HW9

David Schultheiss

11/9/2020

Problem 1

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages -------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.6.2

## -- Conflicts ----------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

a)

```
data = Default %>%
  dplyr::select(-default)

ggplot(data, aes(x = income, y = balance, color = student)) +
  geom_point(shape= '.')
```

The dat appears to be approximately seperable by a linear-boundary.

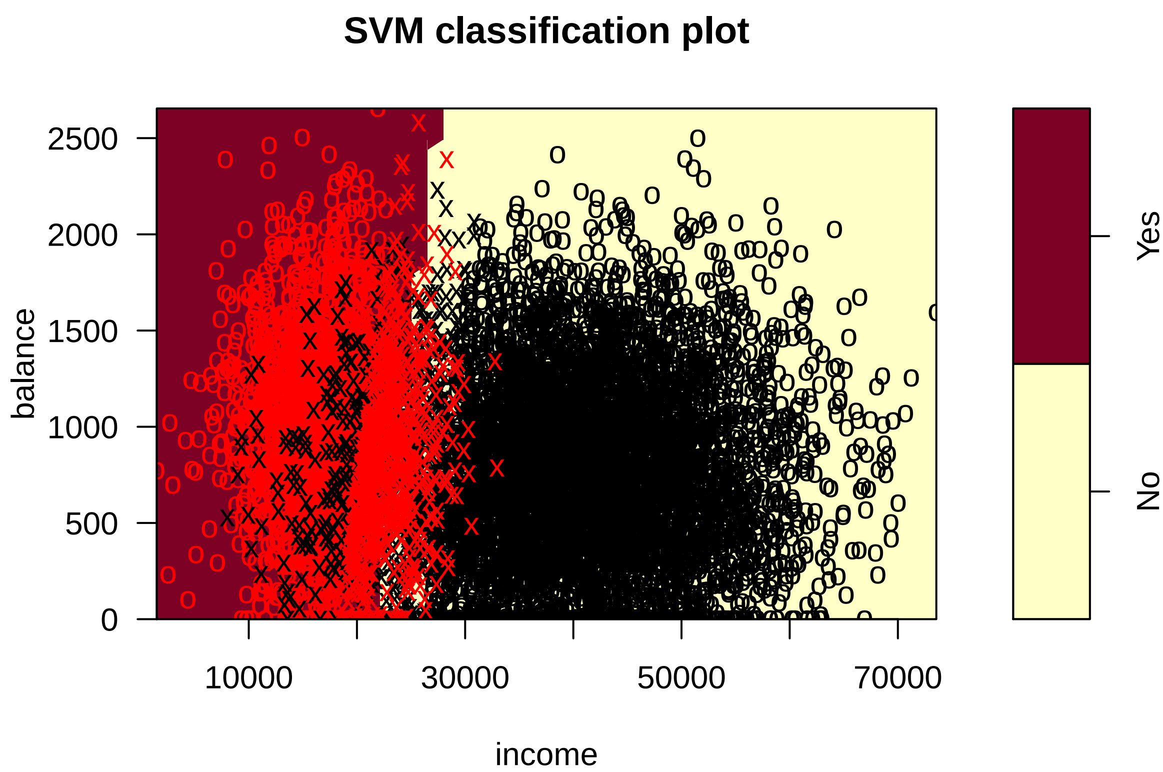
b)

```
set.seed(1)
train = sample(1:nrow(data), .8*nrow(data))
test = data[-train, ]
training = data[train, ]
```

c)

```
svmfit = svm(data= training, as.factor(student)~., kernel= 'linear',
             cost= 1)

plot(svmfit, training)
```




```r
summary(svmfit)
```

```
##
## Call:
## svm(formula = as.factor(student) ~ ., data = training, kernel = "linear",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  1314
##
##  ( 657 657 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
svmpred = predict(svmfit, training, type= 'class')
mean(svmpred != training$student)
```

```
## [1] 0.060625
```

```r
svmpred = predict(svmfit, test, type= 'class')
mean(svmpred != test$student)
```

```
## [1] 0.0665
```

From the summary, we see there are 1314 support vectors, with 657 of each class. Training misclassification rate is 6.06%, Test misclassification rate is 6.65%.

d)

```r
svmtune = tune(svm, data= training, as.factor(student)~., kernel= 'linear',
               ranges = list(cost= c(.001, .01, .1, 1, 2, 4, 6, 8, 10, 100)))
summary(svmtune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.060375
##
## - Detailed performance results:
##     cost    error  dispersion
## 1  1e-03 0.061750 0.008664262
## 2  1e-02 0.060625 0.008212668
## 3  1e-01 0.060375 0.010442468
## 4  1e+00 0.060500 0.009916317
## 5  2e+00 0.060625 0.009686940
## 6  4e+00 0.060500 0.009916317
## 7  6e+00 0.060500 0.009916317
## 8  8e+00 0.060500 0.009916317
## 9  1e+01 0.060500 0.009916317
## 10 1e+02 0.060500 0.009916317
```

Our best cost value is .01.

e)

```r
svmpred = predict(svmtune$best.model, training, type= 'class')
mean(svmpred != training$student)
```

```
## [1] 0.0605
```

```r
svmpred = predict(svmtune$best.model, test, type= 'class')
mean(svmpred != test$student)
```

```
## [1] 0.067
```

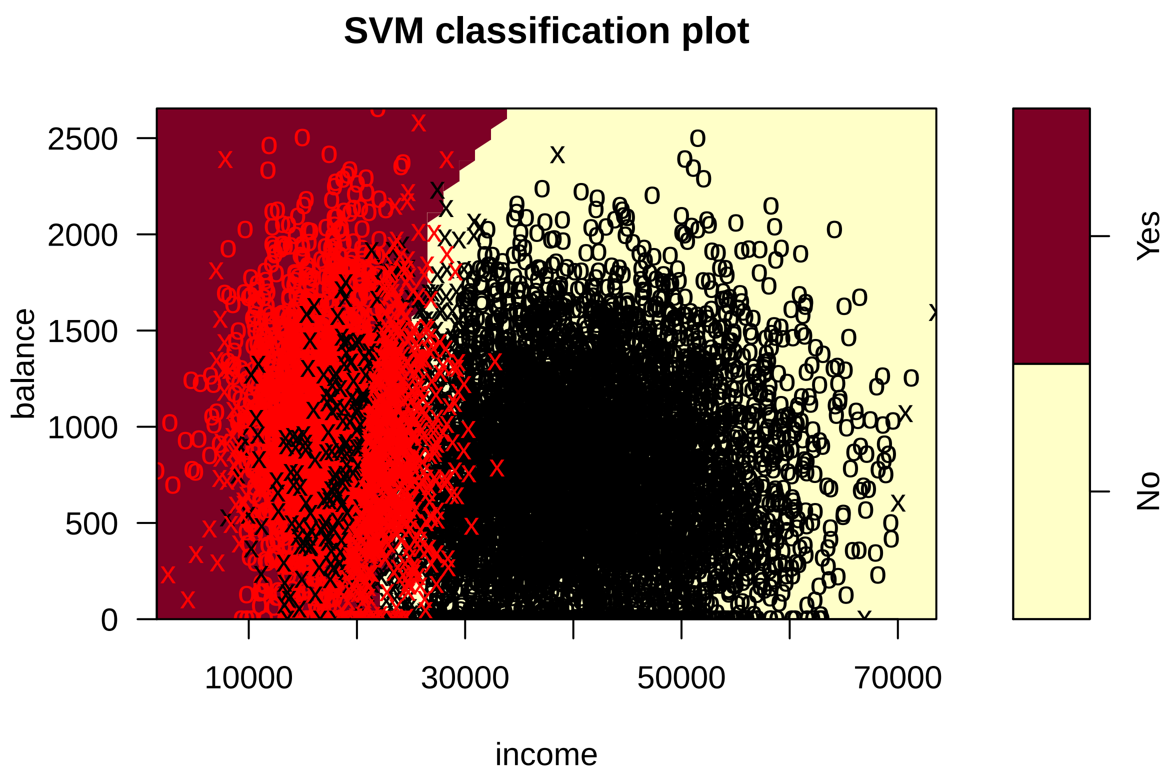
Training misclassification is 6.05%, test is 6.8%.

f)

```
#c
svmfit = svm(data= training, as.factor(student)~., kernel= 'radial',
             cost= 1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = as.factor(student) ~ ., data = training, kernel = "radial",
##     cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1255
##
##  ( 628 627 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
plot(svmfit, training)
```




```r
#d
svmtune = tune(svm, data= training, as.factor(student)~., kernel= 'radial',
               ranges = list(cost= c(.001, .01, .1, 1, 2, 4, 6, 8, 10, 100)))
summary(svmtune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.061125
##
## - Detailed performance results:
##     cost    error  dispersion
## 1  1e-03 0.224625 0.024819137
## 2  1e-02 0.062375 0.007487258
## 3  1e-01 0.061125 0.005415064
## 4  1e+00 0.061750 0.006213784
## 5  2e+00 0.061750 0.006351946
## 6  4e+00 0.062625 0.007008180
## 7  6e+00 0.062625 0.007510409
## 8  8e+00 0.062625 0.007510409
## 9  1e+01 0.062750 0.007402139
## 10 1e+02 0.062875 0.007218081
```

```r
#Best value for cost is 0.1

#e
svmpred = predict(svmtune$best.model, training, type= 'class')
mean(svmpred != training$student)
```

```
## [1] 0.0605
```

```r
#Training misclass rate is 6.05%

svmpred = predict(svmtune$best.model, test, type= 'class')
mean(svmpred != test$student)
```

```
## [1] 0.068
```

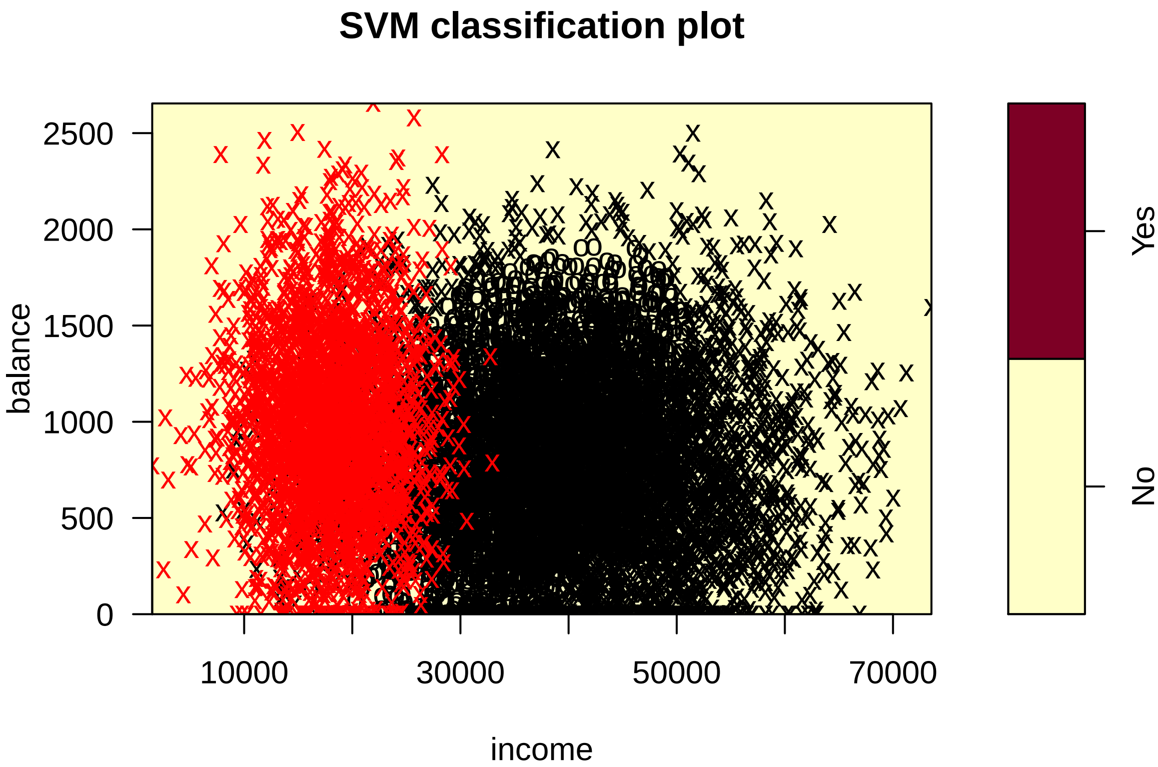
```r
#Test misclass rate is 6.8%
```

g)

```r
#c
svmfit = svm(data= training, as.factor(student)~., kernel= 'polynomial',
             degree= 2, cost= 1)
summary(svmfit)
```

```
##
## Call:
## svm(formula = as.factor(student) ~ ., data = training, kernel = "polynomial",
##     degree = 2, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  4732
##
##  ( 2375 2357 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
plot(svmfit, training)
```




```r
#d
svmtune = tune(svm, data= training, as.factor(student)~., kernel= 'polynomial', degree= 2,
               ranges = list(cost= c(.001, .01, .1, 1, 2, 4, 6, 8, 10, 100)))
summary(svmtune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.001
##
## - best performance: 0.294625
##
## - Detailed performance results:
##     cost    error dispersion
## 1  1e-03 0.294625 0.01044247
## 2  1e-02 0.294625 0.01044247
## 3  1e-01 0.294625 0.01044247
## 4  1e+00 0.294625 0.01044247
## 5  2e+00 0.294625 0.01044247
## 6  4e+00 0.294625 0.01044247
## 7  6e+00 0.294625 0.01044247
## 8  8e+00 0.294625 0.01044247
## 9  1e+01 0.294625 0.01044247
## 10 1e+02 0.294625 0.01044247
```

```r
#Best value for cost is .001

#e
svmpred = predict(svmtune$best.model, training, type= 'class')
mean(svmpred != training$student)
```

```
## [1] 0.294625
```

```r
#Training misclass rate is 29.46%

svmpred = predict(svmtune$best.model, test, type= 'class')
mean(svmpred != test$student)
```

```
## [1] 0.2935
```

```r
#Test misclass rate is 29.35%
```

h) Linear and radial kernels give us about the same misclassification rates. Linear is probably the best for this data set, since the boundary is linear. Also, running the linear kernel seems to be a lot easier on my computer!

Problem 2

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
ldafit = lda(data = training, student~.)

lda.pred = predict(ldafit, training, type='class')
mean(training$student != lda.pred$class)
```

```
## [1] 0.06575
```

```r
lda.pred = predict(ldafit, test, type='class')
mean(test$student != lda.pred$class)
```

```
## [1] 0.072
```

Training error for LDA is 6.58%, and test error is 7.2%. The support vector machine performs better than LDA for this data set.