Report

Sara Lemus

12/9/2021

```
library(foreign)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

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

```
library(e1071)
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(tidyverse)
library(patchwork)
library(UBL)
```

```
## Loading required package: MBA
```

```
## Loading required package: gstat
```

```
## Loading required package: automap
```

```
## Loading required package: sp
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
sesame <- read.dta("sesame.dta")
sesame <- sesame %>%
  mutate(site=factor(site)) %>%
  mutate(bodyDiff = postbody - prebody,
         letDiff = postlet - prelet,
         formDiff = postform - preform,
         numbDiff = postnumb - prenumb,
         relatDiff = postrelat - prerelat,
         clasfDiff = postclasf - preclasf)
sesame.sd <- sesame%>%
  mutate(sd_pBod = scale(prebody, center = TRUE, scale = TRUE),
         sd_plet = scale(prelet, center = TRUE, scale = TRUE),
         sd_pform = scale(preform, center = TRUE, scale = TRUE),
         sd_pnumb = scale(prenumb, center = TRUE, scale = TRUE),
         sd_prelat = scale(prerelat, center = TRUE, scale = TRUE),
         sd_pclasf = scale(preclasf, center = TRUE, scale = TRUE),
         sd_peabody = scale(peabody, center = TRUE, scale = TRUE),
         sd_age = scale(age, center =TRUE, scale = TRUE),
         male=if_else(sex==1, 1, 0),
         female=if_else(sex==2, 1, 0))
```

Q.2 Classification Question: Can we use the pre-test scores and other demographic variables to predict which region the children came from?

SVM

```
set.seed(3241)

n <- nrow(sesame)
train.index <- sample(1:n, size = floor(0.7*n), replace=FALSE)
train.data <- sesame.sd[train.index,]
test.data <- sesame.sd[-train.index,]

train.data %>%
  count(site)
```

```
##   site  n
## 1    1 40
## 2    2 42
## 3    3 48
## 4    4 25
## 5    5 13
```

```
set.seed(315)
costs <- c(0.001, 0.01, 0.1, 1, 5, 10, 100)
# c(0.1, 0.2, 0.5, 0.7, 1, 2, 3, 4)
gammas <- seq(0, 4, by=0.1)
degrees <- c(1,2,3,4,5)

linear.tune <- tune(svm, site~female+ male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+sd_pcl
                    data=train.data, kernel="linear",
                    ranges=list(cost=costs))

radial.tune <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+sd_pc
                    data=train.data, kernel="radial",
                    ranges=list(cost=costs,
                                gamma=gammas))

sigmoid.tune <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+sd_p
                    data=train.data, kernel="sigmoid",
                    ranges=list(cost=costs,
                                gamma=gammas))

poly.tune <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+sd_pcla
                  data=train.data, kernel="polynomial",
                  ranges=list(cost=costs,
                              degree=degrees))
```

```
linear.cm <- table(true=test.data[, "site"],
                           pred=predict(linear.tune$best.model, newdata=test.data))

radial.cm <- table(true=test.data[, "site"],
                           pred=predict(radial.tune$best.model, newdata=test.data))

sigmoid.cm <- table(true=test.data[,"site"],
                    pred=predict(sigmoid.tune$best.model, newdata=test.data))
```

```
poly.cm <- table(true=test.data[, "site"],
                 pred=predict(poly.tune$best.model, newdata=test.data))
```

```
confusionMatrix(linear.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  2  5 13  0  0
##    2  0  8  5  0  0
##    3  1  1 14  0  0
##    4  0  4 14  0  0
##    5  0  1  4  0  0
##
## Overall Statistics
##
##                Accuracy : 0.3333
##                  95% CI : (0.2266, 0.4543)
##     No Information Rate : 0.6944
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1523
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.66667   0.4211   0.2800       NA       NA
## Specificity           0.73913   0.9057   0.9091     0.75  0.93056
## Pos Pred Value        0.10000   0.6154   0.8750       NA       NA
## Neg Pred Value        0.98077   0.8136   0.3571       NA       NA
## Prevalence            0.04167   0.2639   0.6944     0.00  0.00000
## Detection Rate        0.02778   0.1111   0.1944     0.00  0.00000
## Detection Prevalence  0.27778   0.1806   0.2222     0.25  0.06944
## Balanced Accuracy     0.70290   0.6634   0.5945       NA       NA
```

```
confusionMatrix(radial.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  7  3 10  0  0
##    2  3  6  4  0  0
##    3  0  2 14  0  0
##    4  2  4 12  0  0
##    5  0  1  4  0  0
##
## Overall Statistics
##
##                Accuracy : 0.375
##                  95% CI : (0.2636, 0.497)
```

```
##     No Information Rate : 0.6111
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1964
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.58333  0.37500   0.3182       NA       NA
## Specificity           0.78333  0.87500   0.9286     0.75  0.93056
## Pos Pred Value        0.35000  0.46154   0.8750       NA       NA
## Neg Pred Value        0.90385  0.83051   0.4643       NA       NA
## Prevalence            0.16667  0.22222   0.6111     0.00  0.00000
## Detection Rate        0.09722  0.08333   0.1944     0.00  0.00000
## Detection Prevalence  0.27778  0.18056   0.2222     0.25  0.06944
## Balanced Accuracy     0.68333  0.62500   0.6234       NA       NA
```

```
confusionMatrix(sigmoid.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  1  7 12  0  0
##    2  1  8  4  0  0
##    3  3  1 12  0  0
##    4  3  4 11  0  0
##    5  0  1  4  0  0
##
## Overall Statistics
##
##                Accuracy : 0.2917
##                  95% CI : (0.1905, 0.4107)
##     No Information Rate : 0.5972
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0962
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.12500   0.3810   0.2791       NA       NA
## Specificity           0.70312   0.9020   0.8621     0.75  0.93056
## Pos Pred Value        0.05000   0.6154   0.7500       NA       NA
## Neg Pred Value        0.86538   0.7797   0.4464       NA       NA
## Prevalence            0.11111   0.2917   0.5972     0.00  0.00000
## Detection Rate        0.01389   0.1111   0.1667     0.00  0.00000
## Detection Prevalence  0.27778   0.1806   0.2222     0.25  0.06944
## Balanced Accuracy     0.41406   0.6415   0.5706       NA       NA
```

```
confusionMatrix(poly.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  9  2  9  0  0
##    2  4  5  4  0  0
##    3  3  1 12  0  0
##    4  3  3 12  0  0
##    5  0  1  4  0  0
##
## Overall Statistics
##
##                Accuracy : 0.3611
##                  95% CI : (0.2512, 0.4829)
##     No Information Rate : 0.5694
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.1703
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.4737  0.41667   0.2927       NA       NA
## Specificity            0.7925  0.86667   0.8710     0.75  0.93056
## Pos Pred Value         0.4500  0.38462   0.7500       NA       NA
## Neg Pred Value         0.8077  0.88136   0.4821       NA       NA
## Prevalence             0.2639  0.16667   0.5694     0.00  0.00000
## Detection Rate         0.1250  0.06944   0.1667     0.00  0.00000
## Detection Prevalence   0.2778  0.18056   0.2222     0.25  0.06944
## Balanced Accuracy      0.6331  0.64167   0.5818       NA       NA
```

```
set.seed(315)
total.weight <- 60+55+64+43+18
weight.1 <- total.weight/(5*60)
weight.2 <- total.weight/(5*55)
weight.3 <- total.weight/(5*64)
weight.4 <- total.weight/(5*43)
weight.5 <- total.weight/(5*18)

#increase the weight of class 4 & 5 by a little bit over 0.4(chosen arbitraily)
weight.4 <- 1.5
weight.5 <- 3

linear.weighted <- tune(svm, site~female+ male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+s
                    data=train.data, kernel="linear",
                    ranges=list(cost=costs),
                    class.weights=c("1"=weight.1,
                                    "2"=weight.2,
                                    "3"=weight.3,
                                    "4"=weight.4,
```

```
                                    "5"=weight.5),
                    class.type="one.versus.one")

radial.weighted <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+
                    data=train.data, kernel="radial",
                    ranges=list(cost=costs,
                                gamma=gammas),
                    class.weights=c("1"=weight.1,
                                    "2"=weight.2,
                                    "3"=weight.3,
                                    "4"=weight.4,
                                    "5"=weight.5),
                    class.type="one.versus.one")
#radial.tune <- tune(svm, site~sex+age+prebody+prelet+preform+prenumb+prerelat+preclasf,
#                     data=train.data, kernel="radial",
#                     ranges=list(cost=costs,
#                                 gamma=gammas))

sigmoid.weighted <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat
                    data=train.data, kernel="sigmoid",
                    ranges=list(cost=costs,
                                gamma=gammas),
                    class.weights=c("1"=weight.1,
                                    "2"=weight.2,
                                    "3"=weight.3,
                                    "4"=weight.4,
                                    "5"=weight.5),
                    class.type="one.versus.one")

poly.weighted <- tune(svm, site~female + male + sd_age+sd_pBod+sd_plet+sd_pform + sd_pnumb+sd_prelat+sd
                    data=train.data, kernel="sigmoid",
                    ranges=list(cost=costs,
                                degree=degrees),
                    class.weights=c("1"=weight.1,
                                    "2"=weight.2,
                                    "3"=weight.3,
                                    "4"=weight.4,
                                    "5"=weight.5),
                    class.type="one.versus.one")
```

```
linear.w.cm <- table(true=test.data[, "site"],
                          pred=predict(linear.weighted$best.model, newdata=test.data))

radial.w.cm <- table(true=test.data[, "site"],
                          pred=predict(radial.weighted$best.model, newdata=test.data))

sigmoid.w.cm <- table(true=test.data[,"site"],
                    pred=predict(sigmoid.weighted$best.model, newdata=test.data))

poly.w.cm <- table(true=test.data[, "site"],
                  pred=predict(poly.weighted$best.model, newdata=test.data))
```

```
confusionMatrix(linear.w.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  6  1  6  2  5
##    2  1  7  1  1  3
##    3  0  1 11  1  3
##    4  3  3  9  3  0
##    5  0  1  1  1  2
##
## Overall Statistics
##
##                Accuracy : 0.4028
##                  95% CI : (0.2888, 0.525)
##     No Information Rate : 0.3889
##     P-Value [Acc > NIR] : 0.44844
##
##                   Kappa : 0.2554
##
##  Mcnemar's Test P-Value : 0.01728
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.60000  0.53846   0.3929  0.37500  0.15385
## Specificity           0.77419  0.89831   0.8864  0.76562  0.94915
## Pos Pred Value        0.30000  0.53846   0.6875  0.16667  0.40000
## Neg Pred Value        0.92308  0.89831   0.6964  0.90741  0.83582
## Prevalence            0.13889  0.18056   0.3889  0.11111  0.18056
## Detection Rate        0.08333  0.09722   0.1528  0.04167  0.02778
## Detection Prevalence  0.27778  0.18056   0.2222  0.25000  0.06944
## Balanced Accuracy     0.68710  0.71838   0.6396  0.57031  0.55150
```

```
confusionMatrix(radial.w.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  6  2  7  4  1
##    2  4  4  3  0  2
##    3  1  1 11  3  0
##    4  2  2  9  4  1
##    5  0  2  1  1  1
##
## Overall Statistics
##
##                Accuracy : 0.3611
##                  95% CI : (0.2512, 0.4829)
##     No Information Rate : 0.4306
##     P-Value [Acc > NIR] : 0.9056
##
##                   Kappa : 0.181
##
##  Mcnemar's Test P-Value : 0.1807
```

```
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.46154  0.36364   0.3548  0.33333  0.20000
## Specificity           0.76271  0.85246   0.8780  0.76667  0.94030
## Pos Pred Value        0.30000  0.30769   0.6875  0.22222  0.20000
## Neg Pred Value        0.86538  0.88136   0.6429  0.85185  0.94030
## Prevalence            0.18056  0.15278   0.4306  0.16667  0.06944
## Detection Rate        0.08333  0.05556   0.1528  0.05556  0.01389
## Detection Prevalence  0.27778  0.18056   0.2222  0.25000  0.06944
## Balanced Accuracy     0.61213  0.60805   0.6164  0.55000  0.57015
```

```
confusionMatrix(sigmoid.w.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  1  1  6  9  3
##    2  1  3  4  5  0
##    3  1  0  4 11  0
##    4  0  0  7 11  0
##    5  0  0  3  2  0
##
## Overall Statistics
##
##                Accuracy : 0.2639
##                  95% CI : (0.167, 0.381)
##     No Information Rate : 0.5278
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0434
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.33333  0.75000  0.16667   0.2895  0.00000
## Specificity           0.72464  0.85294  0.75000   0.7941  0.92754
## Pos Pred Value        0.05000  0.23077  0.25000   0.6111  0.00000
## Neg Pred Value        0.96154  0.98305  0.64286   0.5000  0.95522
## Prevalence            0.04167  0.05556  0.33333   0.5278  0.04167
## Detection Rate        0.01389  0.04167  0.05556   0.1528  0.00000
## Detection Prevalence  0.27778  0.18056  0.22222   0.2500  0.06944
## Balanced Accuracy     0.52899  0.80147  0.45833   0.5418  0.46377
```

```
confusionMatrix(poly.w.cm)
```

```
## Confusion Matrix and Statistics
##
##     pred
## true  1  2  3  4  5
##    1  6  2  5  1  6
```

```
##    2  4  2  3  0  4
##    3  1  0 11  1  3
##    4  3  2  8  1  4
##    5  1  1  2  0  1
##
## Overall Statistics
##
##                Accuracy : 0.2917
##                  95% CI : (0.1905, 0.4107)
##     No Information Rate : 0.4028
##     P-Value [Acc > NIR] : 0.981095
##
##                   Kappa : 0.1226
##
##  Mcnemar's Test P-Value : 0.006726
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity           0.40000  0.28571   0.3793  0.33333  0.05556
## Specificity           0.75439  0.83077   0.8837  0.75362  0.92593
## Pos Pred Value        0.30000  0.15385   0.6875  0.05556  0.20000
## Neg Pred Value        0.82692  0.91525   0.6786  0.96296  0.74627
## Prevalence            0.20833  0.09722   0.4028  0.04167  0.25000
## Detection Rate        0.08333  0.02778   0.1528  0.01389  0.01389
## Detection Prevalence  0.27778  0.18056   0.2222  0.25000  0.06944
## Balanced Accuracy     0.57719  0.55824   0.6315  0.54348  0.49074
```

In our initial SVM models, Radial Kernel SVM came with the best performance in terms of prediction accuracy, but it was only slightly above 0.30. We came across several online resources that stated standardizing the variables and encoding the categorical variables have empirically imrpoved SVM models performance (https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf). After we standardized the variables and encoded the sex variable (if a data is a male, then it has (1,0)), we observed a light improvement in prediction accuracy across all models. However, one problem that particularly piqued our interests is that no predictions on class 4 class 5 are made before & after variable transformation. We then realized that, relatively, class 4 & 5 have less number of data than the other classes. Although we'd argue that the proportion not severe enough for us to deem the dataset as an imbalanced one, we used the formula below to assign weights to each class and specify "one versus one" comparison, which has been suggested to yield better prediction than "one versus all."

$w_j = \frac{n}{kn_j}$, n is total number of data points, k is number of classes, n_j is the number of data in class j

After class weight assignment, the SVM models began to make prediction on class 4 & 5, at the cost of overall accuracy that more test data are being misclassified even though a few number of 4&5 are correctly classified. Then, we began to experiment with the class weights and increase the class 4 & 5 weights by roughly 0.5, which is arbitrarily chosen, and it boosted linear kernel SVM's prediction accuracy to 0.403.