Assignment 2

Which filter do you return to the user ? filter0, filter1, filter2 or filter3? Why?

At first glance, it would make sense to return *filter3* since it has a significantly lower error (0.0287) than the other three filters. However, this model has been trained on the entire spam data set and its performance is then evaluated on the test data set, which is a subset of the spam data set i.e., the model is evaluated on data points it has already been trained on. This naturally results in a low error rate for the model.

In comparison, the other models are trained and evaluated on different data sets which makes the calculated errors for those models more meaningful. Though for some reason the *validation* data set is used for the predict function in *filter0* instead of the *test* data set which is used in *filter1* and *filter2* which means that we cannot compare them directly. Based on this, we would choose either *filter0* or *filter2.*

Another aspect to consider, is the complexity of the model. Since *filter0* (and *filter1*) is only trained on the train data set, the number of support vectors are fewer than because the models are trained on fewer data points.

What is the estimate of the generalization error of the filter returned to the user? err0, err1, err2 or err3? Why?

- **err0:** 0.0625
- **err1:** 0.07615481
- **err2:** 0.06991261
- **err3:** 0.02871411

As mentioned in the previous question, the generalization error rate is significantly lower for *filter3* since it is trained and tested on the same data. The other models are trained and validated on different data sets which naturally result in a higher error rate. Interestingly, *filter2* which is trained on both the train and validation data sets, performs worse than *filter0* which is only trained on the train data set. This is probably because *filter0* is evaluated on a different data set than the other filters.

Implementation of SVM predictions.

Our own implementation of the predict function yields the exact same results as the predict function. This makes sense since our implementation and the predict function uses the same data points, support vectors and kernel function with the same value for sigma.

Appendix: Code for assignment 2

```
## [1] 0.0625
```

```
## [1] 0.07615481
```

```
## [1] 0.06991261
```

```
## [1] 0.02871411
```

```
## [1] "Our implementation"
##              [,1]
##  [1,]  0.8185775
##  [2,]  0.9999825
##  [3,]  2.2142890
##  [4,] -2.5851401
##  [5,]  1.7727880
##  [6,] -1.2596537
##  [7,]  1.0026088
##  [8,]  2.5794347
##  [9,]  0.9998688
## [10,]  1.2206699
```

```
## [1] "prediction function"
```

```
##              [,1]
##  [1,]  0.8185775
##  [2,]  0.9999825
##  [3,]  2.2142890
##  [4,] -2.5851401
##  [5,]  1.7727880
##  [6,] -1.2596537
##  [7,]  1.0026088
##  [8,]  2.5794347
##  [9,]  0.9998688
## [10,]  1.2206699
```

```r
#Code from Lab3Block1_2021_SVMs_St.R with some modifications
set.seed(12345)
library(kernlab)

data(spam)
foo <- sample(nrow(spam))
spam <- spam[foo,]
spam[,-58]<-scale(spam[,-58])
tr <- spam[1:3000, ] #train
va <- spam[3001:3800, ] #validation
trva <- spam[1:3800, ] #train and validation
te <- spam[3801:4601, ]  #test

by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=i,scaled=FALSE)
  mailtype <- predict(filter,va[,-58])
  t <- table(mailtype,va[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}

filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FALSE
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
err0
```

```
filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FALSE
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
err1

filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FAL
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
err2

filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by,scaled=FAL
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3

# Questions

# 1. Which filter do we return to the user ? filter0, filter1, filter2 or filter3? Why?

# 2. What is the estimate of the generalization error of the filter returned to the user? err0, err1, e

# 3. Implementation of SVM predictions.

sv<-alphaindex(filter3)[[1]] #return index of support vectors
co<-coef(filter3)[[1]] # linear coefficients of support vectors
inte<- - b(filter3) # negative linear intercept of linear combination
train_10 = spam[1:10, -58]
kernel = rbfdot(sigma = 0.05) #Radial Basis kernel function "Gaussian"
pred=matrix(0, nrow=10, ncol=1)
for(i in 1:10){ # We produce predictions for just the first 10 points in the dataset

  sample = unlist(train_10[i,])
  pred_new = inte
  for(j in 1:length(sv)){
    sv_values <- unlist(spam[sv[j], -58]) #return all columns except 58 of the sv for the given index
    pred_new <- pred_new + co[j]*kernel(sample, sv_values)
  }
  pred[i] = pred_new
}
print("Our implementation")
pred
print("prediction function")
predict(filter3,spam[1:10,-58], type = "decision")
```