Завантажимо необхідні бібліотеки та дані; початковий датасет розділимо на дві частини - навчальну та тестову вибірки (доступ до рядків та колонок здійснюється за допомогою квадратних дужок [*рядок*,*стовпець*]). Командою str() переглядаємо основні відомості:

```
library("rpart.plot")
library(readr)
library(magrittr)
library(Metrics)
library(factoextra)
library(caret)
library(class)
library(dplyr)
library(e1071)
library(psych)
library(factoextra)

data <- read.csv(file = "D:/train.csv", header = TRUE)
train <- data[1000:1999,]
test <- train[1:999, -2]
str(train)
```

```
## 'data.frame':    1000 obs. of  9 variables:
##  $ Dates     : chr  "2015-05-11 11:00:00" "2015-05-11 11:00:00" "2015-05-11 11:00:00" "2015-05-11 11:00:00"
...
##  $ Category  : chr  "MISSING PERSON" "LARCENY/THEFT" "OTHER OFFENSES" "LARCENY/THEFT" ...
##  $ Descript  : chr  "FOUND PERSON" "THEFT OF CHECKS OR CREDIT CARDS" "FALSE PERSONATION" "GRAND THEFT FROM LOC
KED AUTO" ...
##  $ DayOfWeek : chr  "Monday" "Monday" "Monday" "Monday" ...
##  $ PdDistrict: chr  "INGLESIDE" "SOUTHERN" "MISSION" "CENTRAL" ...
##  $ Resolution: chr  "NONE" "NONE" "NONE" "NONE" ...
##  $ Address   : chr  "200 Block of CAYUGA AV" "0 Block of HOWARD ST" "400 Block of DOLORES ST" "NORTHPOINT ST /
MASON ST" ...
##  $ X         : num  -122 -122 -122 -122 -122 ...
##  $ Y         : num  37.7 37.8 37.8 37.8 37.8 ...
```

Попередня обробка даних:

Далі бачимо невідповідність змінних та їхніх типів, тому змінюємо останні за допомогою явних перетворень.
Оскільки змінна Dates має тип POSIXct (дата та час), використовуємо відповідну функцію для задання формату, за яким перетворюватимуться дані.

Змінні Month, Year, Day, Hour та Category матимуть тип factor (категоріальний тип, який може приймати як числові, так і символьні значення).

```
train$Dates <- as.POSIXct(train$Dates, format="%Y-%m-%d %H:%M:%S")
train$Month <- factor(format(train$Dates, "%m"))
train$Year <- factor(format(train$Dates, "%Y"))
train$Day <- factor(format(train$Dates,"%d"))
train$Hour <- factor(format(train$Dates,"%H"))
train$Category <- factor(train$Category)
```

Ті ж маніпуляції проводимо із тестовою вибіркою. Варто зауважити, що в ній відсутня цільова змінна Category, значення якої ми й хочемо навчитися встановлювати за допомогою класифікаційного аналізу.

```
test$Dates <- as.POSIXct(test$Dates, format="%Y-%m-%d %H:%M:%S")
test$Month <- factor(format(test$Dates, "%m"))
test$Year <- factor(format(test$Dates, "%Y"))
test$Day <- factor(format(test$Dates,"%d"))
test$Hour <- factor(format(test$Dates,"%H"))
```

Наступним етапом стане **побудова дерева рішень**.
Спершу необхідно створити файл, у який запишеться результат. Тоді, власне, використовуємо функію rpart() для створення дерева рішень.

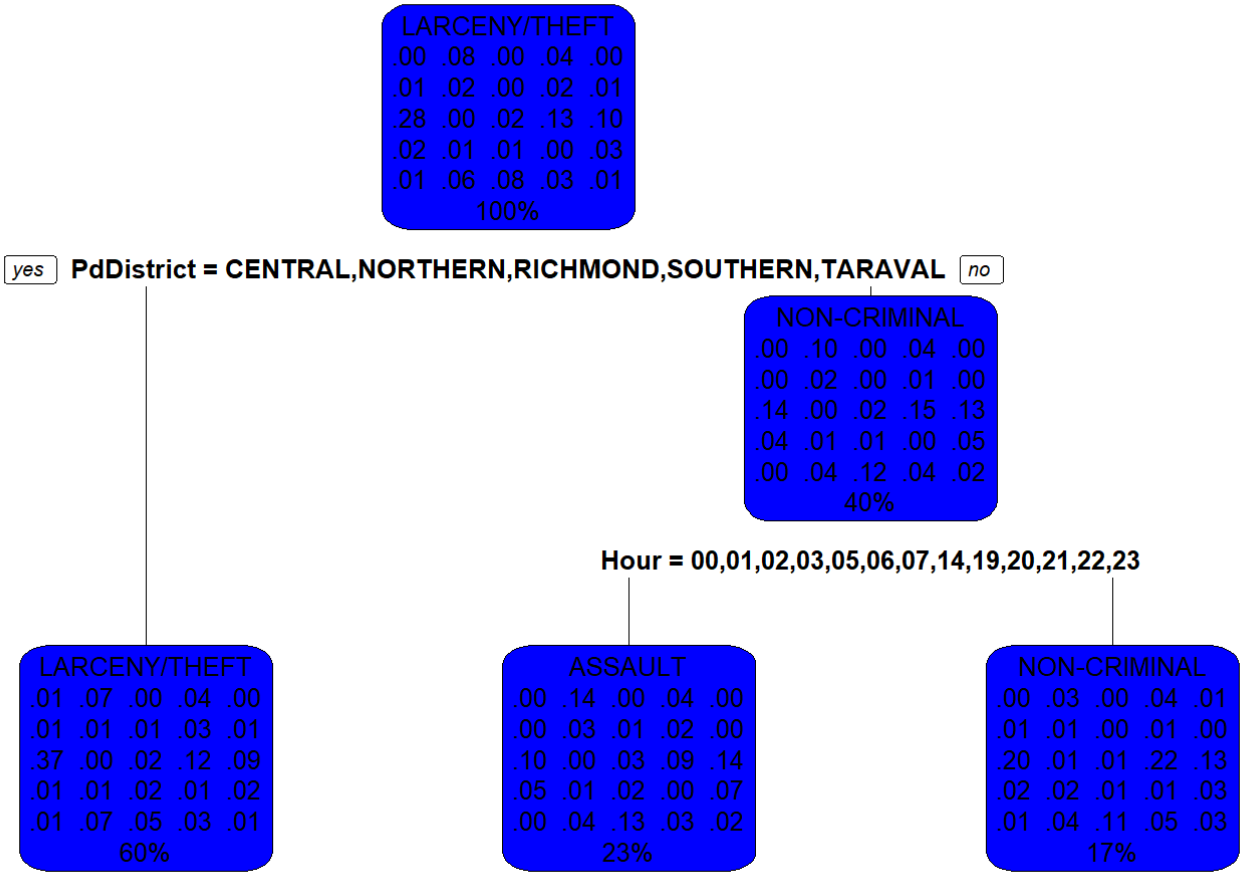
```
pdf(file = "decision_tree2.pdf")
output.tree <- rpart(Category ~ Hour + Day + Year + PdDistrict,
                     data = train,
                     method = "class",
```

```
                      control = rpart.control(minsplit = 400,cp=0))
dev.off()
```

```
## png
##   2
```

(На рисунку зображена невелика частина дерева рішень)

```
rpart.plot(output.tree, box.palette = "blue")
```



Використовуючи функцію predict() та тестову вибірку, прогнозуємо можливе значення змінної Category. У результаті отримуємо матрицю із апостеріорними ймовірностями набуття змінної того чи іншого значення.

(Назви колонок задаємо як значення рівнів Category).

```
predicted <- predict(object = output.tree, newdata = test)
final <- data.frame(predicted)
colnames(final)  <- c(levels(train$Category))
```

Записуємо результат у новий файл та перенумеровуємо рядки. Тоді виводимо матрицю результатів на екран.

```
write.csv(final,file = "Decision_Tree1.csv",row.names = FALSE,quote = F)
result <- read.csv(file = "Decision_Tree1.csv", header = TRUE)
rownames(result)<-1:nrow(result)
head(as.matrix(result), 5)
```

```
##         ARSON    ASSAULT     BRIBERY   BURGLARY DISORDERLY.CONDUCT DRIVING.UNDER.THE.INFLUENCE DRUG.NARCOTIC
## 1 0.000000000 0.03488372 0.000000000 0.04069767        0.005813953                 0.005813953    0.01162791
## 2 0.005008347 0.06677796 0.001669449 0.03672788        0.001669449                 0.006677796    0.01168614
## 3 0.000000000 0.03488372 0.000000000 0.04069767        0.005813953                 0.005813953    0.01162791
## 4 0.005008347 0.06677796 0.001669449 0.03672788        0.001669449                 0.006677796    0.01168614
## 5 0.005008347 0.06677796 0.001669449 0.03672788        0.001669449                 0.006677796    0.01168614
##   DRUNKENNESS      FRAUD  KIDNAPPING LARCENY.THEFT LIQUOR.LAWS MISSING.PERSON NON.CRIMINAL OTHER.OFFENSES
## 1 0.000000000 0.01162791 0.000000000     0.1976744 0.005813953     0.01162791    0.2209302      0.1279070
## 2 0.005008347 0.02504174 0.008347245     0.3722871 0.001669449     0.02337229    0.1168614      0.0851419
## 3 0.000000000 0.01162791 0.000000000     0.1976744 0.005813953     0.01162791    0.2209302      0.1279070
## 4 0.005008347 0.02504174 0.008347245     0.3722871 0.001669449     0.02337229    0.1168614      0.0851419
## 5 0.005008347 0.02504174 0.008347245     0.3722871 0.001669449     0.02337229    0.1168614      0.0851419
##      ROBBERY SECONDARY.CODES SEX.OFFENSES.FORCIBLE STOLEN.PROPERTY SUSPICIOUS.OCC    TRESPASS  VANDALISM
## 1 0.01744186     0.017441860           0.005813953     0.011627907     0.02906977 0.005813953 0.04069767
## 2 0.01335559     0.008347245           0.015025042     0.005008347     0.02170284 0.013355593 0.07011686
## 3 0.01744186     0.017441860           0.005813953     0.011627907     0.02906977 0.005813953 0.04069767
## 4 0.01335559     0.008347245           0.015025042     0.005008347     0.02170284 0.013355593 0.07011686
## 5 0.01335559     0.008347245           0.015025042     0.005008347     0.02170284 0.013355593 0.07011686
##   VEHICLE.THEFT   WARRANTS WEAPON.LAWS
## 1    0.11046512 0.05232558 0.034883721
## 2    0.05175292 0.02838063 0.005008347
## 3    0.11046512 0.05232558 0.034883721
## 4    0.05175292 0.02838063 0.005008347
## 5    0.05175292 0.02838063 0.005008347
```

Далі переходимо до **кластерного аналізу**.

```
test <- read.csv("D:/test.csv", header = TRUE)
test <- test[1:1000,]
str(test)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ Id        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Dates     : chr  "2015-05-10 23:59:00" "2015-05-10 23:51:00" "2015-05-10 23:50:00" "2015-05-10 23:45:00"
...
##  $ DayOfWeek : chr  "Sunday" "Sunday" "Sunday" "Sunday" ...
##  $ PdDistrict: chr  "BAYVIEW" "BAYVIEW" "NORTHERN" "INGLESIDE" ...
##  $ Address   : chr  "2000 Block of THOMAS AV" "3RD ST / REVERE AV" "2000 Block of GOUGH ST" "4700 Block of MIS
SION ST" ...
##  $ X         : num  -122 -122 -122 -122 -122 ...
##  $ Y         : num  37.7 37.7 37.8 37.7 37.7 ...
```

Здійснимо **попередню обробку даних**:

1) перетворюємо типи змінних до потрібних;

2) фільтруємо дані, прибравши actor-змінні, які мають занадто багато або лишень один рівень.

```
test$Dates <- strptime(test$Dates, "%Y-%m-%d %H:%M:%S")
test$Dates <- as.POSIXct(test$Dates, format="%Y-%m-%d %H:%M:%S")
test$Month <- factor(format(test$Dates, "%m"))
test$Year <- factor(format(test$Dates, "%Y"))
test$Day <- factor(format(test$Dates,"%d"))
test$Hour <- factor(format(test$Dates,"%H"))
test <- test[1:1000,3:11]
test <- test[,-c(6,7,9)]
mydata <- test
head(as.matrix(mydata), 6)
```

```
##   DayOfWeek PdDistrict  Address                    X           Y          Day
## 1 "Sunday"  "BAYVIEW"   "2000 Block of THOMAS AV"  "-122.3996" "37.73505" "10"
## 2 "Sunday"  "BAYVIEW"   "3RD ST / REVERE AV"       "-122.3915" "37.73243" "10"
## 3 "Sunday"  "NORTHERN"  "2000 Block of GOUGH ST"   "-122.4260" "37.79221" "10"
## 4 "Sunday"  "INGLESIDE" "4700 Block of MISSION ST" "-122.4374" "37.72141" "10"
## 5 "Sunday"  "INGLESIDE" "4700 Block of MISSION ST" "-122.4374" "37.72141" "10"
## 6 "Sunday"  "TARAVAL"   "BROAD ST / CAPITOL AV"    "-122.4590" "37.71317" "10"
```

Тепер потрібно попрацювати окремо з factor-змінними та змінними типу char і numeric, котрі мають три та більше рівні, проте можуть бути конвертовані неправильно. Їх ми відобразимо у вигляді таблиця, які потім з'єднаємо із початковою таблицею.
Функція dummy.code() дозволяє отримати стільки таблиць із нулів та одиниць, скільки рівнів має factor-змінна.

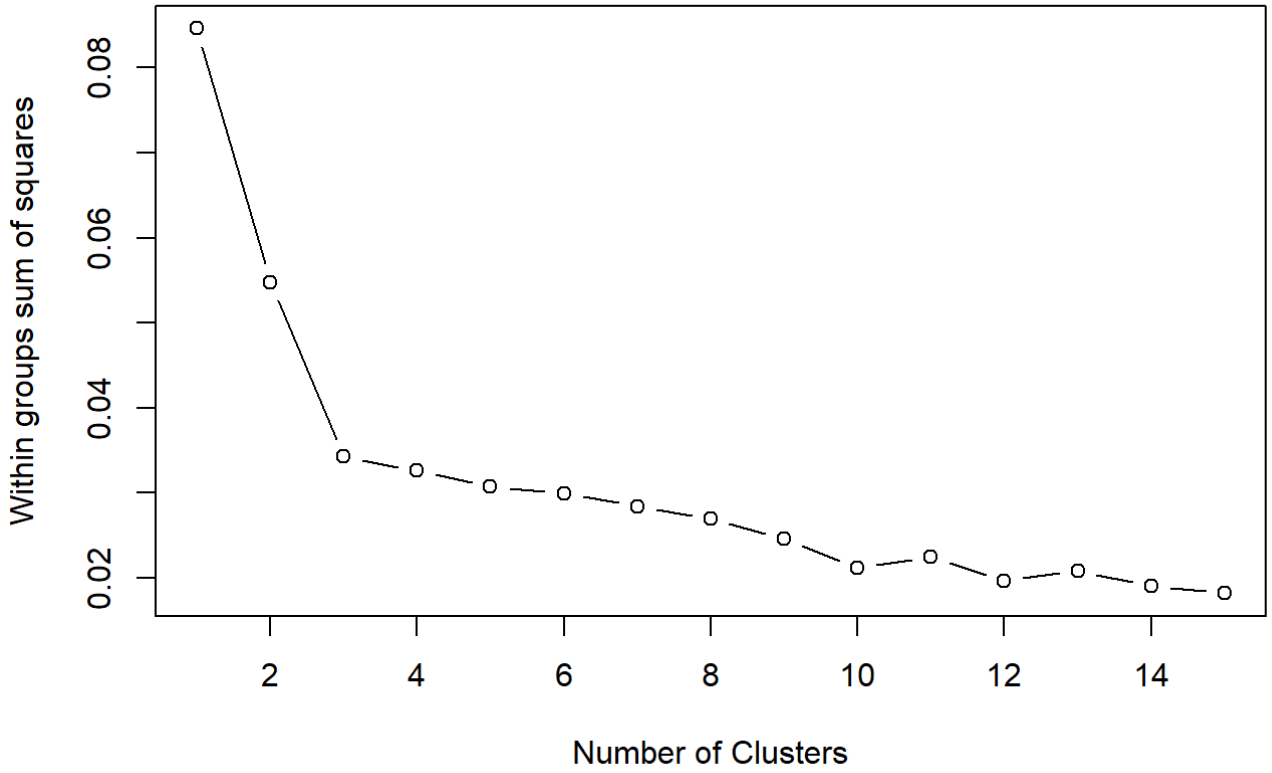
```
mydata$Day <- as.numeric(as.factor(mydata$Day))
DayOfWeek <- as.data.frame(dummy.code(mydata$DayOfWeek))
PdDistrict <- as.data.frame(dummy.code(mydata$PdDistrict))
mydata <- cbind(mydata, DayOfWeek, PdDistrict)
```

Тепер здійснимо перевірку на присутність порожніх рядків, а також змаштабуємо дані за допомогою функії scale().

```
mydata <- mydata[,-c(1,2)]
mydata <- mydata[,-1]
mydata <- na.omit(mydata)
mydata <- scale(mydata)
```

Перейдемо далі до самого **кластерного аналізу**.
Скористаємося алгоритмом, що дозволяє визначити оптимальну кількість кластерів. З рисунку нижче помітно: мінімальна відстань всередині кластерів досягається при *n=14*.

```
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```



Отже, випробуємо нашу модель. Результати демонструють загальну згрупованість на рівні 81%, що означає непогану точність нашої моделі.

```
res <- kmeans(mydata, 14, nstart = 16)
print(res)
```

```
## K-means clustering with 14 clusters of sizes 54, 46, 158, 29, 65, 17, 143, 116, 51, 106, 111, 21, 41, 42
##
## Cluster means:
##                X         Y       Day  Saturday    Sunday    Friday  SOUTHERN  NORTHERN   CENTRAL   MISSION
## 1  0.0005439870 0.9998603 0.7766409 0.7704035 0.7641661 0.7641661 0.7641661 0.7704035 0.7641661 0.7641661
## 2  0.0006502272 0.9999270 0.7704035 0.7641661 0.7641661 0.7704035 0.7641661 0.7641661 0.7704035 0.7641661
## 3  0.0004347208 0.9996288 0.7766409 0.7704035 0.7641661 0.7641661 0.7641661 0.7641661 0.7641661 0.7658636
## 4  0.0005186370 0.9994843 0.7828783 0.7641661 0.7704035 0.7641661 0.7641661 0.7641661 0.7641661 0.7641661
## 5  0.0006783552 0.9998129 0.7704035 0.7641661 0.7641661 0.7704035 0.7704035 0.7641661 0.7641661 0.7641661
## 6  0.0007372383 0.9995398 0.7828783 0.7641661 0.7704035 0.7641661 0.7641661 0.7641661 0.7641661 0.7641661
## 7  0.0004905991 0.9996528 0.7704035 0.7641661 0.7641661 0.7704035 0.7641661 0.7641661 0.7641661 0.7655619
## 8  0.0003695117 0.9997639 0.7828783 0.7641661 0.7704035 0.7641661 0.7641661 0.7667471 0.7641661 0.7641661
## 9  0.0006618042 0.9999179 0.7766409 0.7704035 0.7641661 0.7641661 0.7641661 0.7641661 0.7704035 0.7641661
## 10 0.0006801474 0.9997699 0.7766409 0.7704035 0.7641661 0.7641661 0.7676967 0.7641661 0.7641661 0.7641661
## 11 0.0006483936 0.9997675 0.7828783 0.7641661 0.7704035 0.7641661 0.7680996 0.7641661 0.7641661 0.7664700
## 12 0.0006253550 0.9998363 0.7828783 0.7641661 0.7704035 0.7641661 0.7641661 0.7641661 0.7641661 0.7641661
## 13 0.0006435322 0.9999285 0.7828783 0.7641661 0.7704035 0.7641661 0.7641661 0.7641661 0.7704035 0.7641661
## 14 0.0005409383 0.9998501 0.7704035 0.7641661 0.7641661 0.7704035 0.7641661 0.7704035 0.7641661 0.7641661
##      TARAVAL INGLESIDE TENDERLOIN   BAYVIEW      PARK  RICHMOND
## 1  0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 2  0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 3  0.7655083 0.7656268  0.7641661 0.7641661 0.7651925 0.7648767
## 4  0.7641661 0.7704035  0.7641661 0.7641661 0.7641661 0.7641661
## 5  0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 6  0.7641661 0.7641661  0.7641661 0.7704035 0.7641661 0.7641661
## 7  0.7653438 0.7649949  0.7648640 0.7651257 0.7647331 0.7647768
## 8  0.7656179 0.7641661  0.7641661 0.7641661 0.7651340 0.7654028
## 9  0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 10 0.7641661 0.7641661  0.7656960 0.7653430 0.7641661 0.7641661
## 11 0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 12 0.7641661 0.7641661  0.7704035 0.7641661 0.7641661 0.7641661
## 13 0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
## 14 0.7641661 0.7641661  0.7641661 0.7641661 0.7641661 0.7641661
```

```
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22
##   6   6   8   4   4   8   4   4  11  13   4  11  11  13  13  12   8   4   8   8   8  11
##  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44
##  13   4   8   8   8  11   4   8   8   4   8   8   6   8  11   8   8   8   8   8  11   6
##  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66
##   4   8   8  11   8   4  11   8   4   8   8   8  11  11  11   8   6   8   4   8  11  11
##  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
##  11   4   8  11  11   8   8   8  12   8   6   8   8   8   6  11   8   8   8   8   8   6
##  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110
##  13   8  11  11   4  11  11   8   8   6   8  11  11  11  11   8   8  11  11  11  11  11
## 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
##  11  11  12  11   8  11  11  12  13  13  11   6  12  12  12   8  13  11  11  13   8  11
## 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154
##  11  11  12  11  11   8  11   8  11   4  11   4  12  11   8   6  11  11   6   8  11   8
## 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176
##   4   4   8   8  13   8   8  13   8  11  11  13   8  11  13  13  13   8   8  11   8  11
## 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
##  11   4  11  11   8  11  11   8   4  13  13  13  11  13  11  11  11   8   8  13  13  11
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##  11  11   4   6   6   8   8   8  11  12   8   8   8  12   6  11  13  13   8  12  13   8
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242
##  12   8  13  13  13  13  11   8   4   4   4   4  11  11  11  12  12  12  12   8  13  11
## 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264
##  11  11   8  11  13  13  13  11   8   8  11  13   4  13  13  13  13  11   8  11  11  11
## 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286
##  11  11   8   8   8   6   8  11  11  11   8  12   8   8   8  11  11   8  11  11  11  11
## 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308
##  11   8   8  11  11   8  11  11   8   8   8   8   8   8  13   8   8   8  13  13  11   8
## 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330
##  13  11  12   8  11   8  11  12  11  11   8   8   8   4   4  11   8   8  11  11  11  12
## 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352
##   6  11   8   8   8  10  10  10  10   3   3  10   3  10  10  10  10  10   3   3  10   3
## 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
##  10   3   1  10   9   9   3   3   3   3   9   3   3   3  10  10   3   3  10   9  10  10
## 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396
```

```
##   10    3    3    3    9    9    3    3    3    3   10   10    1    3   10    9    3    3    1   10    3    3
##  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418
##    1    3    3    3    3    1    9    9    3    3    1    1    1    1    1    9    9   10    3    1    1    3
##  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440
##    3    9   10   10    3    3    1   10    3    3    1    1    1   10   10    3    3    9    3    9   10    9
##  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460  461  462
##    9   10    1   10   10   10    9    1    3    3    3    3    3    3   10    1    1    3    3    3    3    3
##  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484
##    3    3   10   10    3   10    3    3   10   10    3   10    3    3    3    3    9   10    3    3    3    3
##  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506
##    3    9    3    3    3   10    3    3    3    1   10   10   10   10    1   10    3    1    3    3    3    1
##  507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528
##    9    3    3    3    9   10   10    9    3    3    3    3   10    1    3    3    3    3   10   10   10   10
##  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550
##    3    3    3    3    1    1    1    9    3    3    1   10    9    9    9    9   10    3    3    3    3   10
##  551  552  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572
##    9   10   10    3   10   10   10   10   10    9    3    3    1   10   10    3   10   10    3   10   10    3
##  573  574  575  576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594
##    1    1    3    1    1    3   10    3   10   10   10    9    9    3    9    1   10   10   10   10   10    3
##  595  596  597  598  599  600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616
##    1    1    3   10   10    3    3    9    3    3    9    9    3    3    3    1   10    9    3    3    9    3
##  617  618  619  620  621  622  623  624  625  626  627  628  629  630  631  632  633  634  635  636  637  638
##    3    3   10   10   10    9    9    1    9    3    3    3    3    3    1    1    3    3    3   10   10    9
##  639  640  641  642  643  644  645  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660
##    3    3    3    3   10    3    3   10    9   10    1    3   10   10    3    3   10   10    3    1   10    1
##  661  662  663  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679  680  681  682
##   10    1    3    3   10    3    9    9    9    3    9   10   10    1   10    9    3    9   10    1   10    3
##  683  684  685  686  687  688  689  690  691  692  693  694  695  696  697  698  699  700  701  702  703  704
##    3    3    9    9    3   10    9    3    1    1   10    3    1    3    1    1    3   10    3    3    1    1
##  705  706  707  708  709  710  711  712  713  714  715  716  717  718  719  720  721  722  723  724  725  726
##    7    2    2    7   14    5   14   14    5    7    7    2    2    7    2    5    5    7    2    7    2    2
##  727  728  729  730  731  732  733  734  735  736  737  738  739  740  741  742  743  744  745  746  747  748
##    7    5   14   14    5    7   14    7    7    7    7    7    7    5   14    7    7    7    7    7    7    7
##  749  750  751  752  753  754  755  756  757  758  759  760  761  762  763  764  765  766  767  768  769  770
##    7    2    2    5    7    7   14   14   14    7    7    5    5    5    7    7    7    7    7    2    5    7
##  771  772  773  774  775  776  777  778  779  780  781  782  783  784  785  786  787  788  789  790  791  792
##    7    5    2    5    5    5    2    5    7   14    2    2   14   14    5    7    5    2    2    5    5    7
```

```
##  793  794  795  796  797  798  799  800  801  802  803  804  805  806  807  808  809  810  811  812  813  814
##    7    7    7    2    5    5   14    7    7    2    5    5    2    7    7   14    5    7    7    7   14    7
##  815  816  817  818  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833  834  835  836
##    7    2    7    7    7    7   14    2    5    7    7    2    2    5    5    5    2    7    7   14   14    7
##  837  838  839  840  841  842  843  844  845  846  847  848  849  850  851  852  853  854  855  856  857  858
##   14    5    7    7    5    5    5    5   14   14   14    7    5    7    7    7    7    7    7    7    7    5
##  859  860  861  862  863  864  865  866  867  868  869  870  871  872  873  874  875  876  877  878  879  880
##    5    2    7    7    7   14    2    7    7    7    7    7    7    5    5    5    7    7    5    5    2    7
##  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899  900  901  902
##    7    7    5    5    7    7    7    7    7    7    7    7    7    7    2    2    2    7    7    7    5    7
##  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920  921  922  923  924
##    7    7    7    7    2    2    5    5    7    7   14    7   14   14    2    7    2   14    7    7    7    5
##  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943  944  945  946
##    5    5    2    2    7    5    2    2   14    7    5    7    2    7    2    7    7    5    7    5    7   14
##  947  948  949  950  951  952  953  954  955  956  957  958  959  960  961  962  963  964  965  966  967  968
##    5    7    7    7    7    5    2    5    5    7    7    7   14    5    2    2    5    7   14    7    7   14
##  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989  990
##    7    7    7    7    7    7   14    7   14   14   14    2   14    7    7    7    5    2    7   14   14   14
##  991  992  993  994  995  996  997  998  999 1000
##    7    7    7    7    5    5    5   14    7    7
##
## Within cluster sum of squares by cluster:
##  [1] 2.494665e-07 1.490678e-07 4.831163e-03 3.215987e-07 2.025029e-07 1.947368e-07 4.696217e-03 3.214841e-03
##  [9] 1.548352e-07 2.409622e-03 2.012877e-03 1.407005e-08 1.351436e-07 2.253090e-07
##  (between_SS / total_SS =  79.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
## [8] "iter"         "ifault"
```

А на цьому рисунку за допомогою функції fviz_cluster() було візуалізовано усі 14 кластерів.

```
fviz_cluster(res, data = mydata)
```