Homework: Data Mining

This homework sheet will test your knowledge of data mining using R.

13

0

a) Load the files `Titanic.csv` into R as follows. This dataset provides information on the survival of the passengers of the Titanic.

```
titanic <- as.data.frame(read.csv("Titanic.csv", header = TRUE, sep = ","))
titanic$survived <- as.factor(titanic$survived)

# Remove NA observations
titanic <- na.omit(titanic[, c("survived", "pclass", "sex", "age", "sibsp",
    "parch", "fare", "embarked")])

# Number of observations
nrow(titanic)
```

```
## [1] 1045
```

```
# List of column names
colnames(titanic)
```

```
## [1] "survived" "pclass"   "sex"      "age"      "sibsp"    "parch"
## [7] "fare"     "embarked"
```

```
# View first rows of the dataset
head(titanic)
```

```
##   survived pclass    sex   age sibsp parch   fare embarked
## 1        1      1 female 29.00     0     0 211.34        S
## 2        1      1   male  0.92     1     2 151.55        S
## 3        0      1 female  2.00     1     2 151.55        S
## 4        0      1   male 30.00     1     2 151.55        S
## 5        0      1 female 25.00     1     2 151.55        S
## 6        1      1   male 48.00     0     0  26.55        S
```

Column Name	Interpretation
survival	Survival (0 = No; 1 = Yes)
pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
sex	Sex
age	Age
sibsp	Number of Siblings/Spouses Aboard
parch	Number of Parents/Children Aboard
fare	Passenger Fare
embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

1

b) Use the k-nearest neighbor method with $k = 3$ to predict if a 35-years-old person from 1st class survived?

Solution:

```
library(class)
```

```
train <- as.data.frame(cbind(titanic$age, titanic$pclass))
knn(train, c(35, 1), titanic$survived, k = 3)

## [1] 1
## Levels: 0 1
```

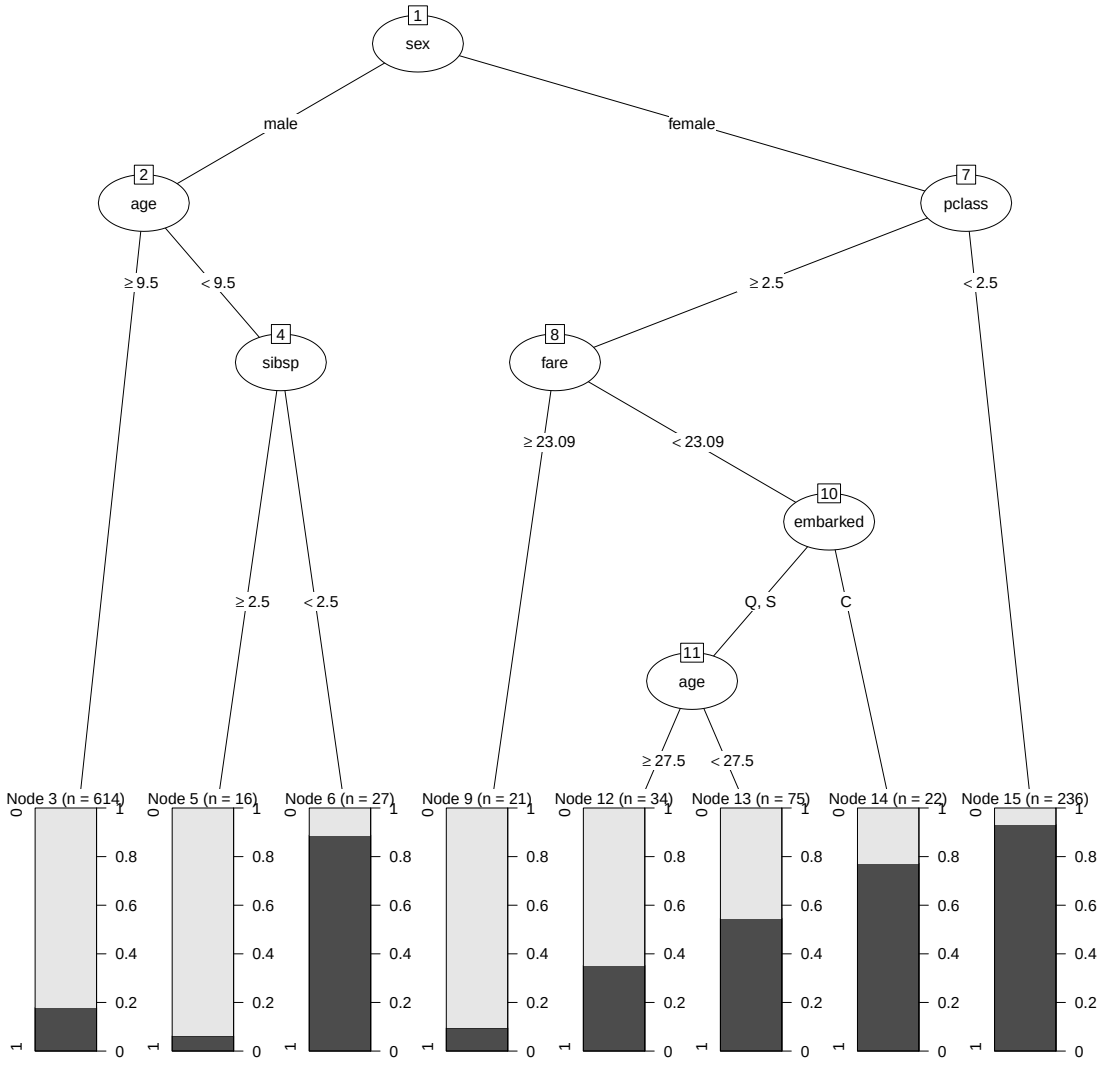
1

c) Build a decision tree with the `train` dataset, that shows which people on the Titanic survived. Use only the variables `pclass`, `age`, `sex`, `fare`, `sibsp`, `parch` and `embarked`. Plot and interpret your result.

Solution:

```
library(rpart)
library(party)
library(partykit)
```

```
dt <- rpart(survived ~ pclass + age + sex + fare + sibsp + parch + embarked,
            method="class", data=titanic)
plot(as.party(dt))
```



Generally speaking, a strong distinguishing power comes from the root note. Thus, *Sex* as the root node determines survival to a large extent. The next nodes are represented by *Age* and *Pclass* respectively. Finally, one can see that females traveling on first/second class have a very high survival rate.

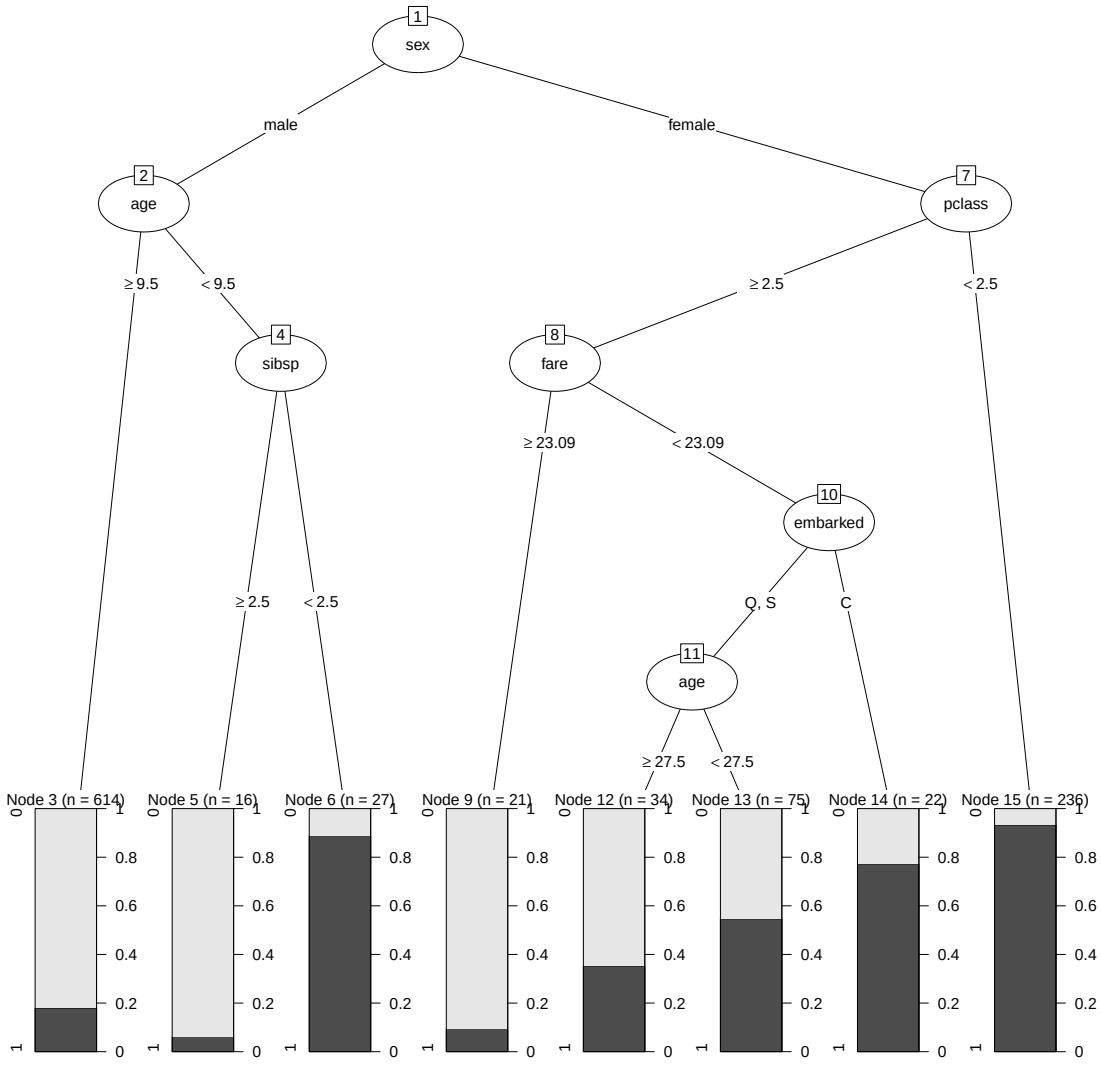
1

d) Prune the decision tree. Plot the pruned tree and interpret your result. Use R to calculate the number of nodes in the pruned tree.

Solution:

```
# Optimal size of nodes according to cross-validation error
which.min(dt$cptable[, "xerror"])

## 5
## 5
```

```
pruned <- prune(dt, cp = dt$cptable[which.min(dt$cptable[, "xerror"]), "CP"])
plot(as.party(pruned))
```



Sex remains the most distinguishing feature. Elder males (with age above 9.5 years) have a relatively low chance of surviving, in contrast to women from 1st/2nd class

1

e) Does the decision tree support the hypothesis that *women and children are saved first*? Why is it difficult to analyze this hypothesis with OLS?

Solution:

When looking at the position of *Age* and *Sex*, we can see that other males have, overall, a very low possibility of survival. As we clearly have a non-linear effect, this hypothesis is difficult to verify by a linear model.

As a remedy, one has to use e. g. interaction terms.

1

f) Split the data into two subsets for training (20 %) and testing (80 %). Recalculate the above pruned decision tree and measure the prediction performance using the confusion matrix, as well as accuracy, precision, sensitivity and F1 score.

Solution:

```
inTrain <- runif(nrow(titanic)) < 0.2
dt <- rpart(survived ~ pclass + age + sex + fare + sibsp + parch + embarked,
    method = "class", data = titanic[inTrain, ])
pruned <- prune(dt, cp = dt$cptable[which.min(dt$cptable[, "xerror"]), "CP"])

pred <- predict(pruned, titanic[-inTrain, ], type = "class")
# Confusion matrix
cm <- table(pred = pred, true = titanic$survived[-inTrain])
cm

##      true
## pred   0   1
##    0 584 182
##    1  34 244

# Accuracy
sum(diag(cm))/sum(sum(cm))

## [1] 0.7931

# Precision
cm[1, 1]/(cm[1, 1] + cm[1, 2])

## [1] 0.7624

# Sensitivity
cm[1, 1]/(cm[1, 1] + cm[2, 1])

## [1] 0.945

# F1 score
2 * cm[1, 1]/(2 * cm[1, 1] + cm[1, 2] + cm[2, 1])

## [1] 0.8439
```

1

g) Split the data into two subsets for training (20 %) and testing (80 %). Train a neural network with 10 nodes in the hidden layer. Use only the variables `pclass`, `age`, `sex`, `fare`, `sibsp`, `parch` and `embarked`. Calculate the confusion matrix, as well as accuracy, precision, sensitivity and F1 score!

Solution:

```
library(nnet)
```

```
inTrain <- runif(nrow(train)) < 0.2
ann <- nnet(survived ~ pclass + age + sex + fare + sibsp + parch + embarked,
    data = titanic[inTrain, ], size = 10, maxit = 100, rang = 0.1, decay = 5e-04)

## # weights:  111
## initial  value 146.442985
## iter  10 value 120.366054
## iter  20 value 91.205901
## iter  30 value 86.912829
## iter  40 value 84.132313
## iter  50 value 75.846613
## iter  60 value 66.863361
## iter  70 value 62.328771
## iter  80 value 60.346086
## iter  90 value 58.972875
## iter 100 value 58.590077
## final  value 58.590077
## stopped after 100 iterations

pred <- predict(ann, titanic[-inTrain, ], type = "class")
cm <- table(pred = pred, true = titanic$survived[-inTrain])
# Confusion matrix
cm <- table(pred = pred, true = titanic$survived[-inTrain])
cm

##     true
## pred   0   1
##    0 547 165
##    1  71 261

# Accuracy
sum(diag(cm))/sum(sum(cm))

## [1] 0.7739

# Precision
cm[1, 1]/(cm[1, 1] + cm[1, 2])

## [1] 0.7683

# Sensitivity
cm[1, 1]/(cm[1, 1] + cm[2, 1])

## [1] 0.8851

# F1 score
2 * cm[1, 1]/(2 * cm[1, 1] + cm[1, 2] + cm[2, 1])

## [1] 0.8226
```

1

h) Now train a neural network with 50 nodes in the hidden layer and a maximum of 1000 iterations (`maxit=1000`). Compare the performance to the previous neural network. What is the cause of the drop in prediction performance, even though we increased both the number of neurons and the training effort?

Solution:

```
ann2 <- nnet(survived ~ pclass + age + sex + fare + sibsp + parch + embarked,
    data = titanic[inTrain, ], size = 50, maxit = 1000, rang = 0.1, decay = 5e-04)

## # weights:  551
## initial  value 153.992853
## iter  10 value 119.038227
## iter  20 value 105.768115
## iter  30 value 83.666900
## iter  40 value 65.600160
## iter  50 value 58.660470
## iter  60 value 47.214664
## iter  70 value 39.568016
## iter  80 value 33.824383
## iter  90 value 27.359904
## iter 100 value 21.928227
## iter 110 value 18.559452
## iter 120 value 17.017699
## iter 130 value 16.674654
## iter 140 value 16.183846
## iter 150 value 14.867264
## iter 160 value 12.880257
## iter 170 value 11.513201
## iter 180 value 10.266482
## iter 190 value 9.612301
## iter 200 value 8.286907
## iter 210 value 6.644052
## iter 220 value 6.049942
## iter 230 value 5.683880
## iter 240 value 5.325210
## iter 250 value 5.201377
## iter 260 value 4.971094
## iter 270 value 4.865632
## iter 280 value 4.775127
## iter 290 value 4.713007
## iter 300 value 4.662917
## iter 310 value 4.604335
## iter 320 value 4.535568
## iter 330 value 4.450913
## iter 340 value 4.406480
## iter 350 value 4.365136
## iter 360 value 4.343770
## iter 370 value 4.340256
```

```
## iter 380 value 4.334456
## iter 390 value 4.322793
## iter 400 value 4.307611
## iter 410 value 4.279850
## iter 420 value 4.252162
## iter 430 value 4.222180
## iter 440 value 4.191748
## iter 450 value 4.170319
## iter 460 value 4.145446
## iter 470 value 4.123724
## iter 480 value 4.106774
## iter 490 value 4.092605
## iter 500 value 4.080908
## iter 510 value 4.070128
## iter 520 value 4.060507
## iter 530 value 4.049078
## iter 540 value 4.032480
## iter 550 value 4.004098
## iter 560 value 3.970268
## iter 570 value 3.944075
## iter 580 value 3.916640
## iter 590 value 3.895990
## iter 600 value 3.879744
## iter 610 value 3.869382
## iter 620 value 3.860096
## iter 630 value 3.853037
## iter 640 value 3.848306
## iter 650 value 3.843514
## iter 660 value 3.839291
## iter 670 value 3.835290
## iter 680 value 3.831139
## iter 690 value 3.826184
## iter 700 value 3.820711
## iter 710 value 3.812902
## iter 720 value 3.805950
## iter 730 value 3.799143
## iter 740 value 3.793635
## iter 750 value 3.787813
## iter 760 value 3.782462
## iter 770 value 3.777692
## iter 780 value 3.773872
## iter 790 value 3.770759
## iter 800 value 3.767196
## iter 810 value 3.764047
## iter 820 value 3.761174
## iter 830 value 3.758768
## iter 840 value 3.756454
## iter 850 value 3.754659
## iter 860 value 3.753133
## iter 870 value 3.751885
## iter 880 value 3.750874
## iter 890 value 3.749709
## iter 900 value 3.748697
## iter 910 value 3.747841
```

```
## iter 920 value 3.746889
## iter 930 value 3.745960
## iter 940 value 3.744924
## iter 950 value 3.744032
## iter 960 value 3.743049
## iter 970 value 3.741921
## iter 980 value 3.740907
## iter 990 value 3.740059
## iter1000 value 3.738835
## final  value 3.738835
## stopped after 1000 iterations

pred <- predict(ann2, titanic[-inTrain, ], type = "class")
cm <- table(pred = pred, true = titanic$survived[-inTrain])
# Confusion matrix
cm <- table(pred = pred, true = titanic$survived[-inTrain])
cm

##      true
## pred   0   1
##    0 518 153
##    1 100 273

# Accuracy
sum(diag(cm))/sum(sum(cm))

## [1] 0.7577

# Precision
cm[1, 1]/(cm[1, 1] + cm[1, 2])

## [1] 0.772

# Sensitivity
cm[1, 1]/(cm[1, 1] + cm[2, 1])

## [1] 0.8382

# F1 score
2 * cm[1, 1]/(2 * cm[1, 1] + cm[1, 2] + cm[2, 1])

## [1] 0.8037
```

Both accuracy and precision decrease. A possible cause of this behavior might be overfitting. With only 7 input values, a total of 50 neurons in the hidden layer might be too much.

1

i) Split the data into two subsets for training (20 %) and testing (80 %). Train a support vector machine and predict the survival. Use only the variables `pclass`, `age`, `sex`, `fare`, `sibsp`, `parch` and `embarked`. Calculate the confusion matrix, as well as accuracy, precision, sensitivity and F1 score!

Solution:

```
library(e1071)
```

```
inTrain <- runif(nrow(titanic)) < 0.2
model.svm <- svm(survived ~ pclass + age + sex + fare + sibsp + parch + embarked,
                 data=titanic[inTrain,], type="C-classification")
pred <- predict(model.svm, titanic[-inTrain,])

# Confusion matrix
cm <- table(pred=pred, true=titanic$survived[-inTrain])
cm

##     true
## pred   0   1
##    0 544 146
##    1  74 280

# Accuracy
sum(diag(cm))/sum(sum(cm))

## [1] 0.7893

# Precision
cm[1, 1]/(cm[1, 1] + cm[1, 2])

## [1] 0.7884

# Sensitivity
cm[1, 1]/(cm[1, 1] + cm[2, 1])

## [1] 0.8803

# F1 score
2*cm[1, 1]/(2*cm[1,1]+cm[1, 2]+cm[2, 1])

## [1] 0.8318
```

1

j) Which of the above machine learning approaches has the best performance?

Solution:

Model	Accuracy	Precision	Sensitivity	F1
Decision Tree (Pruned)	0.7931	0.7624	0.945	0.8439
Neural Network (10 Hidden Neurons)	0.7739	0.7683	0.8851	0.8226
Support Vector Machine	0.7893	0.7884	0.8803	0.8318

In the given setting, one achieves the highest accuracy with decision trees, while support vector machines

have the highest precision rate. The best trade-off between precision and sensitivity originates from using the SVM.

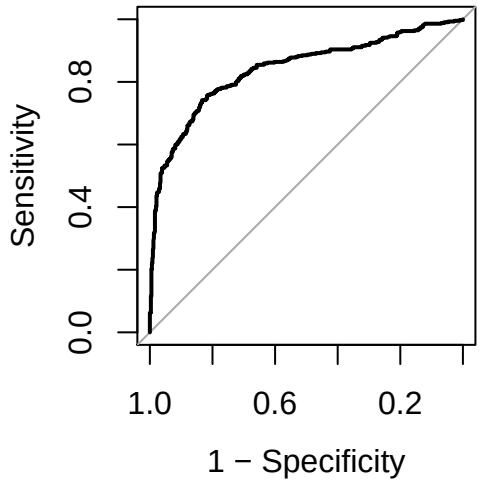
1

k) Plot the receiver operating curve (ROC) for the trained SVM.

Solution:

```
library(pROC)
```

```
pred <- predict(model.svm, titanic[-inTrain, ], decision.values = TRUE)
dv <- attributes(pred)$decision.values
plot.roc(as.numeric(titanic$survived[-inTrain]), dv, xlab = "1 - Specificity")
```




```
##
## Call:
## plot.roc.default(x = as.numeric(titanic$survived[-inTrain]),     predictor = dv, xlab = "1 - Specificity
##
## Data: dv in 618 controls (as.numeric(titanic$survived[-inTrain]) 1) > 426 cases (as.numeric(titanic$surv
## Area under the curve: 0.84
```

1

l) Perform a k-means clustering to determine the origin of wines. Use $k = 3$ means with $n = 10$ random initializations. What is the within-cluster sum of squares (WCSS) value? As the input data we use the dataset `wines` that is included in the `kohonen` package.

```
library(kohonen)
data(wines)
```

The dataset contains 177 rows and thirteen columns; object *vintages* contains the class labels. For compatibility with older versions of the package, variable wine.classes is retained too. This data is the

result of chemical analyses of wines grown in the same region of Italy (Piedmont) but derived from three different cultivars: Nebbiolo, Barberas and Grignolino grapes. The wine from the Nebbiolo grape is called Barolo. The data contains the quantities of several constituents found in each of the three types of wines, as well as some spectroscopic variables.

```
head(wines)
```

```
##      alcohol malic acid  ash ash alkalinity magnesium tot. phenols
## [1,]   13.20       1.78 2.14           11.2       100         2.65
## [2,]   13.16       2.36 2.67           18.6       101         2.80
## [3,]   14.37       1.95 2.50           16.8       113         3.85
## [4,]   13.24       2.59 2.87           21.0       118         2.80
## [5,]   14.20       1.76 2.45           15.2       112         3.27
## [6,]   14.39       1.87 2.45           14.6        96         2.50
##      flavonoids non-flav. phenols proanth col. int. col. hue OD ratio
## [1,]       2.76              0.26    1.28      4.38     1.05     3.40
## [2,]       3.24              0.30    2.81      5.68     1.03     3.17
## [3,]       3.49              0.24    2.18      7.80     0.86     3.45
## [4,]       2.69              0.39    1.82      4.32     1.04     2.93
## [5,]       3.39              0.34    1.97      6.75     1.05     2.85
## [6,]       2.52              0.30    1.98      5.25     1.02     3.58
##      proline
## [1,]    1050
## [2,]    1185
## [3,]    1480
## [4,]     735
## [5,]    1450
## [6,]    1290
```

Solution:

```
km <- kmeans(wines, 3, nstart = 10)
km
```

```
## K-means clustering with 3 clusters of sizes 62, 46, 69
##
## Cluster means:
##   alcohol malic acid   ash ash alkalinity magnesium tot. phenols
## 1   12.93      2.504 2.408          19.89    103.60        2.111
## 2   13.80      1.887 2.426          17.05    105.04        2.869
## 3   12.52      2.494 2.289          20.82     92.35        2.071
##   flavonoids non-flav. phenols proanth col. int. col. hue OD ratio proline
## 1      1.584            0.3884   1.503     5.650   0.8840    2.365   728.3
## 2      3.013            0.2854   1.902     5.704   1.0791    3.097  1198.0
## 3      1.758            0.3901   1.452     4.087   0.9412    2.491   458.2
##
## Clustering vector:
##   [1] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 2 2 1 2 2 2 2 2 2 1
##  [36] 1 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 3 1 3 3 1 3 3 1 1 1
##  [71] 3 3 2 1 3 3 3 1 3 3 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 3 1 3 1 3 3 3 1 3
## [106] 3 3 3 1 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 3 3 1 1 1 1 3 3 3 1
## [141] 1 3 3 1 1 3 1 1 3 3 3 3 1 1 1 3 1 1 1 3 1 3 1 1 3 1 1 1 1 3 3 1 1 1 1
## [176] 1 3
##
```

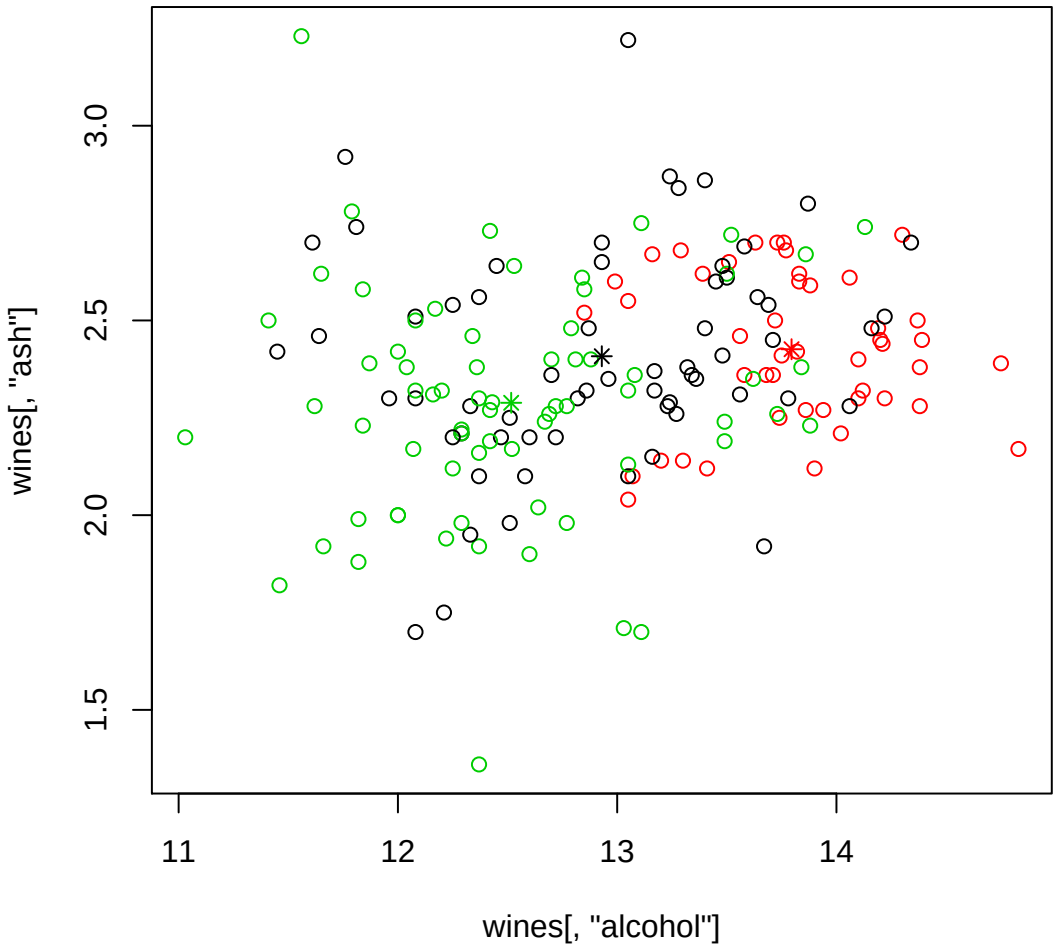
```
## Within cluster sum of squares by cluster:
## [1]  566573 1343169  443167
##  (between_SS / total_SS =  86.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"

km$tot.withinss

## [1] 2352908
```

1

m) Use the above result from the clustering procedure to plot data points and clusters in a 2-dimensional plot showing only the dimensions `alcohol` and `ash`.

Solution:

```
plot(wines[, "alcohol"], wines[, "ash"], col = km$cluster)
points(km$centers[, "alcohol"], km$centers[, "ash"], col = 1:nrow(km$centers),
    pch = 8)
```

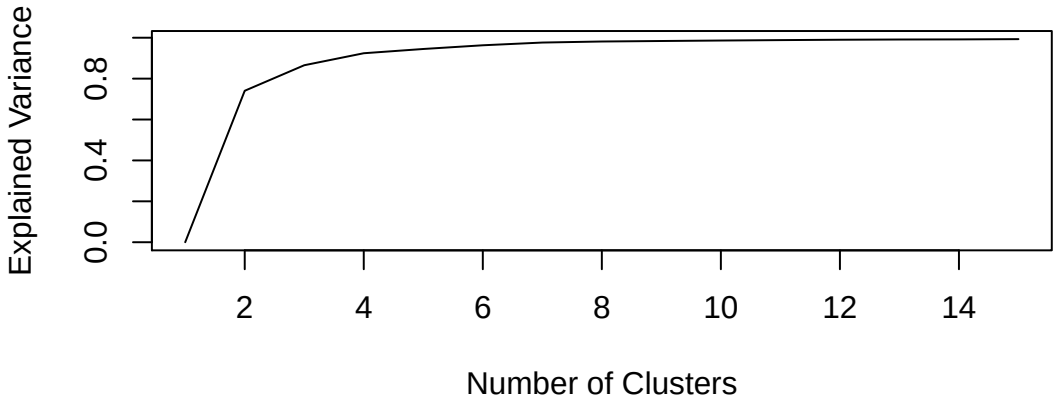


1

n) What is the recommended number of clusters according to the Elbow plot?

Solution:

```
ev <- c()
for (i in 1:15) {
    km <- kmeans(wines, i, nstart = 10)
    ev[i] <- sum(km$betweenss)/km$totss
}
plot(1:15, ev, type = "l", xlab = "Number of Clusters", ylab = "Explained Variance")
```



The recommended number seems to be 3 clusters, matching the data description naming 3 origins.