Classification

Aloksai Choudari, Alekhya Pinnamaneni

2022-10-19

Select a data set

Data set: Avila Data Set

Source: https://archive.ics.uci.edu/ml/datasets/Avila

Attributes: intercolumnar_dist, upper_margin, lower_margin, exploitation, row_no, modular_ratio, inter-linear_spacing, weight, peak_no, ratio_spacing, copyist

No. of instances (rows): 10,430 rows

Load the data

```
avila <- read.csv("avila-tr.txt", header = FALSE)
```

Clean the data

```
# Add column names to the data set
colnames(avila) <- c("intercolumnar_dist", "upper_margin", "lower_margin", "exploitation", "row_no", "m

# Factor the copyist column
avila$copyist <- as.factor(avila$copyist)
```

Divide data into train, test, and validate sets

```
set.seed(1234)
spec <- c(train=.7, test=.2, validate=.1)
i <- sample(cut(1:nrow(avila),
                nrow(avila)*cumsum(c(0,spec)), labels=names(spec)))
train <- avila[i=="train",]
test <- avila[i=="test",]
val <- avila[i=="validate",]
```

Explore the data statistically and graphically

```r
# Outputs the first 5 rows of the train data
head(train, n=5)
```

```
##   intercolumnar_dist upper_margin lower_margin exploitation   row_no
## 3          -0.116585     0.069915     0.068476    -0.783147 0.261718
## 5           0.229043     0.807926    -0.052442     0.082634 0.261718
## 7           0.389513    -0.220579    -3.210528    -2.624155 0.261718
## 8           0.019197    -0.040001     0.288973    -0.042597 0.261718
## 9           0.500607     0.140576     0.388552    -0.637358 0.261718
##   modular_ratio interlinear_spacing    weight   peak_no ratio_spacing copyist
## 3      0.439463           -0.081827 -0.888236 -0.123005      0.582939       A
## 5      0.148790            0.635431  0.051062  0.032902     -0.086652       F
## 7     -0.764757            0.484429 -0.597510 -0.372457     -0.810261       A
## 8     -1.013906            0.069175  0.890701  0.095265     -0.842014       F
## 9     -0.681707            0.295677  0.931046  0.500624     -0.642297       H
```

```r
# Outputs the mean intercolumnar distance
mean(train$intercolumnar_dist)
```

```
## [1] -0.002122084
```

```r
# Outputs the lowest and highest upper margin
range(train$upper_margin)
```

```
## [1] -2.426761 43.133656
```

```r
# Outputs the most common class for the copyist column
names(which.max(table(train$copyist)))
```

```
## [1] "A"
```

```r
# Outputs the median weight
median(train$weight)
```

```
## [1] 0.1174725
```

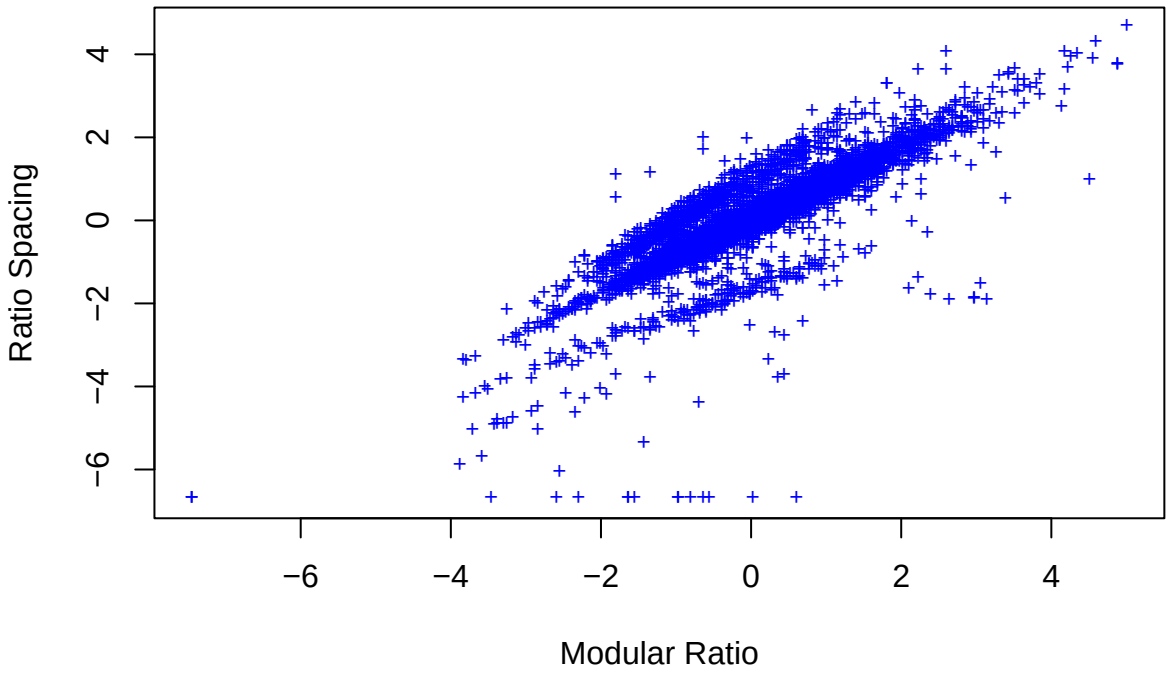
```r
# Outputs the mean exploitation
mean(train$exploitation)
```

```
## [1] 0.002016289
```

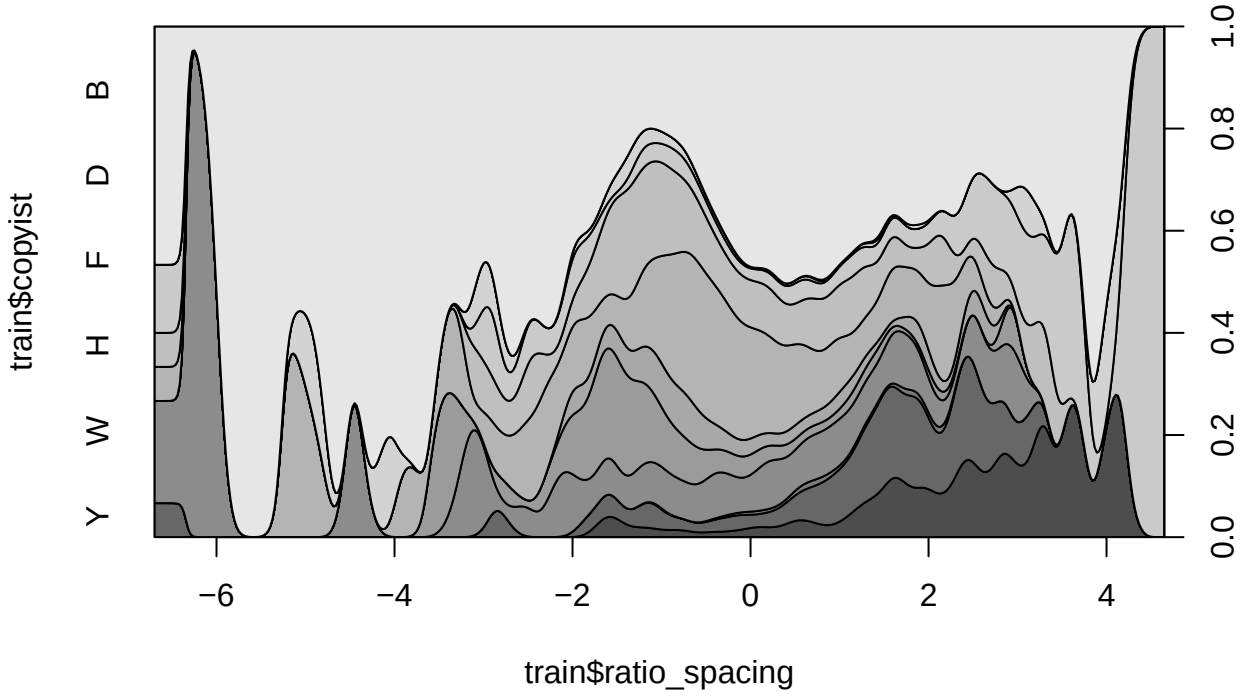
```r
# Outputs statistics for ratio spacing
summary(train$ratio_spacing)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -6.719324 -0.514624 -0.023164  0.002884  0.535815  4.671232
```

```
# Outputs a scatterplot of modular ratio vs. ratio spacing
plot(train$modular_ratio, train$ratio_spacing, pch='+', cex=0.75, col="blue", xlab="Modular Ratio", yla
```




```
# Outputs a conditional density plot showing how the copyist changes over the various ratio spacings
cdplot(train$ratio_spacing, train$copyist)
```



SVM Classification

Linear Kernel

```
library(e1071)
library(MASS)
svm1 <- tune(svm, copyist~., data=val, kernel="linear", ranges=list(cost=c(0.1, 1, 10)))
summary(svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.4206044
##
## - Detailed performance results:
##   cost     error dispersion
## 1  0.1 0.4637179 0.04303003
```

```
## 2  1.0 0.4273260 0.03712678
## 3 10.0 0.4206044 0.03810698
```

```
pred1 <- predict(svm1$best.model, newdata=test)
acc1 <- mean(pred1==test$copyist)
print(paste("accuracy = ", acc1))
```

```
## [1] "accuracy =  0.552732502396932"
```

Polynomial Kernel

```
svm2 <- tune(svm, copyist~., data=val, kernel="polynomial", ranges=list(cost=c(10, 100, 150)))
summary(svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   150
##
## - best performance: 0.3993498
##
## - Detailed performance results:
##   cost     error dispersion
## 1   10 0.4309799 0.03914545
## 2  100 0.4079670 0.03595662
## 3  150 0.3993498 0.04610010
```

```
pred2 <- predict(svm2$best.model, newdata=test)
acc2 <- mean(pred2==test$copyist)
print(paste("accuracy = ", acc2))
```

```
## [1] "accuracy =  0.585810162991371"
```

Radial Kernel

```
svm3 <- tune(svm, copyist~., data=val, kernel="radial", ranges=list(cost=c(1, 10, 100), gamma=c(0.1, 0.1
summary(svm3)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
```

```
##  cost gamma
##   100   0.1
##
## - best performance: 0.3275549
##
## - Detailed performance results:
##   cost gamma     error dispersion
## 1    1  0.10 0.3849908 0.06483067
## 2   10  0.10 0.3399817 0.03704049
## 3  100  0.10 0.3275549 0.03023684
## 4    1  0.25 0.3638828 0.04905775
## 5   10  0.25 0.3275916 0.03533776
## 6  100  0.25 0.3391026 0.03526382
## 7    1  0.50 0.3648260 0.04977931
## 8   10  0.50 0.3371429 0.04835418
## 9  100  0.50 0.3554396 0.04261253
```

```
pred3 <- predict(svm3$best.model, newdata=test)
acc3 <- mean(pred3==test$copyist)
print(paste("accuracy = ", acc3))
```

```
## [1] "accuracy =  0.664908916586769"
```

Analysis

The linear kernel had the worst performance out of the three kernels with an accuracy of 0.5527, when using C=10 (cost hyperparameter/slack). When tuning for the most optimal cost hyperparameter, we discovered that 10 generated the highest correlation because lower slack values prevent underfitting and reduce bias. The linear SVM kernel was not suitable for this dataset because the classes are not linearly separable. This explains why the accuracy was so low for the linear kernel algorithm. The polynomial kernel had a slightly improved performance with an accuracy of 0.5858, when C=150. A slack of 150 produced the highest accuracy for this kernel because larger slack values reduce overfitting and variance. The polynomial kernel algorithm was more accurate because the polynomial kernel maps the observations to a higher dimensional space. The radial kernel had the best performance with 0.6649 for accuracy, when C=100 and gamma=0.1. The additional hyperparameter, gamma, controls the bias-variance tradeoff, which allows for even higher accuracy when tuned to find the best value.