# Klasyfikacja a regresja

## Unsupervised learning

Labratorium 3

Sebastian Kuzara
KRUK S.A.
Statistical Methods Development Area
Wrocław, 2024

# K-means

Idea: podzielenie podobnych obserwacji na k grup

Cel: znalezienie wzorców „ukrytych” w danych

Algorytm:

1. Wybieramy k (na ile grup chcemy podzielić obserwacje)
2. wyznaczane zostają losowo pierwsze współrzędne centroidów grup
3. iteracyjnie optymalizujemy współrzędne centroidów aż do momentu: (1) braku zmiany współrzędnych, (2) wykonania maksymalnej liczby zadanych iteracji
   - co iterację przypisujemy punkty do grupy wg najmniejszej odległości od aktualnych centroidów
   - aktualizujemy współrzędne centroidów dla wyznaczonych grup
4. ze względu na losowe inicjowanie współrzędnych w pierwszej iteracji, przydzielenie obserwacji do konkretnych grup może być różne dla modeli zbudowanych na tych samych danych.

# K-means – przygotowanie danych


```
library(data.table)
library(ggplot2)

n_samples <- 100

set.seed(123)

data_sample1 <- data.table(
            SampleNum = "1",
            X = rnorm(n_samples),
            Y = rnorm(n_samples, sd = 2.0)
)

data_sample2 <- data.table(
            SampleNum = "2",
            X = rnorm(n_samples, mean = 2.0),
            Y = rnorm(n_samples, mean = 2.0, sd = 2.0)
)
```

# K-means – dane na wykresie

```
ggplot(data = data) +
        geom_point(aes(x = X, y= Y, color = SampleNum), size=2)
```

# K-means - model


```
set.seed(123)

model <- kmeans(x = data[,.(X, Y)], centers = 2)

data[,PredictedCluster := as.character(model$cluster)]
data[,IfCorrect := SampleNum == PredictedCluster]
```

# K-means - predykcje

```
ggplot() +
        geom_point(data = data, aes(x = X, y= Y, color = PredictedCluster, shape = IfCorrect), size=2) +
        geom_point(data = as.data.frame(model$centers), aes(x = X, y = Y), size = 4)
```

# hierarchical clustering

Idea: grupowanie obserwacji podobnych (najbliższych)

**Dwa kierunki:**

- *Agglomerative* (łączenie) - wychodzimy od klastrów jako pojedynczych obserwacji i łączymy je w grupy aż do momentu aż cały zbiór stanowi jeden klaster
- *Divisive* (podział) - odwrotny kierunek

**Algorytm (*Agglomerative*) iteracyjnie:**

- oblicz odległość między klastrami
- połącz najbliższe kastry w jeden
- powtarzaj aż zostanie jeden klaster

**Do wyboru:**

- funkcja odległości
- funkcja połączenia (*linkage*) - wyznaczamy współrzędne klastra
- poziom odcięcia lub liczba grup

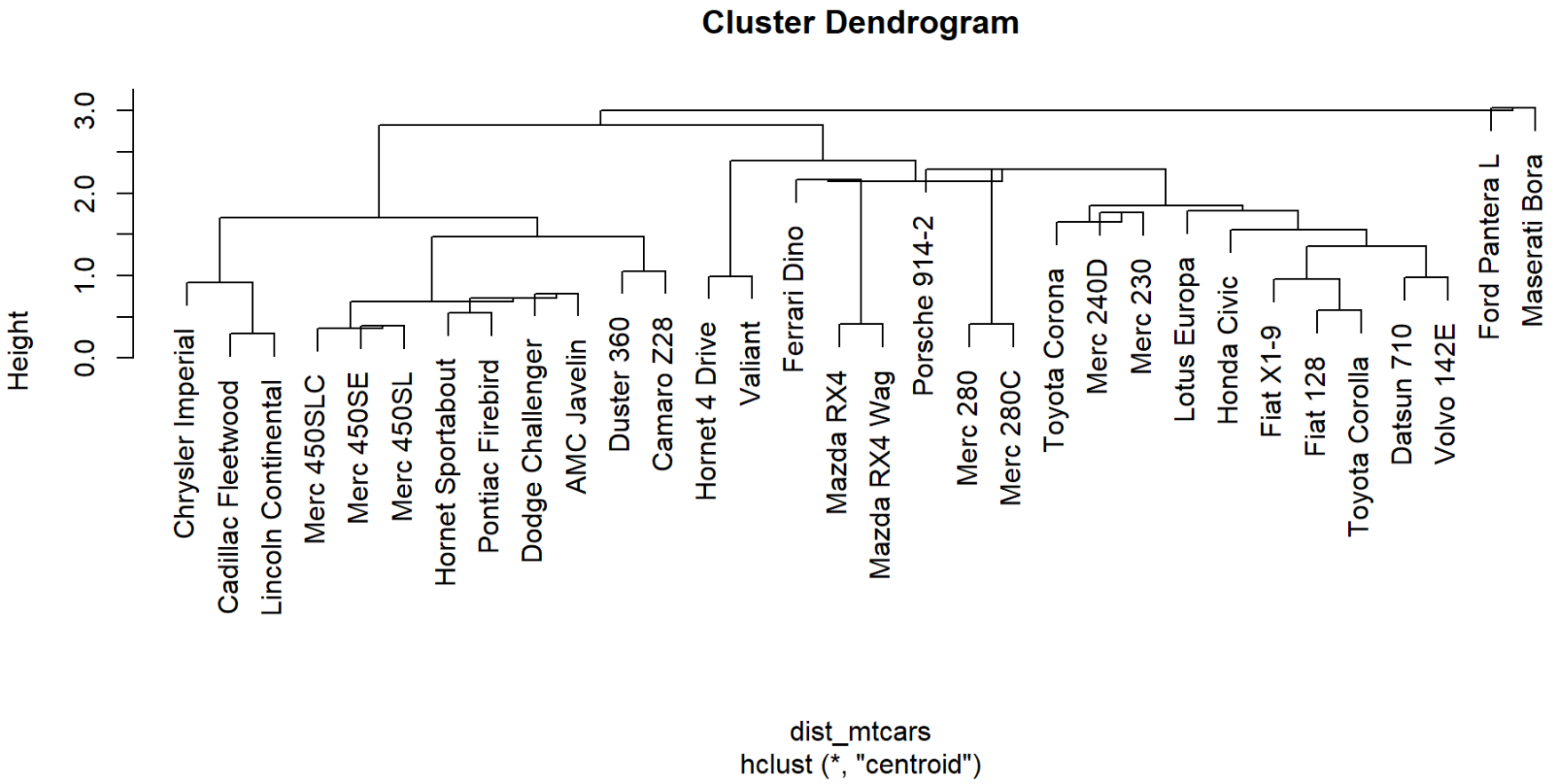
Należy przekształcić cechy do identycznej skali.

# hierarchical clustering – przygotowanie danych


```
library(data.table)
library(caret)


mtcars <- datasets::mtcars
scaler <- caret::preProcess(mtcars, method="scale")
mtcars <- predict(scaler, mtcars)
dist_mtcars <- dist(mtcars, method = "euclidean")
```

# hierarchical clustering – model

```
model <- hclust(dist_mtcars, method = "centroid")


# Dendogram

plot(model)
```



# hierarchical clustering – grupowanie

```
predicted_clusters <- cutree(model, k=8)


# argument k-liczba grup, h wysokość odcięcia
predicted_clusters_dt <- data.table(
            Car = names(predicted_clusters),
            Cluster = predicted_clusters
)


setorder(predicted_clusters_dt, Cluster)
print(predicted_clusters_dt)
```

```
> print(predicted_clusters_dt)
                    Car Cluster
                 <char>   <int>
 1:          Mazda RX4       1
 2:      Mazda RX4 Wag       1
 3:      Ferrari Dino        1
 4:         Datsun 710       2
 5:         Merc 240D        2
 6:          Merc 230        2
 7:          Fiat 128        2
 8:        Honda Civic       2
 9:     Toyota Corolla       2
10:     Toyota Corona        2
11:         Fiat X1-9        2
12:      Lotus Europa        2
13:        Volvo 142E        2
14:    Hornet 4 Drive        3
15:           Valiant        3
16: Hornet Sportabout        4
17:        Duster 360        4
18:         Merc 450SE       4
19:         Merc 450SL       4
20:        Merc 450SLC       4
21: Cadillac Fleetwood       4
22: Lincoln Continental      4
23:  Chrysler Imperial       4
24:  Dodge Challenger        4
25:       AMC Javelin        4
26:        Camaro Z28        4
27:  Pontiac Firebird        4
28:          Merc 280        5
29:         Merc 280C        5
30:      Porsche 914-2       6
31:      Ford Pantera L      7
32:      Maserati Bora       8
                    Car Cluster
```

# k nearest neighbours

Idea: Podobne obserwacje położone są w niedalekiej odległości od siebie w zadanej przestrzeni

Algorytm:

1. jeśli zmienne objaśniające opisane są na różnych skalach, należy je odpowiednio przekształcić (ujednolicić)
2. wybieramy k (ilu sąsiadów chcemy dobrać)
3. Iteracyjnie: dla każdej obserwacji z badanej próbki:
   - obliczamy odległość między tą obserwacją a obserwacjami w próbie referencyjnej
   - wybieramy k sąsiadów z próby referencyjnej, które charakteryzują się najmniejszymi odległościami
4. klasyfikacja: każdej obserwacji ze zbioru modelowanego przypisujemy najczęściej występującą klasę (*label*) występującą wśród jej sąsiadów
5. regresja: dla każdej obserwacji modelowanej obliczamy predykcję zmiennej objaśnianej (dostępnej w zbiorze referencyjnym) za pomocą funkcji agregującej (np. średniej)

# kNN – przygotowanie danych

```
library(data.table)
library(ggplot2)
library(caret)


# implementacja knn i przeskalowanie cech
n_ref <- 3000
n_sample <- 100
set.seed(123)
reference <- data.table(
              X1 = rnorm(n_ref, mean = 5, sd = 1),
              X2 = runif(n_ref))
reference_target <- data.table(
              Y = sample(c("A", "B"), n_ref, replace = TRUE))
data_sample <- data.table(
              X1 = rnorm(n_sample, mean = 4.2, sd = 1),
              X2 = runif(n_sample, min = 0.3, max = 0.85))
```

# kNN - przeskalowanie zmiennych

```
scaler <- caret::preProcess(reference, method = "range") # min-max scaler (normalizacja)
reference <- predict(scaler, reference)
data_sample <- predict(scaler, data_sample)
```

```
summary(reference)
```

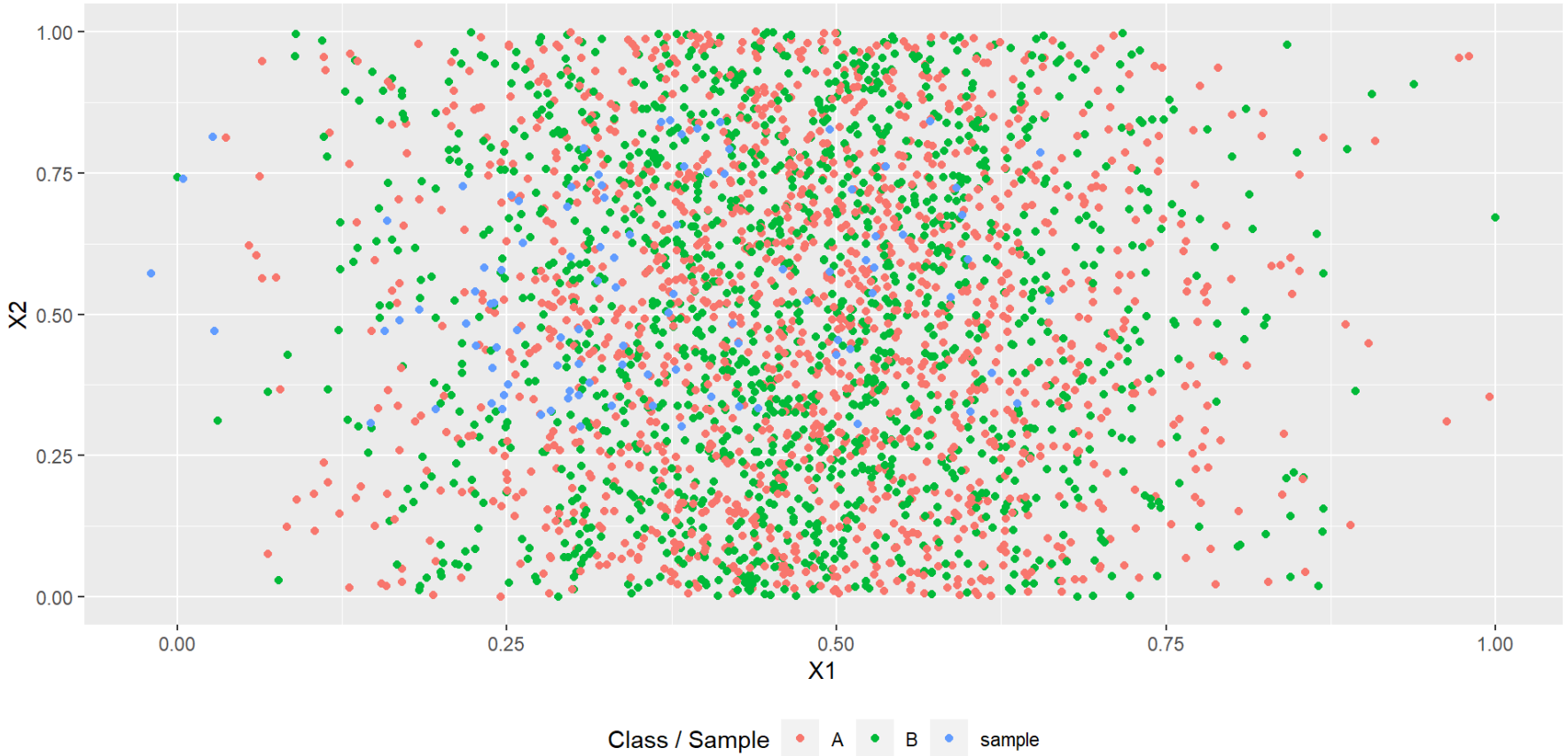
```
> summary(reference)
       X1                X2
 Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.3706   1st Qu.:0.2556
 Median :0.4705   Median :0.4934
 Mean   :0.4731   Mean   :0.4942
 3rd Qu.:0.5768   3rd Qu.:0.7368
 Max.   :1.0000   Max.   :1.0000
```

```
summary(data_sample)
```

```
> summary(data_sample)
       X1                 X2
 Min.   :-0.01965   Min.   :0.3005
 1st Qu.: 0.25655   1st Qu.:0.4037
 Median : 0.33209   Median :0.5274
 Mean   : 0.35544   Mean   :0.5448
 3rd Qu.: 0.44546   3rd Qu.:0.6931
 Max.   : 0.66189   Max.   :0.8432
>
```

# kNN – wykres

```
ggplot() +
        geom_point(data = reference, aes(x = X1, y = X2, color = reference_target$Y)) +
        geom_point(data = data_sample, aes(x = X1, y = X2, color="sample")) +
        guides(color = guide_legend(title = "Class / Sample")) + theme(legend.position = "bottom")
```



# kNN – model

```
fitted <- caret::knn3Train(
            train = reference,
            test = data_sample,
            cl = reference_target$Y,
            k = 11)


table(fitted)
```

```
> table(fitted)
fitted
 A  B
45 55
```

# Self Organizing Maps (1)


Idea: przekształcenie danych o dużej wymiarowości do dwóch wymiarów

Algorytm:

- zainicjowanie wag (losowe)
- Iteracyjnie (epoki, jednokrotne przejście przez wszystkie przykłady):
  - wyznaczenie BMU (Best Matched Unit) = wygrywającego wektora wag, czyli wskazanie który neuron posiada wagi najbliższe wektorowi wejściowemu
  - update wygrywającego wektora:

$$w_{j,k} = w_{j,k} + learnig\_rate(t) \times (x_i - w_{j,k})$$

  - update wag sąsiednich do BMU:

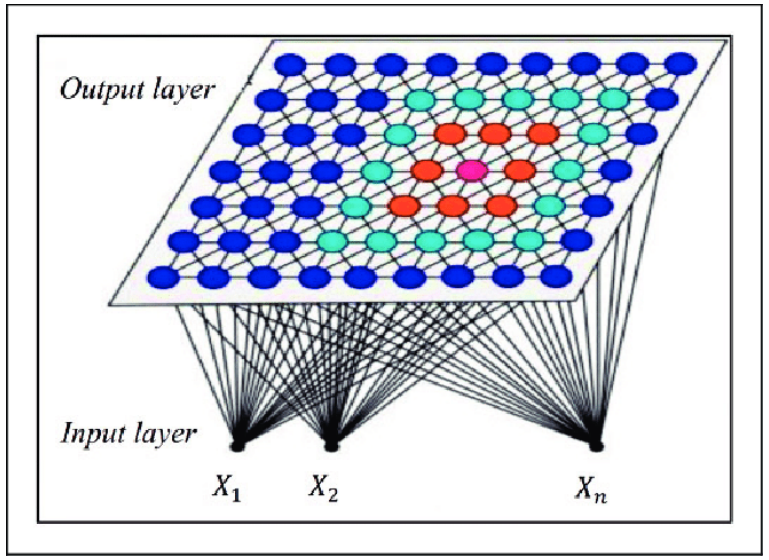
$$w_{j,k} = w_{j,k} + neighborhood(u, v, t) \times learnig\_rate(t) \times (x_i - w_{j,k})$$

*w* - wektor wag
*x* - dane wejściowe (jedna obserwacja)
*learining_rate* - funkcja parametru uczenia
*neighborhood* - funkcja sąsiedztwa (dystansu) od BMU

# Self Organizing Maps (2)

Architektura modelu:

- 2-warstwowa sieć neuronowa bez użycia propagacji wstecznej oraz bez funkcji aktywacji
- model wykorzystuje uczenie poprzez rywalizację (*competetive learning*), na który składają się trzy procesy:
  - rywalizacja (*competetion)* między neurnami w celu wybrania BMU
  - współpraca (*cooperation*) - uwzględnienie sąsiedztwa BMU
  - adaptacja (*adaptation*) - update wag BMU oraz neuronów sąsiednich




źródło: https://www.researchgate.net/figure/Structural-model-of-self-organizing-map-neural-network-Figure-2-Experimental-benchmark_fig1_329337931

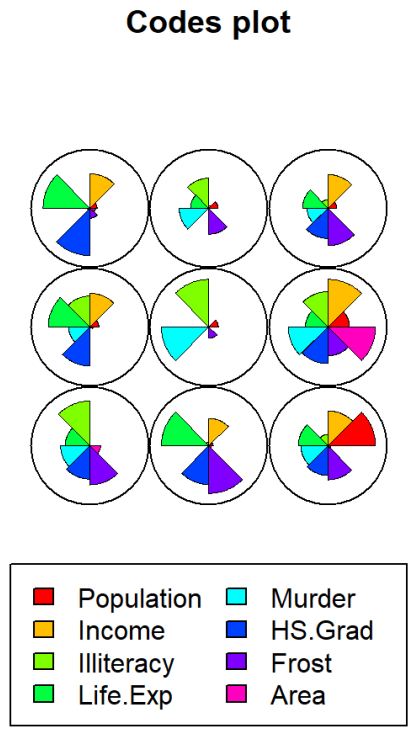
# SOM – przygotowanie danych

```
library(data.table)

library(caret)

library(kohonen) # implementacja som


states_raw <- data.frame(datasets::state.x77)

state_names <- rownames(states_raw)

scaler <- caret::preProcess(x = states_raw, method = "scale")

states <- predict(scaler, states_raw)

states <- as.matrix(states)

summary(states_raw)
```

```
> summary(states_raw)
   Population        Income       Illiteracy      Life.Exp         Murder        HS.Grad         Frost             Area
 Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96   Min.   : 1.400   Min.   :37.80   Min.   :  0.00   Min.   :  1049
 1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12   1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25   1st Qu.: 36985
 Median : 2838   Median :4519   Median :0.950   Median :70.67   Median : 6.850   Median :53.25   Median :114.50   Median : 54277
 Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88   Mean   : 7.378   Mean   :53.11   Mean   :104.46   Mean   : 70736
 3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89   3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75   3rd Qu.: 81163
 Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60   Max.   :15.100   Max.   :67.30   Max.   :188.00   Max.   :566432
>
```

# SOM – model

```
som_grid <- kohonen::somgrid(
            xdim=3,
            ydim=3,
             topo = "rectangular")
set.seed(123)
model <- som(
            X = states,
            grid = som_grid,
            alpha = c(0.05, 0.01),
            radius = 1)


plot(model, type="codes", palette.name = rainbow)
```



# SOM – predykcje

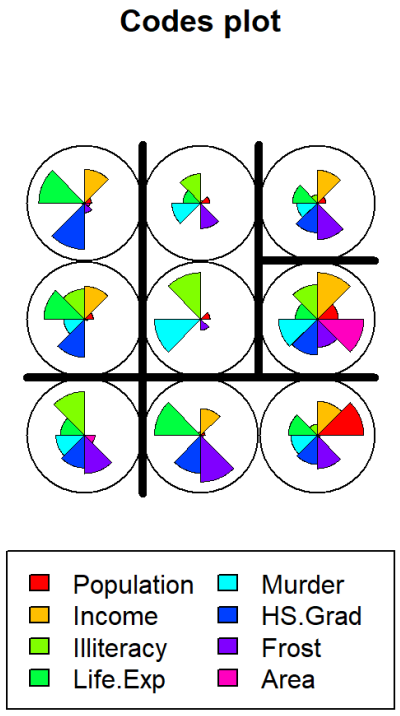
```
prediction <- data.table(
              StateName = state_names,
              Group = model$unit.classif )


setorder(prediction, Group)


print(prediction)
```

```
> print(prediction)
          StateName Group
              <char> <num>
 1:      New Mexico     1
 2:        Colorado     2
 3:     Connecticut     2
 4:           Idaho     2
 5:            Iowa     2
 6:          Kansas     2
 7:           Maine     2
 8:   Massachusetts     2
 9:       Minnesota     2
10:         Montana     2
11:        Nebraska     2
12:   New Hampshire     2
13:    North Dakota     2
14:    Rhode Island     2
15:    South Dakota     2
```

KRUK

# SOM – klastrowanie

```
# Utworzoną mapę możemy podzielić na klastry za pomocą klastrowania hierarchicznego

levels <- 3

map_clusters <- cutree(hclust(dist(model$codes[[1]])), k = levels)


plot(model, type="codes", palette.name = rainbow)

add.cluster.boundaries(model, map_clusters)
```



# SOM – predykcje na nowych danych


```
new_state_raw <- data.table(
        Population = 10000,
        Income = 4000,
        Illiteracy = 1.0,
        Life.Exp = 69.23,
        Murder = 3.12,
        HS.Grad = 53.5,
        Frost = 7.0,
        Area = 40000)


new_state <- predict(scaler, new_state_raw)
new_state <- as.matrix(new_state)
new_state_pred <- predict(model, new_state) # simillar states:
prediction[Group==new_state_pred$unit.classif,]
```

```
> prediction[Group==new_state_pred$unit.classif,]
       StateName Group
          <char> <num>
1:    California     3
2:      Illinois     3
3:      Michigan     3
4:      New York     3
5:          Ohio     3
6: Pennsylvania     3
```

# Miary błędów - regresja


Przykładowe miary:

- MAE
- RMSE
- MSE
- MAPE
- …

https://mlr.mlr-org.com/articles/tutorial/measures.html#regression-1

# Miary błędów - klasyfikacja


Macierz błędu:

| | Predicted Good | Predicted Bad | |
|---|---|---|---|
| Actual Good | TP | FN | = P |
| Actual Bad | FP | TN | = N |

TOTAL - total number of observation

- Accuracy (ACC) = TP + TN / TOTAL
- sensitivity or recall = true positive rate (TPR) = TP/P = TP/(TP + FN)
- specificity = true negative rate (TNR) = TN/N = TN/(FP + TN)
- AUC (Area Under Curve) - pole pod krzywą ROC (funkcja FPR(TPR))

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

- logloss
- Gini index
- ...

https://mlr.mlr-org.com/articles/tutorial/measures.html#classification-1

# Miary błędów - regresja


```
# niezbedne wczesniej zainstalowane pakiety data.table oraz caret
source("lab03/lab3-data-preparation.R")
```