results[i, 'accuracy'] = mean(predictions == train$genus[-c(flds[[i]])])

    }
  opt.fld.tune = fld.results[which.max(fld.results$accuracy),]
  total.results = rbind(total.results, opt.fld.tune)
  }
  total.results$error = 1- total.results$accuracy
  return(total.results)
}

svm.out = svm.cv.func()

ggplot(svm.out, aes(cost, error,
                    label = fld)) + geom_point() + theme_classic()
```



```r
print(svm.out)
```

```
##   fld cost  accuracy     error
## 1   1  5.1 0.4151899 0.5848101
## 2   2  5.1 0.3852041 0.6147959
## 3   3  5.1 0.3592965 0.6407035
## 4   4  5.1 0.4282116 0.5717884
## 5   5  5.1 0.3366834 0.6633166
```

Our function performs runs a svm model for each cost parameter for 5 fold cross validation. For each fold, the cost value that yields the greatest classification rate (percent of genera classified correctly) is returned along with the classification rate. For all 5 of our folds, the optimal cost value was 5.1 based on the classification error rate. Each of our folds had about 100 observations to train on so we feel reasonably comfortable that 5.1 is close to the true optimal cost parameter for the linear svm. Our fourth fold had the best classification rate at almost 43%.

##(d) (15 points)

(i) Run the SVM model on the **training data** with **linear kernel** and the cost determined in part (c). If you are unable to do part (c), use a cost of 1, the default. Report a summary of the fitted class label counts.

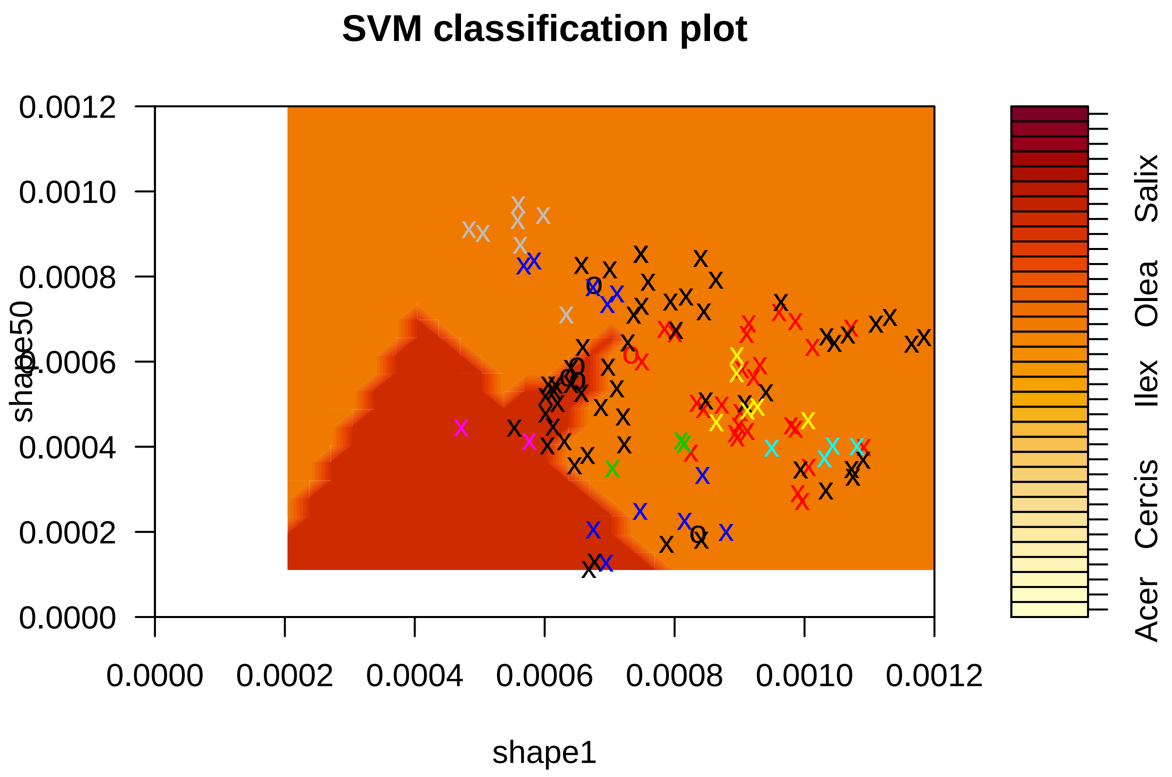
```r
svm.mod.opt  = svm(genus ~ .,
                   data = train.data,
                   kernel = 'linear',
                cost = 5.1,
               type = "C-classification")
```

```r
summary(svm.mod.opt$fitted)
```

```
##         Acer        Alnus  Arundinaria       Betula   Callicarpa
##           33           18            1            5            0
##     Castanea       Celtis       Cercis       Cornus      Cotinus
##            2            0            0           11            2
##    Crataegus      Cytisus   Eucalyptus        Fagus       Ginkgo
##            3            4           14            0            7
##         Ilex  Liquidambar Liriodendron  Lithocarpus     Magnolia
##            7            7            3            8            7
##        Morus         Olea  Phildelphus      Populus       Prunus
##            5            1            0            5            1
##   Pterocarya      Quercus Rhododendron        Salix       Sorbus
##            3          316            4           10            2
##        Tilia        Ulmus     Viburnum      Zelkova
##           11            1            0            4
```

(ii) Create a classification plot from the model, plotting the variables `shape50` by `shape1`. See `?plot.svm`. In your plot statement, use the argument `xlim = c(0, 0.0012), ylim = c(0, 0.0012)`.

```r
plot(svm.mod.opt,
          data = train.data,
          shape50 ~ shape1,
     xlim = c(0, 0.0012), ylim = c(0, 0.0012))
```



See the linked demo for an explanation of the plot. Write two sentences explaining what you see **using concepts and terminology from class.**

The points with marked by an X are the support vectors; i.e. the points that directly affect the classification line (or decision boundary). The points marked by an ‘o’ do not affect the decision boundary. The colors correspond to the genus assigned by our svm model.

(iii) Predict outcomes based on your model in (i) for the test data. Display a confusion matrix and compute sensitivity, specificity statistics. You may use the function demonstrated in class.

Warning: the confusion matrix will be awkward to display. Don’t worry about it so much. The sensitivity and specificity are good summaries.

```r
predictions = predict(svm.mod.opt, test)

confusion <- function(yhat, y, quietly = FALSE){
if(!quietly)
if(!is.factor(y) & is.factor(yhat))
y <- as.factor(y)
if(!all.equal(levels(yhat), levels(y)))
stop("Factor levels of yhat and y do not match.")
confusion_mat <- table(yhat, y, deparse.level = 2)
stats <- data.frame(sensitivity = confusion_mat[1, 1]/sum(confusion_mat[, 1]),
specificity = confusion_mat[2, 2]/sum(confusion_mat[, 2]))
return(stats)
}
# Many actual survivors predicted to die
confusion(yhat = predictions, y = test$genus, quietly = FALSE)
```

```
##   sensitivity specificity
## 1        0.45         0.5
```

##(e) (15 points) This question will use a non-linear kernel for the SVM and compare results.

(i) Modify your function in part (c) to find the optimal cost value for the SVM on the **training data** with **radial kernel** with gamma parameter 0.55. Use the same cost range. Report the optimal cost.

(ii) Run the radial SVM model with these optimal parameters on the training data.

(iii) Repeat part (d)(iii) but for the radial SVM model instead of the linear one.

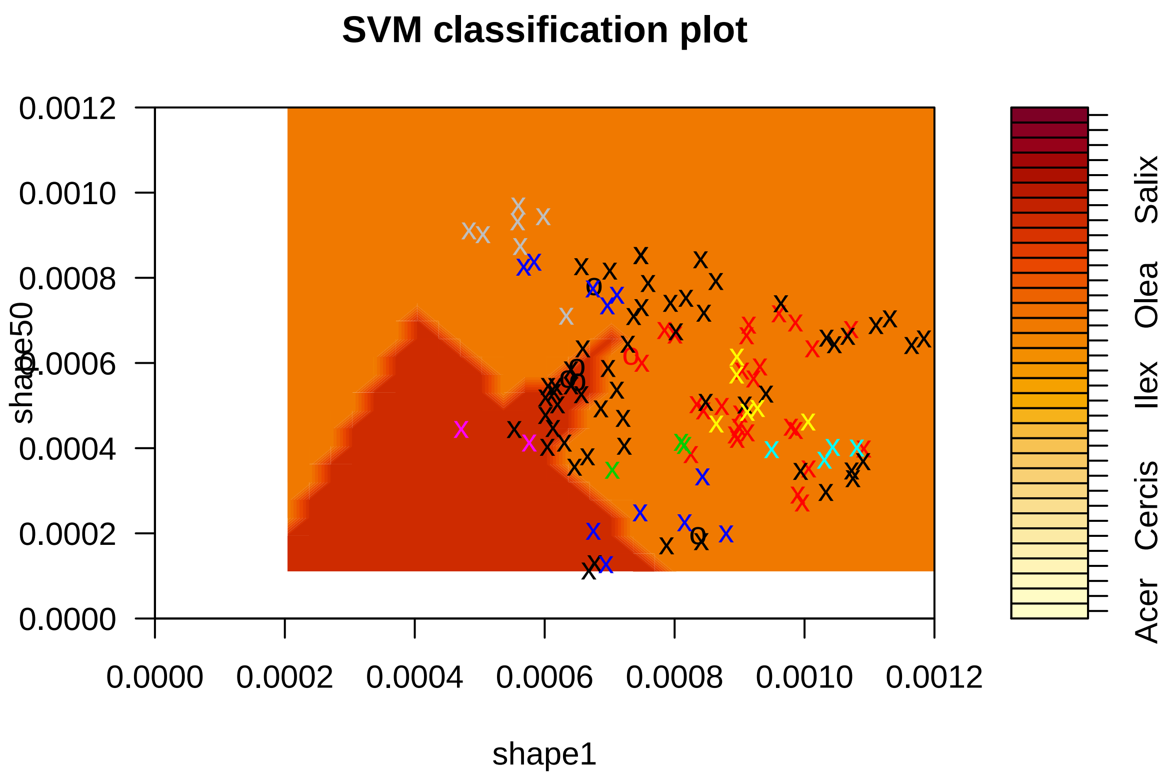
(iv) Discuss briefly your results in (e)(iii) as compared to (d)(iii) **using concepts discussed in class**.

The optimal cost was still 5.1 for each of our folds. The cost parameter was chosen the same way as in the linear case: for each fold, the cost corresponding to the highest classification accuracy rate was returned, along with the classification rate. As in the linear case, we had about 100 observations to train on so we feel pretty comfortable that 5.1 is close to the true optimal cost parameter. The best accuracy was almost 50% by fold 3.

```r
svm.radial.mod.opt  = svm(genus ~ .,
                   data = train.data,
                   kernel = 'radial',
                   gamma = .55,
                cost = 5.1,
               type = "C-classification")

summary(svm.radial.mod.opt$fitted)
```

```
##        Acer       Alnus  Arundinaria      Betula   Callicarpa
##          60          28            3          11            4
##    Castanea      Celtis       Cercis      Cornus      Cotinus
##           2           6            7          15            5
##    Crataegus     Cytisus   Eucalyptus        Fagus       Ginkgo
##            5           4           19            7            7
##         Ilex  Liquidambar Liriodendron  Lithocarpus     Magnolia
##           10            7            3            8            9
##        Morus         Olea  Phildelphus      Populus       Prunus
##            5            3            2           14           11
##   Pterocarya      Quercus Rhododendron        Salix       Sorbus
##            6          182            4           11            4
##        Tilia        Ulmus     Viburnum      Zelkova
##           18            3            8            4
```

```r
plot(svm.mod.opt,
         data = train.data,
         shape50 ~ shape1,
     xlim = c(0, 0.0012), ylim = c(0, 0.0012))
```



```r
predictions = predict(svm.radial.mod.opt, test)

confusion <- function(yhat, y, quietly = FALSE){
if(!quietly)
if(!is.factor(y) & is.factor(yhat))
y <- as.factor(y)
if(!all.equal(levels(yhat), levels(y)))
stop("Factor levels of yhat and y do not match.")
confusion_mat <- table(yhat, y, deparse.level = 2)
stats <- data.frame(sensitivity = confusion_mat[1, 1]/sum(confusion_mat[, 1]),
specificity = confusion_mat[2, 2]/sum(confusion_mat[, 2]))
return(stats)
}
# Many actual survivors predicted to die
confusion(yhat = predictions, y = test$genus, quietly = FALSE)
```

```
##   sensitivity specificity
## 1       0.725   0.6363636
```

Our sensitivity and specificity were both higher for our radial model than our linear svm. As shown in the plot above, it would be easier to divide our genera with non linear classification boundaries. As discussed in the beginning, there may be a non linear relationship between Shape50, Shape1, and genus, and our radial svm was able to capture that relationship better than the linear svm.