

# BSc Coursework 2

Salik Tariq / Student ID: 12516369

## 1) Bayesian Networks and Naïve Bayes Classifiers

(a) Given a training dataset including 30 instances and a Bayesian network indicating the relationships between 3 features (i.e. Income, Student and Credit Rate), and the class attribute (i.e. Buy Computer), please create the conditional probability tables by hand.

**Q1(a) and (c) completed by hand in a separate file CW2-R-Q1.pdf**

(b) Make predictions for 2 testing instances by using the Bayesian network classifier

Prediction for Instance\_31 Income = Low Student = False Credit Rating = Excellent

To predict: Buy Computer?

$P(\text{Buy Computer}=\text{Yes}, \text{Income} = \text{Low}, \text{Student}=\text{False}, \text{Credit Rating} = \text{Excellent}) = P(\text{Income}=\text{Low} \mid \text{Buy Computer} = \text{Yes}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{Yes}) P(\text{Credit Rating} = \text{Excellent} \mid \text{Income} = \text{Low}, \text{Student} = \text{False}, \text{Buy Computer} = \text{Yes}) * P(\text{Buy Computer} = \text{Yes})$

$$= 0.643 * 0.5 * 0.5 * 0.467 = 0.075$$

$P(\text{Buy Computer}=\text{No}, \text{Income} = \text{Low}, \text{Student}=\text{False}, \text{Credit Rating} = \text{Excellent}) = P(\text{Income}=\text{Low} \mid \text{Buy Computer} = \text{No}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{No}) P(\text{Credit Rating} = \text{Excellent} \mid \text{Income} = \text{Low}, \text{Student} = \text{False}, \text{Buy Computer} = \text{No}) * P(\text{Buy Computer} = \text{No})$

$$= 0.5625 * 0.3125 * 0.5 * 0.533 = 0.0468$$

**As the probability of Buy Computer = Yes is greater than Buy Computer = No (0.075>0.0468)**

**Buy Computer = Yes for Instance\_31**

Prediction for Instance\_32 Income = High Student = False Credit Rating = Fair

To predict: Buy Computer?

$P(\text{Buy Computer} = \text{Yes}, \text{Income} = \text{High}, \text{Student} = \text{False}, \text{Credit Rating} = \text{Fair}) = P(\text{Income} = \text{High} \mid \text{Buy Computer} = \text{Yes}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{Yes}) P(\text{Credit Rating} = \text{Fair} \mid \text{Buy Computer} = \text{Yes}, \text{Student} = \text{False}, \text{Income} = \text{High}) * P(\text{Buy Computer} = \text{Yes})$

$$= 0.357 * 0.5 * 0.334 * 0.467 = 0.0278$$

$P(\text{Buy Computer} = \text{No}, \text{Income} = \text{High}, \text{Student} = \text{False}, \text{Credit Rating} = \text{Fair}) = P(\text{Income} = \text{High} \mid \text{Buy Computer} = \text{No}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{No}) P(\text{Credit Rating} = \text{Fair} \mid \text{Buy Computer} = \text{No}, \text{Student} = \text{False}, \text{Income} = \text{High}) * P(\text{Buy Computer} = \text{No})$

$$= 0.4375 * 0.3125 * 0.334 * 0.533 = 0.02434$$

**As the probability of Buy Computer = Yes is greater than Buy Computer = No (0.0278>0.02434)**

**Buy Computer = Yes for Instance\_32**

(c) Based on the conditional independence assumption between features, please create the conditional probability tables by hand.

**Q1(a) and (c) completed by hand in a separate file CW2-R-Q1.pdf**

(d) Make predictions for 2 testing instances by using the naïve Bayes classifier

Prediction for Instance\_31 Income = Low Student = False Credit Rating = Excellent

To predict: Buy Computer?

$P(\text{Buy Computer}=\text{Yes}, \text{Income} = \text{Low}, \text{Student}=\text{False}, \text{Credit Rating} = \text{Excellent}) = P(\text{Income}=\text{Low} \mid \text{Buy Computer} = \text{Yes}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{Yes}) P(\text{Credit Rating} = \text{Excellent} \mid \text{Buy Computer} = \text{Yes}) * P(\text{Buy Computer} = \text{Yes})$

$= 0.643 * 0.5 * 0.5 * 0.467 = 0.075$

$P(\text{Buy Computer}=\text{No}, \text{Income} = \text{Low}, \text{Student}=\text{False}, \text{Credit Rating} = \text{Excellent}) = P(\text{Income}=\text{Low} \mid \text{Buy Computer} = \text{No}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{No}) P(\text{Credit Rating} = \text{Excellent} \mid \text{Buy Computer} = \text{No}) * P(\text{Buy Computer} = \text{No})$

$= 0.5625 * 0.3125 * 0.4375 * 0.533 = 0.041$

**As the probability of Buy Computer = Yes is greater than Buy Computer = No (0.075>0.041)**

**Buy Computer = Yes for Instance\_31**

Prediction for Instance\_32 Income = High Student = False Credit Rating = Fair

To predict: Buy Computer?

$P(\text{Buy Computer} = \text{Yes}, \text{Income} = \text{High}, \text{Student} = \text{False}, \text{Credit Rating} = \text{Fair}) = P(\text{Income} = \text{High} \mid \text{Buy Computer} = \text{Yes}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{Yes}) P(\text{Credit Rating} = \text{Fair} \mid \text{Buy Computer} = \text{Yes}) * P(\text{Buy Computer} = \text{Yes})$

$= 0.357 * 0.5 * 0.5 * 0.467 = 0.04167$

$P(\text{Buy Computer} = \text{No}, \text{Income} = \text{High}, \text{Student} = \text{False}, \text{Credit Rating} = \text{Fair}) = P(\text{Income} = \text{High} \mid \text{Buy Computer} = \text{No}) P(\text{Student} = \text{False} \mid \text{Buy Computer} = \text{No}) P(\text{Credit Rating} = \text{Fair} \mid \text{Buy Computer} = \text{No}) * P(\text{Buy Computer} = \text{No})$

$= 0.4375 * 0.3125 * 0.5625 * 0.533 = 0.0410$

**As the probability of Buy Computer = Yes is greater than Buy Computer = No (0.04167>0.0410)**

**Buy Computer = Yes for Instance\_32**

## 2) Decision Trees and Random Forests

To predict room occupancy using the decision tree classification algorithm.

(a) Load the room occupancy data and train a decision tree classifier. Evaluate the predictive performance by reporting the accuracy obtained on the testing dataset.

```
library("rpart")
library("rpart.plot")
```

```
## Warning: package 'rpart.plot' was built under R version 3.6.2
```

```

library("randomForest")

## Warning: package 'randomForest' was built under R version 3.6.2
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.

library("gplots")

##
## Attaching package: 'gplots'
## The following object is masked from 'package:stats':
##
##      lowess

library("ROCR")
library("pROC")

## Warning: package 'pROC' was built under R version 3.6.2
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
## The following objects are masked from 'package:stats':
##
##      cov, smooth, var

set.seed(300)
data_train <- read.csv(file="RoomOccupancy_Training.txt", header=TRUE, sep=",")
data_test  <- read.csv(file="RoomOccupancy_Testing.txt", header=TRUE, sep=",")
#Exploring train DataSet
head(data_train)

##      Temperature Humidity Light      CO2 HumidityRatio Occupancy
## 1      23.18    27.2720 426.0 721.25   0.004792988      Yes
## 2      23.15    27.2675 429.5 714.00   0.004783441      Yes
## 3      23.15    27.2450 426.0 713.50   0.004779464      Yes
## 4      23.15    27.2000 426.0 708.25   0.004771509      Yes
## 5      23.10    27.2000 426.0 704.50   0.004756993      Yes
## 6      23.10    27.2000 419.0 701.00   0.004756993      Yes

#printing all columns with their data type
str(data_train)

## 'data.frame':    2000 obs. of  6 variables:
## $ Temperature : num  23.2 23.1 23.1 23.1 23.1 ...
## $ Humidity     : num  27.3 27.3 27.2 27.2 27.2 ...
## $ Light        : num  426 430 426 426 426 ...
## $ CO2          : num  721 714 714 708 704 ...
## $ HumidityRatio: num  0.00479 0.00478 0.00478 0.00477 0.00476 ...
## $ Occupancy    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 ...

#Checking Null values
any(is.na(data_train))

## [1] FALSE

```

```
#Training Decision Tree Model
train_tree <- rpart(Occupancy ~ ., method = "class", data = data_train)
```

```
#Evaluate the predictive performance
```

```
tree.preds <- predict(train_tree, data_test)
print(head(tree.preds))
```

```
##           No           Yes
## 1 0.01010101 0.9898990
## 2 0.01010101 0.9898990
## 3 0.01010101 0.9898990
## 4 0.03007519 0.9699248
## 5 0.03007519 0.9699248
## 6 0.03007519 0.9699248
```

```
tree_pred <- as.data.frame(tree.preds)
```

```
prob <- function(a){
  if(a>=0.5){
    return('Yes')
  }else{
    return('No')
  }
}
```

```
tree_pred$Occupancy <- sapply(tree_pred$Yes, prob)
```

```
print(head(tree_pred))
```

```
##           No           Yes Occupancy
## 1 0.01010101 0.9898990           Yes
## 2 0.01010101 0.9898990           Yes
## 3 0.01010101 0.9898990           Yes
## 4 0.03007519 0.9699248           Yes
## 5 0.03007519 0.9699248           Yes
## 6 0.03007519 0.9699248           Yes
```

```
#reporting the accuracy obtained on the testing dataset
```

```
#confusion matrix
```

```
table_mat <- table(tree_pred$Occupancy, data_test$Occupancy)
print(table_mat)
```

```
##
##           No Yes
##  No  179  15
##  Yes  61  45
```

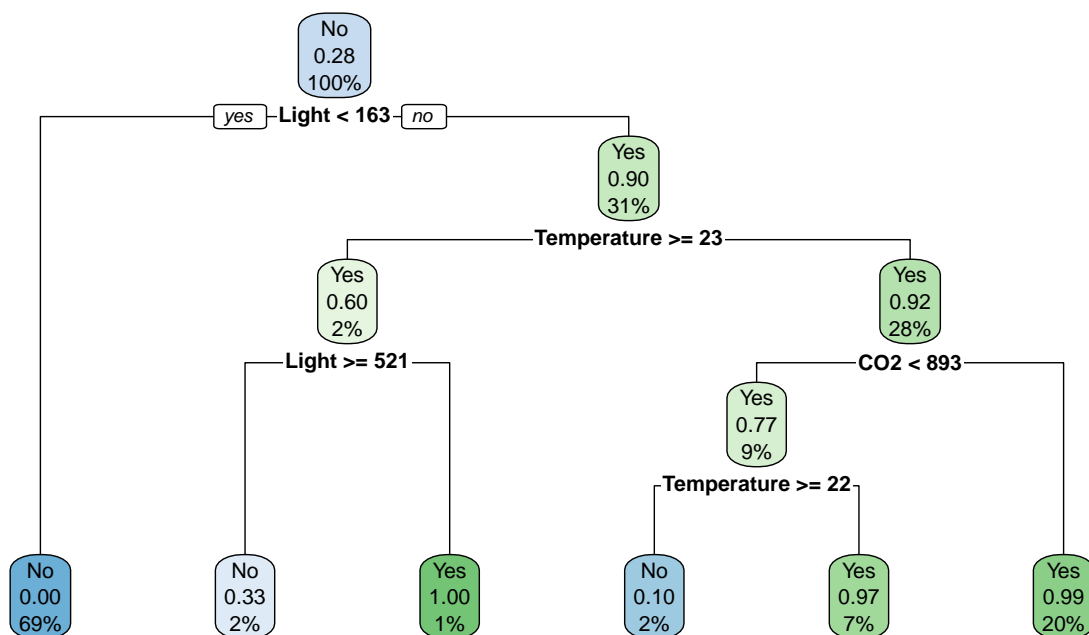
```
accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy for test', accuracy_Test))
```

```
## [1] "Accuracy for test 0.746666666666667"
```

(b) Output and analyse the tree learned by the decision tree algorithm, i.e. plot the tree structure and make a discussion about it.

```
library("rpart")
library("rpart.plot")
library("randomForest")
d_tree <- rpart(Occupancy ~ ., method = 'class', data = data_train)
#Output and analyse and plotting a tree
rpart.plot(d_tree, uniform = T, main = 'Occupancy Tree')
```

## Occupancy Tree



*#interpretation of decision tree*

```
rpart(formula = Occupancy ~ ., data = data_train, method = "class")
```

```
## n= 2000
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 2000 555 No (0.72250000 0.27750000)
##    2) Light< 162.875 1381 0 No (1.00000000 0.00000000) *
##    3) Light>=162.875 619 64 Yes (0.10339257 0.89660743)
##      6) Temperature>=22.64167 50 20 Yes (0.40000000 0.60000000)
##        12) Light>=520.5 30 10 No (0.66666667 0.33333333) *
##        13) Light< 520.5 20 0 Yes (0.00000000 1.00000000) *
##        7) Temperature< 22.64167 569 44 Yes (0.07732865 0.92267135)
##          14) CO2< 893.125 173 40 Yes (0.23121387 0.76878613)
##            28) Temperature>=22.21125 40 4 No (0.90000000 0.10000000) *
```

```
##          29) Temperature< 22.21125 133    4 Yes (0.03007519 0.96992481) *
##          15) CO2>=893.125 396    4 Yes (0.01010101 0.98989899) *
```

Error rate is small, hence pruning is not required.

(c) Train a random forests classifier, and evaluate the predictive performance by reporting the accuracy obtained on the testing dataset.

```
rf_model <- randomForest(Occupancy ~., data = data_train , importance = TRUE)
print(rf_model)

##
## Call:
## randomForest(formula = Occupancy ~ ., data = data_train, importance = TRUE)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 2
##
## OOB estimate of  error rate: 1.3%
## Confusion matrix:
##      No Yes class.error
## No  1430  15  0.01038062
## Yes   11 544  0.01981982

#evaluate the predictive performance
#obtained on the testing dataset
rf_pred <- predict(rf_model,data_test)
rf_mat <- table(rf_pred,data_test$Occupancy)
print(rf_mat)

##
## rf_pred  No Yes
##      No  176   6
##      Yes   64  54

#reporting the accuracy
accuracy_Test <- sum(diag(rf_mat)) / sum(rf_mat)
print(paste('Accuracy for test', accuracy_Test))

## [1] "Accuracy for test 0.766666666666667"
```

Accuracy of 0.7666 is less accurate than a single tree.

(d) Output and analyse the feature importance obtained by the random forests classifier.

```
#Feature Importance
rf_model$importance

##              No              Yes MeanDecreaseAccuracy MeanDecreaseGini
## Temperature  0.03376326 0.07969206          0.04645670        114.56211
## Humidity     0.01344981 0.08876716          0.03439724         29.43368
## Light        0.16822273 0.56439203          0.27797941        403.06189
## CO2          0.03830877 0.24704547          0.09617212        205.31916
## HumidityRatio 0.01623301 0.07209788          0.03177614         47.47406
```

From the output we can determine that 'Light' is greatest factor in determining the occupancy.

### 3) SVM

To predict the wine quality using the support vector machine classification algorithm.

(a) Download the wine quality data and use the training dataset to conduct the grid-search to find the optimal hyperparameters of svm by using the linear kernel.

```
set.seed(300)
data_train <- read.csv(file="WineQuality_training.txt", header=TRUE, sep=",")
data_test <- read.csv(file="WineQuality_testing.txt", header=TRUE, sep=",")
#Exploring train DataSet
#head(data_train)
#printing all columns with their data type
#str(data_train)
#Checking Null values
#any(is.na(data_train))

library("e1071")

## Warning: package 'e1071' was built under R version 3.6.2
model <- svm(quality ~., data = data_train, kernel = 'linear')
#summary(model)

#Grid Search using Linear Kernel and finding optimal hyperparameter
h_tune <- tune(svm, train.x = data_train[1:11] , train.y = data_train[,12] ,
              kernel = 'linear' ,
              ranges = list(cost = c(0.01, 0.1, 1, 5, 10)))
print(paste("Optimal Cost: ", h_tune$best.parameters[1]))

## [1] "Optimal Cost: 0.1"
#summary(h_tune)
```

(b) Train a svm classifier by using the linear kernel and the corresponding optimal hyperparameters, then make predictions on the testing dataset, report the predictive performance.

```
model_lin <- svm(quality ~., data = data_train, kernel = 'linear'
               , cost = 0.1)

#make predictions on the testing dataset,
#report the predictive performance

model_pred <- predict(model, data_test)

#reporting the accuracy
svm_mat <- table(model_pred, data_test$quality)
print(svm_mat)

##
## model_pred Bad Good
```

```
##      Bad 104   89
##      Good 38 169

accuracy_Test <- sum(diag(svm_mat)) / sum(svm_mat)
print(paste('Accuracy for test', accuracy_Test))
```

```
## [1] "Accuracy for test 0.6825"
```

(c) Conduct the grid-search to find the optimal hyperparameters of svm by using the RBF kernel.

```
#Training SVM using RBF
model <- svm(quality ~., data = data_train, kernel = 'radial')

#Grid Search using RBF Kernel and finding optimal hyperparameter
hrbf_tune <- tune(svm, train.x = data_train[1:11] , train.y = data_train[,12] ,
                 kernel = 'radial' ,
                 ranges = list(cost = c(0.01, 0.1, 1, 5, 10), gamma = c(0.01, 0.03, 0.1, 0.5, 1)))
summary(hrbf_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost gamma
##     5    0.5
##
## - best performance: 0.159
##
## - Detailed performance results:
##   cost gamma   error dispersion
## 1  0.01 0.01 0.2886667 0.02079886
## 2  0.10 0.01 0.2520000 0.01853925
## 3  1.00 0.01 0.2363333 0.01842234
## 4  5.00 0.01 0.2076667 0.01785055
## 5 10.00 0.01 0.2070000 0.01702939
## 6  0.01 0.03 0.2673333 0.01698220
## 7  0.10 0.03 0.2400000 0.02479546
## 8  1.00 0.03 0.2063333 0.02151371
## 9  5.00 0.03 0.1956667 0.02024846
## 10 10.00 0.03 0.1966667 0.01706921
## 11 0.01 0.10 0.2676667 0.02378141
## 12 0.10 0.10 0.2073333 0.01824389
## 13 1.00 0.10 0.1930000 0.01828782
## 14 5.00 0.10 0.1806667 0.01553967
## 15 10.00 0.10 0.1750000 0.01649916
## 16 0.01 0.50 0.5126667 0.07542145
## 17 0.10 0.50 0.2323333 0.02336242
## 18 1.00 0.50 0.1656667 0.01625795
## 19 5.00 0.50 0.1590000 0.01735967
## 20 10.00 0.50 0.1613333 0.02440401
## 21 0.01 1.00 0.5153333 0.06737036
## 22 0.10 1.00 0.2706667 0.05941692
```



```
## 23 1.00 1.00 0.1673333 0.01748368
## 24 5.00 1.00 0.1613333 0.01813529
## 25 10.00 1.00 0.1623333 0.01918397
```

```
model_rbf <- svm(quality ~., data = data_train, kernel = 'radial'
, cost =5 , gamma = 0.5 ,decision.values = TRUE,probability = TRUE)
```

```
model_predrbf <- predict(model_rbf,data_test)
```

```
print(model_predrbf)
```

```
##      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15
## Good Good Bad Bad Good Good Good Bad Good Good Good Good Bad Good Bad
## 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
## Good Good Good Good Bad Good Good Good Good Good Good Good Bad Bad Bad
## 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
## Bad Good Bad Bad Good Good Good Good Good Good Good Good Bad Bad Bad
## 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
## Bad Bad Bad Good Bad Bad Bad Bad Good Good Bad Bad Good Good Good
## 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
## Good Good Good Good Good Good Good Good Good Bad Good Good Good Good Good
## 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
## Good Good Good Bad Bad Good Bad Bad Good Good Good Good Good Bad Bad
## 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105
## Good Bad Good Bad Good Bad Bad Bad Good Good Bad Good Good Bad Bad
## 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## Bad Good Good Bad Good Good Bad Good Good Bad Good Bad Good Good Bad
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135
## Good Good Good Good Good Good Bad Good Bad Bad Good Bad Good Good Bad
## 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150
## Bad Bad Bad Bad Good Bad Bad Bad Bad Good Good Bad Good Good Good
## 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
## Good Bad Good Bad Bad Bad Good Bad Bad Good Good Bad Bad Good Good
## 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## Good Bad Good Good Good Good Good Good Good Bad Good Bad Bad Good Good
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
## Bad Good Bad Bad Good Good Bad Bad Bad Bad Bad Bad Good Bad Bad
## 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
## Bad Good Good Bad Bad Bad Good Bad Bad Bad Good Bad Bad Good Bad
## 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225
## Good Good Good Good Good Good Good Good Bad Bad Good Good Good Good Bad
## 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
## Good Bad Good Bad Bad Good Good Good Bad Good Bad Good Good Good Good
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
## Bad Good Bad Good Good Good Good Bad Bad Bad Bad Bad Bad Good Good
## 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## Good Good Good Bad Bad Bad Bad Good Bad Bad Bad Bad Bad Bad Bad
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
## Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad
## 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
## Bad Bad Good Bad Bad Bad Bad Bad Bad Bad Good Good Bad Bad Good
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
## Good Bad Bad Good Bad Bad Bad Bad Bad Bad Bad Bad Good Bad Bad
## 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
## Good Bad Good Bad Bad Bad Bad Good Good Bad Bad Bad Bad Bad Bad
```

```
## 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345
## Good Good Bad Bad Good Bad Good Bad Bad Good Good Bad Bad Bad Bad
## 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
## Bad Bad Bad Good Good Bad Bad Bad Bad Bad Bad Bad Bad Bad
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375
## Good Bad Bad Bad Good Good Bad Good Bad Good Bad Bad Bad Bad
## 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390
## Bad Good Bad Good Bad Bad Bad Bad Bad Bad Bad Bad Bad Bad
## 391 392 393 394 395 396 397 398 399 400
## Bad Bad Bad Bad Bad Good Good Bad Bad Bad
## Levels: Bad Good
```

(d) Train a svm classifier by using the RBF kernel and the corresponding optimal hyperparameters, then make predictions on the testing dataset, report the predictive performance.

```
svm.rbf_mat <- table(model_predrbf,data_test$quality)
print(svm.rbf_mat)

##
## model_predrbf Bad Good
##           Bad 113 108
##           Good 29 150

accuracy_Test <- sum(diag(svm.rbf_mat)) / sum(svm.rbf_mat)
print(paste('Accuracy for test', accuracy_Test))

## [1] "Accuracy for test 0.6575"
```

(e) Conduct the ROC curve analysis to compare the predictive performance of svm classifiers trained by using the linear and RBF kernels respectively.

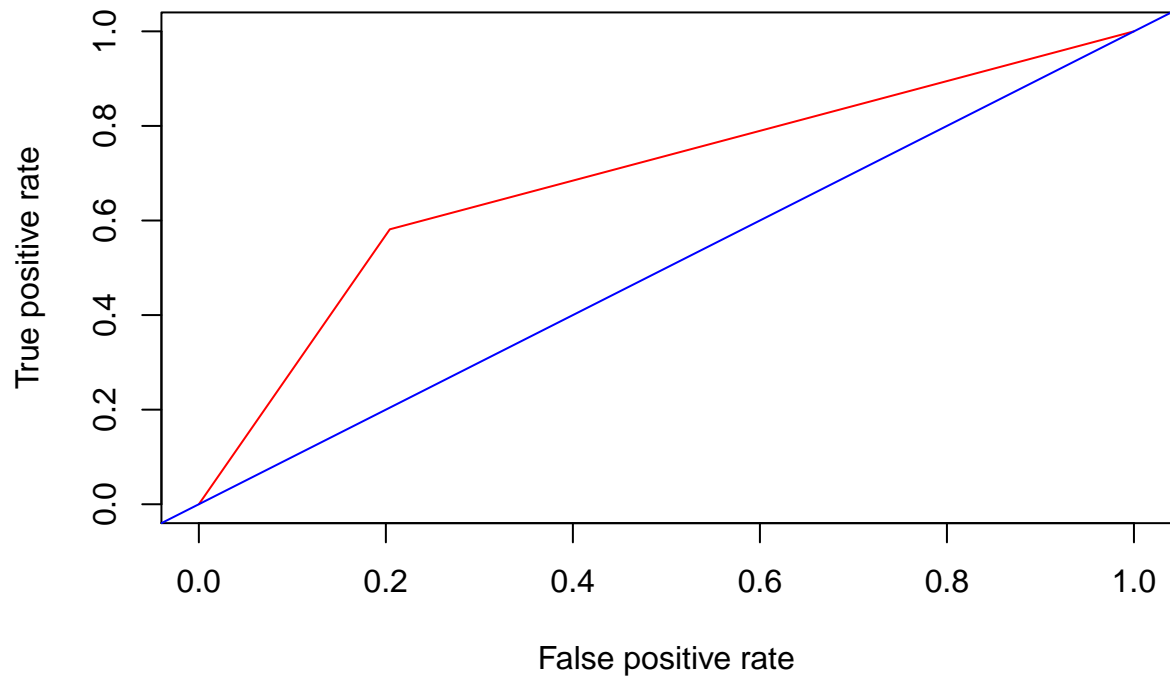
```
library("ROSE")

## Warning: package 'ROSE' was built under R version 3.6.2
## Loaded ROSE 0.0-3

rbf_pred <- predict(model_rbf,data_test)
lin_pred <- predict(model_lin,data_test)

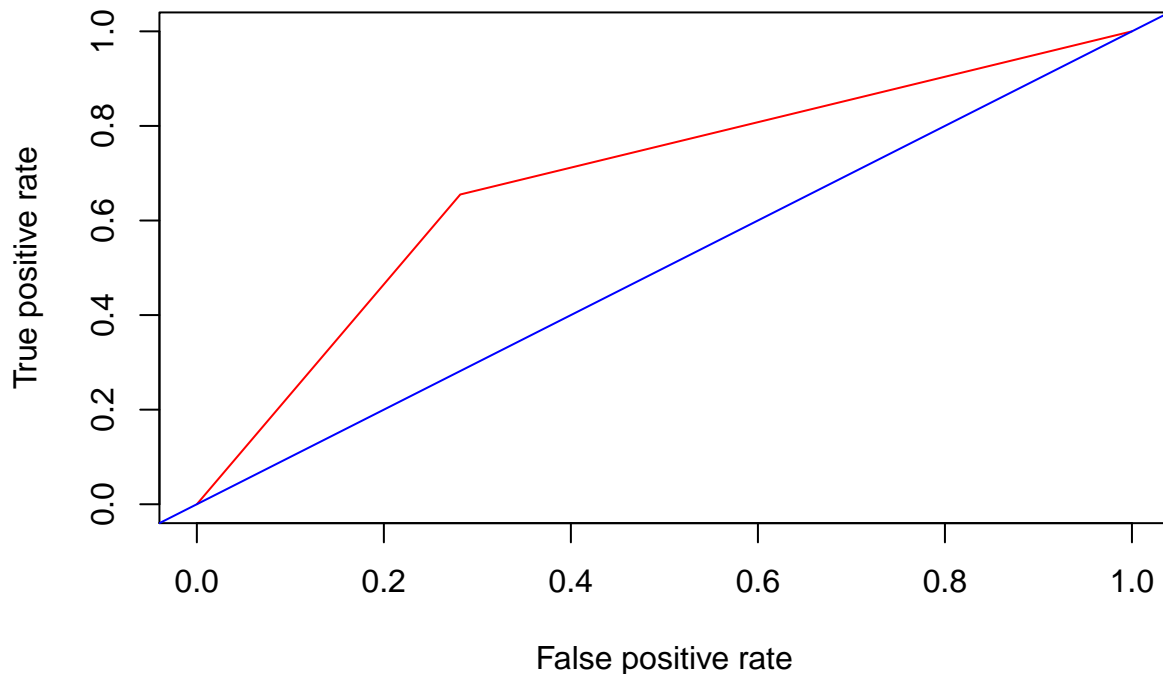
rbf_pred <- prediction(as.numeric(rbf_pred) , as.numeric(data_test$quality))
roc <- performance(rbf_pred,'tpr','fpr')
plot(roc , main = "ROC for RBF Kernel",col = 'red')
abline(a=0, b=1,col = 'blue')
```

## ROC for RBF Kernel



```
lin_pred <- prediction(as.numeric(lin_pred) , as.numeric(data_test$quality))
roc <- performance(lin_pred,'tpr','fpr')
plot(roc , main = "ROC for Linear Kernel",col = 'red')
abline(a=0, b=1,col = 'blue')
```

## ROC for Linear Kernel



## 4) Hierarchical Clustering

Consider the USArrests data. We will now perform hierarchical clustering on the states.

(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
data("USArrests")
data_sets<-USArrests
head(data_sets)
```

```
##           Murder Assault UrbanPop Rape
## Alabama      13.2      236       58 21.2
## Alaska       10.0      263       48 44.5
## Arizona       8.1      294       80 31.0
## Arkansas      8.8      190       50 19.5
## California    9.0      276       91 40.6
## Colorado      7.9      204       78 38.7
```

```
str(data_sets)
```

```
## 'data.frame':   50 obs. of  4 variables:
## $ Murder : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
## $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
## $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
```

```
## $ Rape      : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

```
any(is.na(data_sets))
```

```
## [1] FALSE
```

```
summary(data_sets)
```

```
##      Murder      Assault      UrbanPop      Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
## 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
## Median : 7.250   Median :159.0   Median :66.00   Median :20.10
## Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
## 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
## Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
data<-dist(data_sets, method = 'euclidean')
```

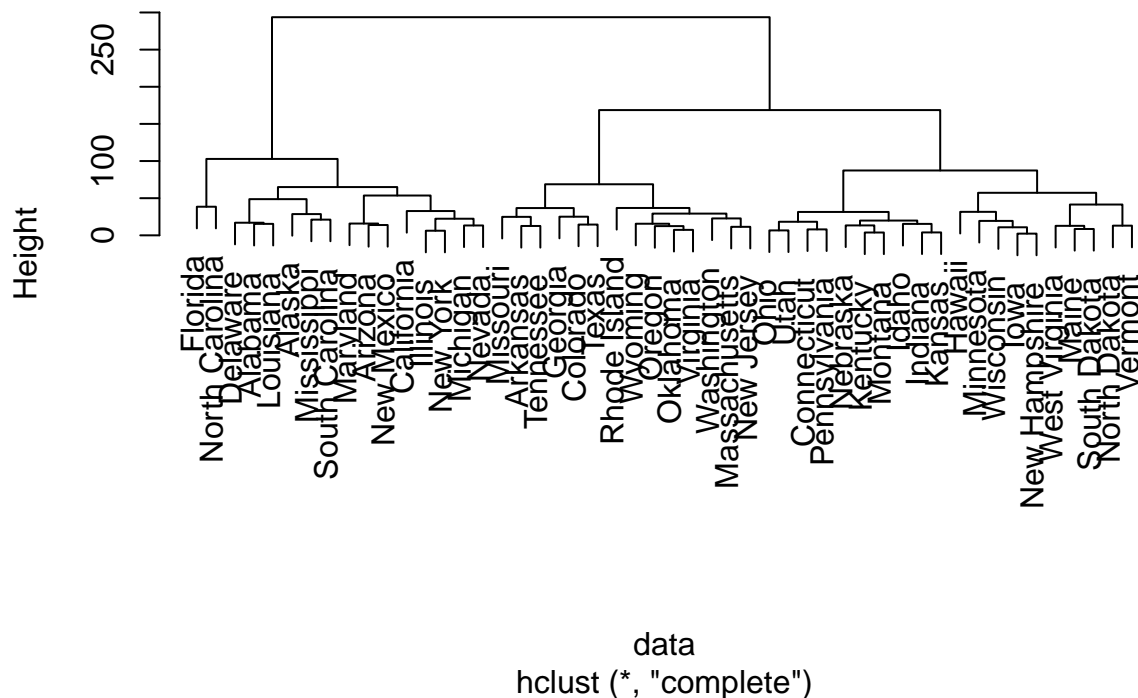
```
#hierarchical clustering
```

```
hiera_data<-hclust(data, method="complete")
```

```
#cluster the states
```

```
plot(hiera_data)
```

## Cluster Dendrogram



(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
clust_avg <- hclust(data, method = 'average')
```

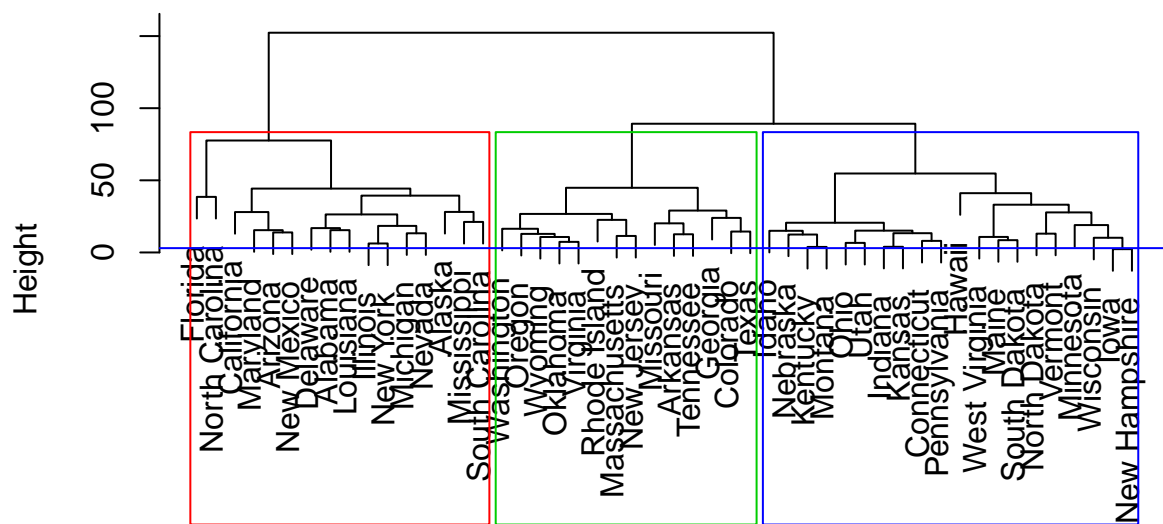
```
plot(clust_avg)
```

```
sort(cutree(clust_avg, k = 3))
```

```
##      Alabama      Alaska      Arizona      California      Delaware
##      1            1            1            1            1
##      Florida      Illinois      Louisiana      Maryland      Michigan
##      1            1            1            1            1
##      Mississippi      Nevada      New Mexico      New York      North Carolina
##      1            1            1            1            1
##      South Carolina      Arkansas      Colorado      Georgia      Massachusetts
##      1            2            2            2            2
##      Missouri      New Jersey      Oklahoma      Oregon      Rhode Island
##      2            2            2            2            2
##      Tennessee      Texas      Virginia      Washington      Wyoming
##      2            2            2            2            2
##      Connecticut      Hawaii      Idaho      Indiana      Iowa
##      3            3            3            3            3
##      Kansas      Kentucky      Maine      Minnesota      Montana
##      3            3            3            3            3
##      Nebraska      New Hampshire      North Dakota      Ohio      Pennsylvania
##      3            3            3            3            3
##      South Dakota      Utah      Vermont      West Virginia      Wisconsin
##      3            3            3            3            3
```

```
cut_avg <- cutree(clust_avg, k = 3)
plot(clust_avg)
rect.hclust(clust_avg, k = 3, border = 2:6)
abline(h = 3, col = 'blue')
```

## Cluster Dendrogram

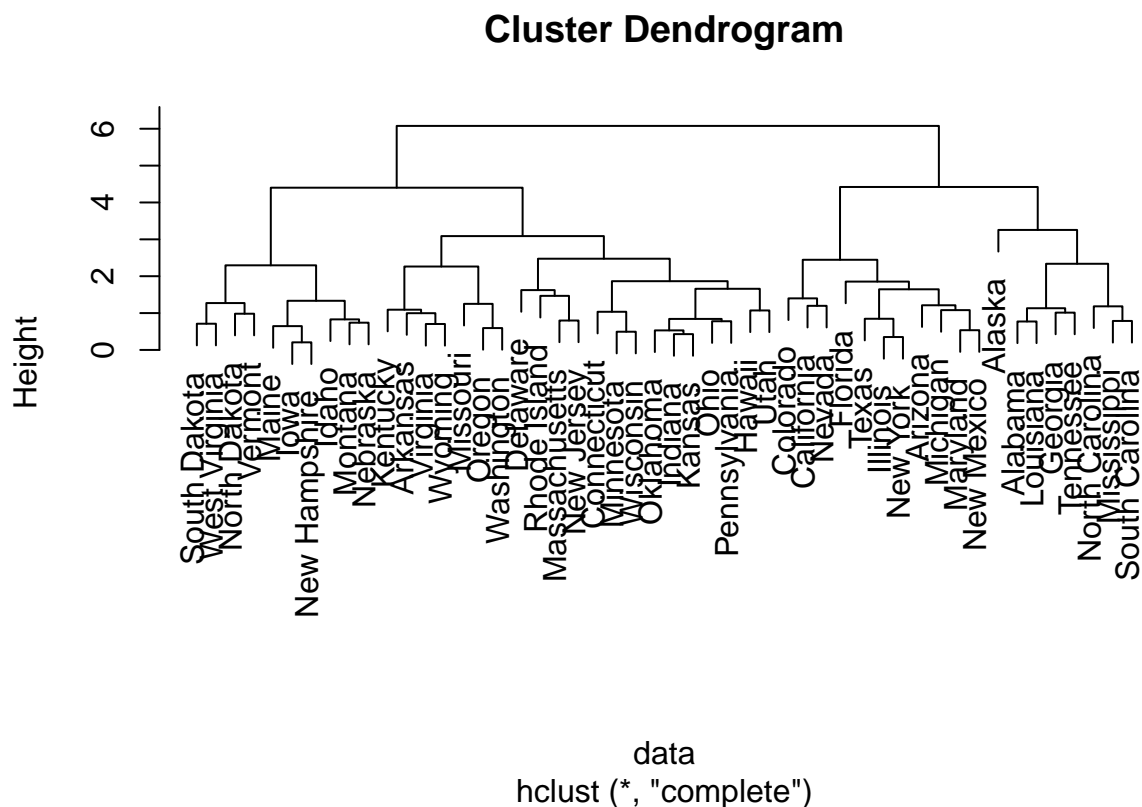


data  
hclust (\*, "average")

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
data_sets <- as.data.frame(scale(data_sets))

data<-dist(data_sets, method = 'euclidean')
#hierarchical clustering
hiera_data<-hclust(data)
#cluster the states
plot(hiera_data)
```



(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

In scaling we are transforming numerical values to get specific helpful properties. In scaling we are changing the range of data as we can see earlier the range/height was 0-300 and after scaling it shrinks to 0-6. We should have scaled the data before because whenever we have parameters/features that differ from each other in terms of range of values then you have to normalise the data so that the difference in these range of values does not affect your outcome.