LOENG 8. Klassifitseerimise mudelid

8.1. Lähima naabri meetod (*Nearest Neighbor Classification*)

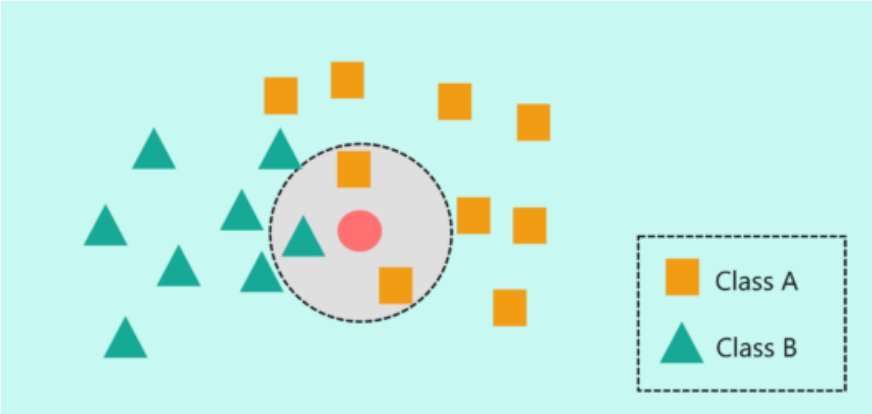
- **Klassifitseerimisprobleemide korral prognoositav tunnus *Y* on kategoriaalne (nominaal- või järjestustunnus), kuid see võib olla arvudega kodeeritud (nt 1 = "jah", 0 = "ei").**
- **Ülesanne: „mudeli" või „reegli" väljatöötamine andmestiku objektide *Y*-*i* kategooriatele klassifitseerimiseks sõltuvalt argumentide $X_1, X_2, \dots, X_p$ väärtustest.**
- **Argumentide $X_1, X_2, \dots, X_p$ tüüp: arvuline, mittearvulistest moodustatakse *dummy variables*.**
- **Klassifitseerimisreegel: moodustada klasterid *k-lähima naabri meetodil* valitud kauguse alusel ning klassifitseerida objekte vastavalt suurema sagedusega klassi väärtusele klasteris ehk kui prognoositav *Y* võiks olla klassist C_j ja k_j on klassist C_j objektide arv klasteri *k* objektide seas,**

  ***j* =1,…,*m*, siis** $\sum_{j=1}^{m} k_j = k$ **ja *Y*-i prognoos klasteris on** $k_l = \max_j k_j$

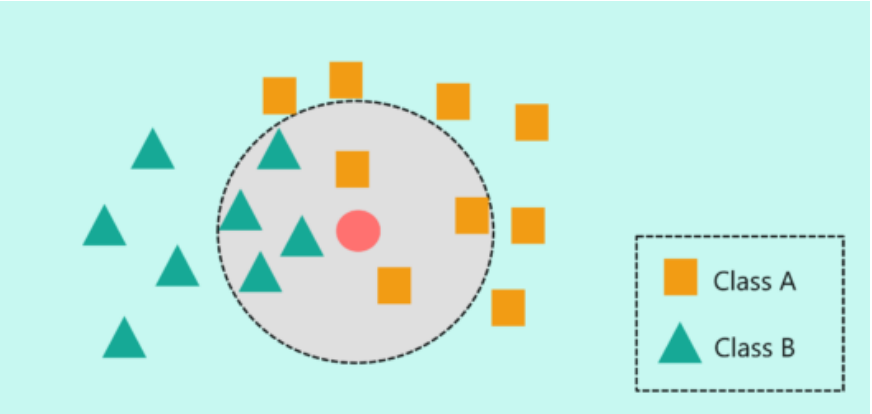
- **Lähima naabri meetod on rakendatav ka regressiooni probleemide korral, *Y*-i prognoos klasteris määratakse sel juhul klasteri keskmisena**

$$\hat{y} = \frac{1}{k}\sum_{i \in \mathbb{N}} y_i$$

k-lähima naabri meetod



k = 3



k = 7

KNN Algorithm Pseudocode:

- Calculate D(x, xi), where 'i' =1, 2, ….., n and 'D' is the distance (Euclidean measure) between the data points.
- The calculated (Euclidean) distances must be arranged in ascending order.
- Initialize k and take the first k distances from the sorted list.
- Figure out the k points for the respective k distances.
- Calculate ki, which indicates the number of data points belonging to the ith class among k points i.e. k ≥ 0 If ki >kj ∀ i ≠ j; put x in class i.

https://www.edureka.co/blog/knn-algorithm-in-r/

- ***k-lähima naabri meetodil*** **klassifitseerimisel probleemiks võib olla võrdsed klasside väärtuste sagedused klasteris, nt binaarse klassifikatsiooni korral Y = 1 ja Y = 0 naabrite arvud on võrdsed.**
- **Sel põhjusel kasutatakse *kaalutud lähima naabri klassifikaatorit*, mille korral lähimatele objektidele antakse klassifitseerimisel suurem kaal.**
- ***Kaalutud k-NN klassifitseerimise algoritm*:**

1. **Olgu treenimisandmed** $T = \{(\boldsymbol{x_i}, y_i), i = 1, \dots, n_T\}$
2. **Leida objekti** $\boldsymbol{x}$ $(k+1)$ **lähima naabrit kauguste** $d(\boldsymbol{x}, \boldsymbol{x_i})$ **alusel.**
3. **Kaugus *k* +1 naabrist kasutatakse *k* teiste kauguste standardiseerimiseks**

$$D_{(i)} = D\left(\boldsymbol{x}, \boldsymbol{x}_{(\boldsymbol{i})}\right) = \frac{d(\boldsymbol{x}, \boldsymbol{x}_{(\boldsymbol{i})})}{d(\boldsymbol{x}, \boldsymbol{x}_{(\boldsymbol{k+1})})}$$

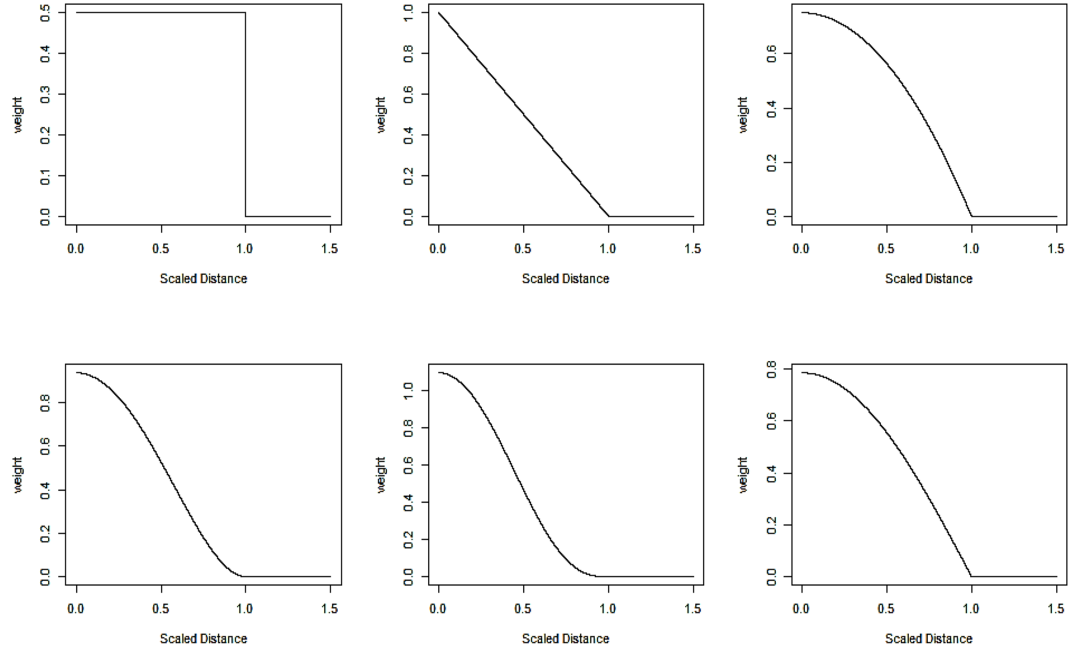
4. **Kaalude arvutamiseks kasutatakse sobivat kauguste tuuma (*kernel*) funktsiooni** $w_{(i)} = K(D_{(i)})$
5. **Prognoositav klass arvutatakse valemiga**

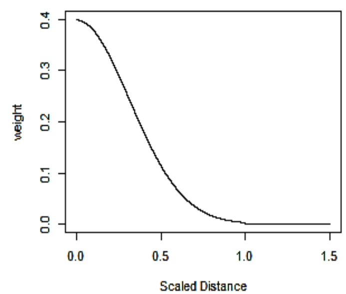
$$\hat{y} = \max_j \sum_{i=1}^{k} w_{(i)} I\left(y_{(i)} = Cj\right), \text{kus} \quad I\left(y_{(i)} = Cj\right) = \begin{cases} 1 & if\ \ y_{(i)}\ is\ class\ Cj \\ 0 & if\ y_{(i)}\ is\ NOT\ class\ Cj \end{cases}$$

6. **Klassi kuulumise tõenäosus edasiseks prognoosimiseks hinnatakse valemiga**

$$\hat{P}(y = Cj|\boldsymbol{x}) = \frac{\sum_{i=1}^{k} w_{(i)} I\left(y_{(i)} = Cj\right)}{\sum_{i=1}^{k} w_{(i)}}, \qquad j = 1, \dots, m$$

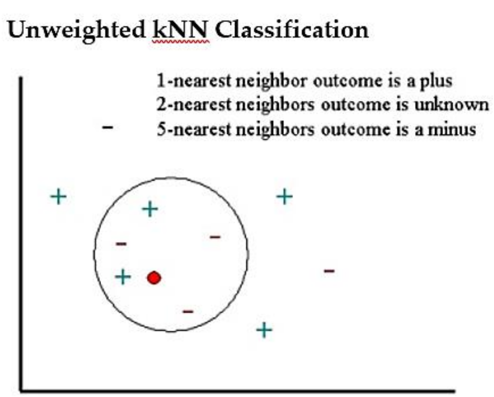
❑ *Kernelid*

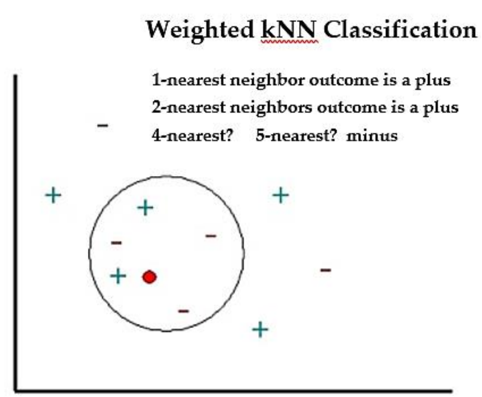




Ülemisest vaskust ja vasakult paremale $w_{(i)} = K(D_{(i)})$:
`rectangular`
`triangular`
`epanechnikov`
`biweight` or quartic
`triweight` or tricube
`cosine`
`gaussian` ← never 0!

❑ Kaalutud ja kaalumata klassifikaatorite võrdlus





k-lähima naabri meetod R-is

NÄIDE: **kasutame andmestiku** olive **paketist** dslabs

```
library(dslabs)
data(olive)
olive <- olive[,-1] # jätame regiooni välja
levels(olive$area) <- make.names(levels(olive$area))
str(olive)
```

```
'data.frame':   572 obs. of  10 variables:
 $ region     : Factor w/ 3 levels "Northern Italy",..: 3 3 3 3 3 3 3 3
...
 $ area       : Factor w/ 9 levels "Calabria","Coast-Sardinia",..: 5 5 5
 5 5 5 5 5 ...
 $ palmitic   : num  10.75 10.88 9.11 9.66 10.51 ...
 $ palmitoleic: num  0.75 0.73 0.54 0.57 0.67 0.49 0.66 0.61 0.6 0.55 ..
 $ stearic    : num  2.26 2.24 2.46 2.4 2.59 2.68 2.64 2.35 2.39 2.13 ..
 $ oleic      : num  78.2 77.1 81.1 79.5 77.7 ...
 $ linoleic   : num  6.72 7.81 5.49 6.19 6.72 6.78 6.18 7.34 7.09 6.33 .
 $ linolenic  : num  0.36 0.31 0.31 0.5 0.5 0.51 0.49 0.39 0.46 0.26 ...
 $ arachidic  : num  0.6 0.61 0.63 0.78 0.8 0.7 0.56 0.64 0.83 0.52 ...
 $ eicosenoic : num  0.29 0.29 0.29 0.35 0.46 0.44 0.29 0.35 0.33 0.3 ..
```

- **Kuna meetod põhineb kaugustel, arvulised tunnused nõuvad skaleerimist, mittearvulistest → *dummy variables***

```
olive[,2:9] = scale(olive [,2:9])
RNGkind(sample.kind = "Rounding")
set.seed(123)
sam <- sample(1:nrow(olive),
   floor(nrow(olive)*0.2))
olive.train <- olive[-sam,]
olive.test <- olive[sam,]
```

- **Kasutame R-i funktsiooni train.kknn paketist kknn**

```
library(kknn)
```

Usage

```
train.kknn(formula, data, kmax = 11, ks = NULL, distance = 2, kernel = "optimal",
        ykernel = NULL, scale = TRUE, contrasts = c('unordered' = "contr.dummy",
        ordered = "contr.ordinal"), ...)
cv.kknn(formula, data, kcv = 10, ...)
```

Arguments

`formula`	A formula object.
`data`	Matrix or data frame.
`kmax`	Maximum number of k, if `ks` is not specified.
`ks`	A vector specifying values of k. If not null, this takes precedence over `kmax`.
`distance`	Parameter of Minkowski distance. https://en.wikipedia.org/wiki/Minkowski_distance
`kernel`	Kernel to use. Possible choices are "rectangular" (which is standard unweighted knn), "triangular", "epanechnikov" (or beta(2,2)), "biweight" (or beta(3,3)), "triweight" (or beta(4,4)), "cos", "inv", "gaussian" and "optimal".
`ykernel`	Window width of an y-kernel, especially for prediction of ordinal classes.
`scale`	logical, scale variable to have equal sd.
`contrasts`	A vector containing the 'unordered' and 'ordered' contrasts to use.
`...`	Further arguments passed to or from other methods.
`kcv`	Number of partitions for k-fold cross validation.

Details

`train.kknn` performs leave-one-out crossvalidation and is computatioanlly very efficient. `cv.kknn` performs k-fold crossvalidation and is generally slower and does not yet contain the test of different models yet.

k-lähima naabri meetod R-is

Kasutame R-i funktsiooni train.kknn **paketist** kknn

```
library(kknn)
RNGkind(sample.kind = "Rounding")
set.seed(123)
olive.knn <- train.kknn(area~.,data=olive.train, kmax=10,kernel=c("triangular",
"rectangular", "epanechnikov", "optimal"))
olive.knn
```

```
Call:
train.kknn(formula = area ~ ., data = olive.train, kmax = 10,     kernel = c("triangular", "rectangular",
 "epanechnikov", "optimal"))

Type of response variable: nominal
Minimal misclassification: 0.03056769
Best kernel: triangular
Best k: 6
```

Segadusmaatriks testandmetel

```
library(caret)
confusionMatrix(predict(olive.knn,olive.test),olive.test$area,mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      1            0      0            0
  Coast.Sardinia        0              7            0               0            0      0            0      0            0
  East.Liguria          0              0            8               0            0      0            0      0            0
  Inland.Sardinia       0              1            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      2            3      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            0               0            0      0            0     11            0
  West.Liguria          0              0            1               0            0      0            0      0           11

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.889, 0.9804)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9345
```

k-lähima naabri meetodi parameetrite tuunimine paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5 ,search = "random",classProbs =
TRUE,summaryFunction = multiClassSummary)
tuneGrid <- expand.grid(kmax = 3:10,distance = 1:4,kernel =
c('gaussian','triangular','rectangular','epanechnikov','optimal'))
RNGkind(sample.kind = "Rounding")
set.seed(123)
kknn_fit <- train(area~.,olive.train, method = 'kknn',trControl = train.control,tuneGrid =
tuneGrid,metric = "Kappa")
kknn_fit
head(kknn_fit$results[order(-kknn_fit$results$Kappa),],20)
```

```
Kappa was used to select the optimal model using the largest value.
The final values used for the model were kmax = 7, distance = 3 and kernel = epanechnikov.
> head(kknn_fit$results[order(-kknn_fit$results$Kappa),],7)
```

Segadusmaatriks testandmetel

```
library(caret)
confusionMatrix(predict(kknn_fit,olive.test),olive.test$area,mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0              0            0      1            1      0            0
  Coast.Sardinia        0              6            0              0            0      0            0      0            0
  East.Liguria          0              0            7              0            0      0            0      0            1
  Inland.Sardinia       0              2            0             10            0      0            0      0            0
  North.Apulia          0              0            0              0            3      0            0      0            0
  Sicily                0              0            0              0            0      2            2      0            0
  South.Apulia          0              0            0              0            0      0           41      0            0
  Umbria                0              0            0              0            0      0            0     11            0
  West.Liguria          0              0            2              0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9211
                 95% CI : (0.8554, 0.9633)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9016
```

8.2. Naïve Bayes Classification

- **NB klassifitseerimine põhineb *Bayesi reeglil***
- **Klass määratakse suurima klassi kuulumise tõenäosusega**

$$P(C_k|\boldsymbol{x}) = \max_{1\le j\le m} P(C_j|\boldsymbol{x})$$

- **Kus klassi tinglik tõenäosus määratakse Bayesi valemiga**

$$P(C_j|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_j)P(C_j)}{P(\boldsymbol{X}=\boldsymbol{x})} = \frac{P(X_1=x_1, X_2=x_2, \dots, X_p=x_p|C_j)P(C_j)}{P(X_1=x_1, X_2=x_2, \dots, X_p=x_p)}$$

- **Maksimeerimist nõuab ainult lugeja, kus $P(C_j)$ määratakse klasside osakaaludega treenimisandmestikus**
- **$P(\boldsymbol{x}|C_j)$ hinnangu saamiseks eeldatakse litsustamise mõttes, et $X_1, X_2, \dots, X_p$ on sõltumatud, mis praktiliselt kunagi ei ole täidetud, siit pärinebki nimetus „naiivne“**

$$P(\boldsymbol{x}|C_j) = P(\boldsymbol{X}=\boldsymbol{x}|C_j) = P(X_1=x_1, X_2=x_2, \dots, X_p=x_p|C_j)$$

$$= P(X_1=x_1|C_j)P(X_2=x_2|C_j)\dots P(X_p=x_p|C_j) = \prod_{i=1}^{p} P(X_i=x_i|C_j)$$

$P(X_i = x_i|C_j)$ hindamine

- **Kategoriaalne X_i *on tasemel l* → kasutame treenimisandmed**

$$\hat{P}(X_i = xi|C_j) = \frac{\hat{P}(X_i{=}xi \cap Y{=}Cj)}{\hat{P}(Y{=}Cj)} = \frac{\#(X_i{=}xi \cap Y{=}Cj)}{\#(Y{=}Cj)}$$

  - **Kuna lugeja võiks olla 0, ehk võivad puududa paarid $X_i = xi \cap Y = Cj$, siis lisatakse virtuaalsed paarid ehk rakendatakse *Laplace Smoothing***

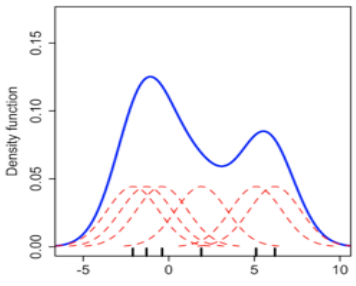
$$\hat{P}(X_i = xi \cap Y = Cj) = \frac{\#(X_i = xi \cap\ Y = Cj) + 1}{\#(Y = Cj) + \#(levels\ for\ X_i)}\ \ or\ \ \frac{\#(X_i = xi\ \cap\ Y = Cj) + k}{\#(Y = Cj) + k \cdot \#(levels\ for\ X_i)}$$

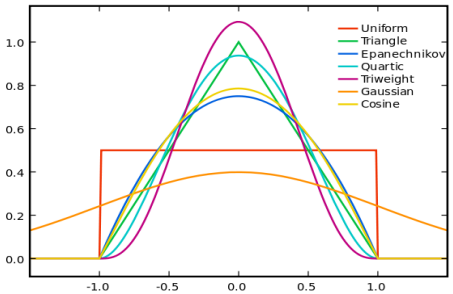
- **Pidev X_i → kasutame normaalsuse eeldust või, kui eeldus ei ole täidetud rakendame Box-Cox/Yeo-Johnson teisendusi ja lähendame tõenäosust normaaljaotuse tihedusega**

$$\hat{P}(X_i = x_i|C_j) = \hat{f}(x|C_j) = \frac{1}{\sqrt{2\pi\hat{\sigma}_{ij}^2}} exp\left(-\frac{(x_i - \hat{\mu}_{ij})^2}{2\hat{\sigma}_{ij}^2}\right)$$

- **Praktikal tiheduse asemel rakendatakse *kernel smoother***

$$\hat{f}(x|C_j) = \frac{1}{nh}\sum_{i=1}^{n_j} K\left(\frac{x - x_i}{h}\right)\ \ where\ n_j = \#\ of\ obs.with\ Y = level\ j\ \&\ h > 0$$





Kernel Functions, $K(\cdot)$

bandwidth h > 0 (adjust) kontrollib kerneli laiust. Suuremad h annavad siledama tiheduse hinnangu, väikesed h annavad rohkem müra hinnangule.

Naïve Bayesi klassifitseerimise näide:

Lähtume valemist

$$\hat{P}(\boldsymbol{x}|C_k) = \max_{1\le j\le m} \hat{P}(\boldsymbol{x}|C_j) \propto \max_{1\le j\le m} \prod_{i=1}^{p} \hat{P}(X_i = x_i|C_j) = \max_{1\le j\le m} \hat{P}(X_1 = x_1|C_j)\hat{P}(X_2 = x_2|C_j)\cdots\hat{P}(Xp = xp\,|Cj)$$

Fruit	Long?	Sweet?	Green?	Total
Banana	400	350	50	500
Apple	0	200	150	300
Other	100	150	50	200

- **Eeldusel $P(C_j) = 1/3$ klassifitseeri**

$\boldsymbol{x}_1 = (Long? = Yes, Sweet? = No, Green? = No)$

$\hat{P}(\boldsymbol{x}_1|Banana) \approx$

$0.8 \times 0.3 \times 0.9 = .216$ (not Laplace corrected)

$0.786 \times 0.311 \times 0.893 = 0.218$ (Laplace corrected)

$\hat{P}(\boldsymbol{x}_1|Apple) \approx$

$0 \times 0.333 \times 0.5 = 0$ (not Laplace corrected)

$0.016 \times 0.349 \times 0.508 = 0.003$ (Laplace corrected)

$\hat{P}(\boldsymbol{x}_1|Other) \approx$

$0.5 \times 0.25 \times 0.75 = .094$ (not Laplace corrected)

$0.488 \times 0.279 \times 0.744 = 0.101$ (Laplace corrected)

Thus $\boldsymbol{x}_1$ is classified as a Banana

$\hat{P}(Yes\|Class)$	$\hat{P}(Yes\|Class)$Laplace ($k = 5$)
$\hat{P}(Long\|Banana) = \frac{400}{500} = 0.8$	$\frac{400+5}{500+5*3} = 0.786$
$\hat{P}(Sweet\|Banana) = \frac{350}{500} = 0.7$	$\frac{350+5}{500+5*3} = 0.689$
$\hat{P}(Green\|Banana) = \frac{50}{500} = 0.1$	$\frac{50+5}{500+5*3} = .107$
$\hat{P}(Long\|Apple) = \frac{0}{300} = 0$	$\frac{0+5}{300+5*3} = 0.016$
$\hat{P}(Sweet\|Apple) = \frac{200}{300} = 0.667$	$\frac{200+5}{300+5*3} = 0.651$
$\hat{P}(Green\|Apple) = \frac{150}{300} = 0.5$	$\frac{150+5}{300+5*3} = 0.492$
$\hat{P}(Long\|Other) = \frac{100}{200} = 0.5$	$\frac{100+5}{200+5*3} = 0.488$
$\hat{P}(Sweet\|Other) = \frac{150}{200} = 0.75$	$\frac{150+5}{200+5*3} = 0.721$
$\hat{P}(Green\|Other) = \frac{50}{200} = 0.25$	$\frac{50+5}{200+5*3} = 0.256$

❑ Naïve Bayesi klassifitseerimine R-is

Kasutame R-i funktsiooni NaiveBayes paketist klaR

Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Usage

```
## S3 method for class 'formula'
NaiveBayes(formula, data, ..., subset, na.action = na.pass)
## Default S3 method:
NaiveBayes(x, grouping, prior, usekernel = FALSE, fL = 0, ...)
```

Kernels	Bandwidth selectors
Gaussian	nrd0 (Silverman's rule-of-thumb)
Epanechnikov	nrd (variation of the rule-of-thumb)
Rectangular	ucv (unbiased cross-validation)
Triangular	bcv (biased cross-validation)
Biweight	SJ (Sheather & Jones method)
Cosine	
Optcosine	

Arguments

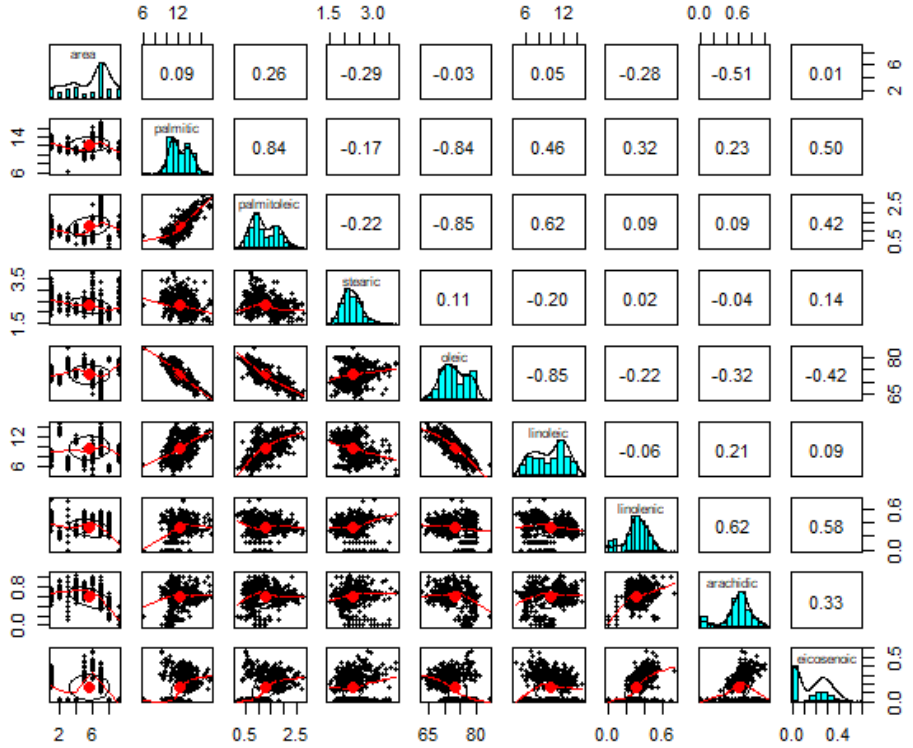
`x`	a numeric matrix, or a data frame of categorical and/or numeric variables.
`grouping`	class vector (a factor).
`formula`	a formula of the form `class ~ x1 + x2 + ...`. Interactions are not allowed.
`data`	a data frame of predictors (caegorical and/or numeric).
`prior`	the prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.
`usekernel`	if `TRUE` a kernel density estimate (`density`) is used for denstity estimation. If `FALSE` a normal density is estimated.
`fL`	Factor for Laplace correction, default factor is 0, i.e. no correction.
`...`	arguments passed to `density`.
`subset`	for data given in a data frame, an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
`na.action`	a function to specify the action to be taken if `NA`s are found. The default action is not to count them for the computation of the probability factors. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)

```
library(klaR)
```

- ❑ **Arvulised tunnused nõuvad skaleerimist, mittearvulistest → *dummy variables***

NÄIDE: kasutame andmestiku olive paketist dslabs

```
library(psych)
pairs.panels(olive[,-1])
```



- **Mõned argumendid ei vasta normaalsuse eeldusele**

```
olive.NB =
NaiveBayes(area~.,data=olive.train)
ypred =
predict(olive.NB,newdata=olive.test)
```

```
Warning messages:
1: In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 5
2: In FUN(X[[i]], ...) :
  Numerical 0 probability for all classes with observation 31
```

```
confusionMatrix(predict(olive.NB,olive.test)$class,olive.test$area,mode = "everything")
```

```
Confusion Matrix and Statistics

                 Reference
Prediction        Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria              14              0            0               0            0      1           2      0            0
  Coast.Sardinia         0              8            0               0            0      0           0      0            0
  East.Liguria           0              0            6               0            0      0           0      0            1
  Inland.Sardinia        0              0            0              10            0      0           0      0            0
  North.Apulia           0              0            0               0            3      0           0      0            0
  Sicily                 1              0            0               0            0      2           2      0            0
  South.Apulia           0              0            0               0            0      0          40      0            0
  Umbria                 0              0            1               0            0      0           0     11            0
  West.Liguria           0              0            2               0            0      0           0      0           10

Overall Statistics

               Accuracy : 0.9123
                 95% CI : (0.8446, 0.9571)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8911
```

Naïve Bayesi parameetrite tuunimine paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5 ,search = "random",classProbs =
TRUE,summaryFunction = multiClassSummary)
tuneGrid <- expand.grid(usekernel = c(FALSE,TRUE),fL = 0:5,adjust = 0:5)
RNGkind(sample.kind = "Rounding")
set.seed(123)
nb_fit <- train(area~.,olive.train, method = 'nb',trControl = train.control,tuneGrid =
tuneGrid,metric = "Kappa")
nb_fit
head(nb_fit$results[order(-nb_fit$results$Kappa),],5)
```

```
Kappa was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = FALSE and adjust = 0.
```

Segadusmaatriks testandmetel

```
library(caret)
confusionMatrix(predict(nb_fit,olive.test),olive.test$area,mode = "everything")
```

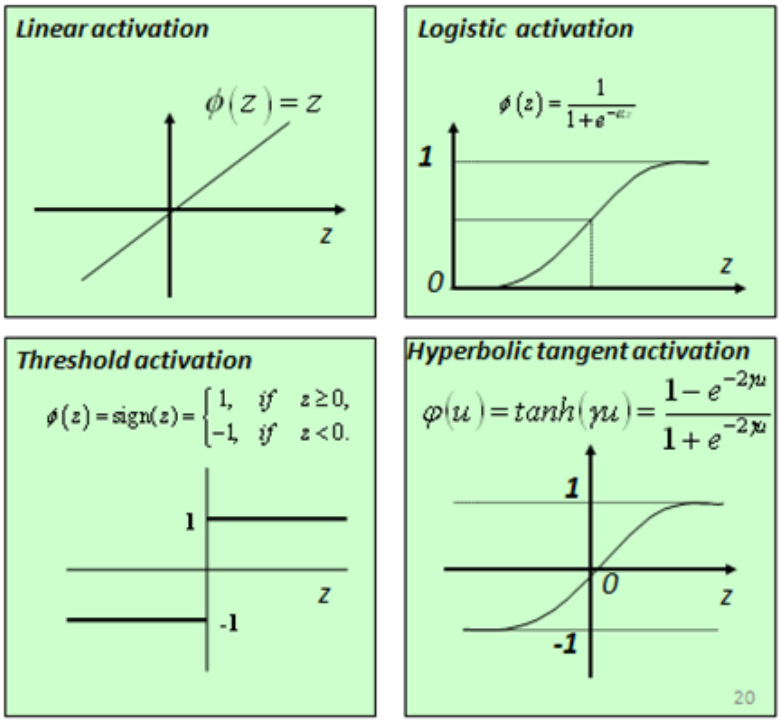
```
Confusion Matrix and Statistics

                 Reference
Prediction        Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria              14              0            0               0            0      1            2      0            0
  Coast.Sardinia         0              8            0               0            0      0            0      0            0
  East.Liguria           0              0            6               0            0      0            0      0            1
  Inland.Sardinia        0              0            0              10            0      0            0      0            0
  North.Apulia           0              0            0               0            3      0            0      0            0
  Sicily                 1              0            0               0            0      2            2      0            0
  South.Apulia           0              0            0               0            0      0           40      0            0
  Umbria                 0              0            1               0            0      0            0     11            0
  West.Liguria           0              0            2               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9123
                 95% CI : (0.8446, 0.9571)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8911
```

8.3. Närvivõrkude klassifitseerimismudelid

- **Ühekihilise närvivõrgu mudeli võrrandi üldkuju klassifitseerimise ülesannete korral on järgmine**

$$P(y_i = Ck|\boldsymbol{X} = \boldsymbol{x_i}) = \phi_o(\alpha_k + \sum_h w_{hk}\phi_h(\alpha_h + \sum_{j=1}^{p} w_{ih}x_{ij}))$$

- **Mudel prognoosib *Y*-i klassi C_k kuulumise tõenäosust sõltuvalt argumentide $X_1, X_2, \ldots, X_p$ väärtustest**
- **Aktiveerimisfunktsioonid ϕ_h**



$\phi_o(z)$ *logistic* → klassifitseerimine

$$\text{Sigmoid: } f(x) = \frac{1}{1+e^{-x}}$$

$\phi_o(z)$ funktsioonina on lubatud ka

- ***threshold*** **binaarklassifitseerimisel ja**
- **(*hyperbolic tangent* +1)/2, mis annab väärtused vahemikus (0,1)**

- **Närvivõrkude mudelid on kõrgelt parametriseeritud, ristvalideerimine on vajalik ülesobituse (*overfitting*) vältimiseks**

Närvivõrgud klassifitseerimiseks R-is

Kasutame R-i funktsiooni nnet paketist nnet ühe varjutud kihiga mudeli loomiseks, 2- või 3- kihilise närvivõrgu loomiseks kasutame paketti neuralnet

```
library(nnet)
```

Fit Neural Networks

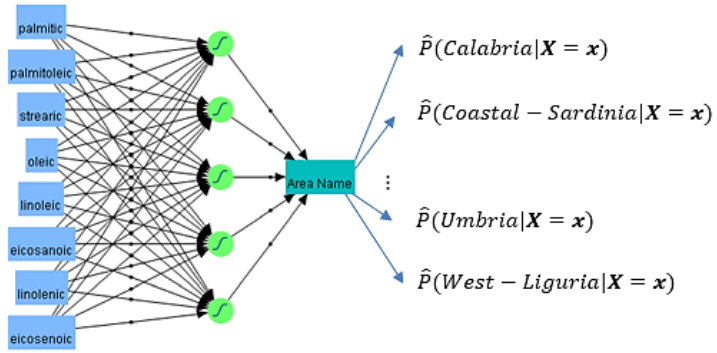
Description:

Fit single-hidden-layer neural network, possibly with skip-layer connections.

Usage:

```
nnet(x, ...)

## S3 method for class 'formula':
nnet(formula, data, weights, ...,
    subset, na.action = na.fail, contrasts = NULL)

## Default S3 method:
nnet(x, y, weights, size, Wts, mask,
    linout = FALSE, entropy = FALSE, softmax = FALSE,
    censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,
    maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,
    abstol = 1.0e-4, reltol = 1.0e-8, ...)
```

Närvivõrk olive andmestiku klassifitseerimisel



- **Arvulised tunnused nõuvad skaleerimist, mittearvulistest → *dummy variables***

Märkused:

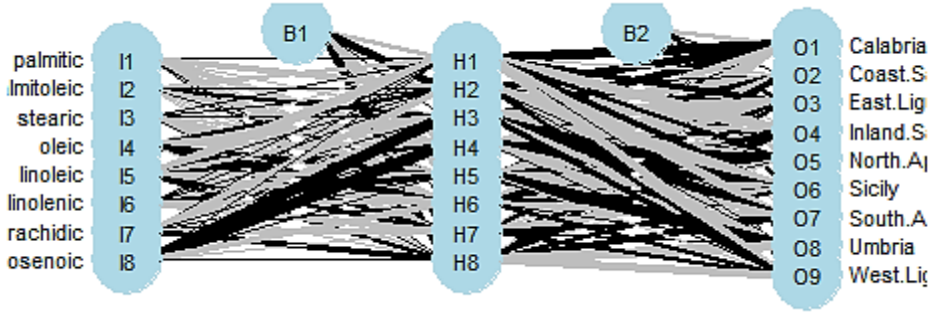
- **Suurem *size* suurendab mudeli täpsust.**
- **Suurem *decay* pikendab mudeli otsingut, kuid reeglina leitakse parem, ehkki kerkib ülesobituse probleem.**
- **Maksimaalse korduste arvu (*maxit*) suurendamine on vajalik, eriti *size* ja/või *decay* suurenemisel.**
- **linout=FALSE → klassifitseerimine**

NÄIDE: kasutame andmestiku olive paketist dslabs

```
library(nnet)
RNGkind(sample.kind = "Rounding")
set.seed(123)
olive.nn<-nnet(area~.,data=olive.train,size=8,decay=.025,maxit=5000,trace=FALSE)
olive.nn
```

```
a 8-8-9 network with 153 weights
inputs: palmitic palmitoleic stearic oleic linoleic linolenic arachidic eicosenoic
output(s): area
options were - softmax modelling  decay=0.025
```

```
library(NeuralNetTools)
plotnet(olive.nn, cex=0.7)
```




```
library(caret)
confusionMatrix(as.factor(predict(olive.nn,type="class")),olive.train$area,mode =
"everything")
confusionMatrix(as.factor(predict(olive.nn,newdata= olive.test, type="class")),
olive.test$area,mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             40              0            0               0            0      0            0      0            0
  Coast.Sardinia        0             25            0               0            0      0            0      0            0
  East.Liguria          0              0           41               0            0      0            0      0            0
  Inland.Sardinia       0              0            0              55            0      0            0      0            0
  North.Apulia          0              0            0               0           22      0            0      0            0
  Sicily                1              0            0               0            0     33            0      0            0
  South.Apulia          0              0            0               0            0      0          162      0            0
  Umbria                0              0            0               0            0      0            0     40            0
  West.Liguria          0              0            0               0            0      0            0      0           39

Overall Statistics

               Accuracy : 0.9978
                 95% CI : (0.9879, 0.9999)
    No Information Rate : 0.3537
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9973
```

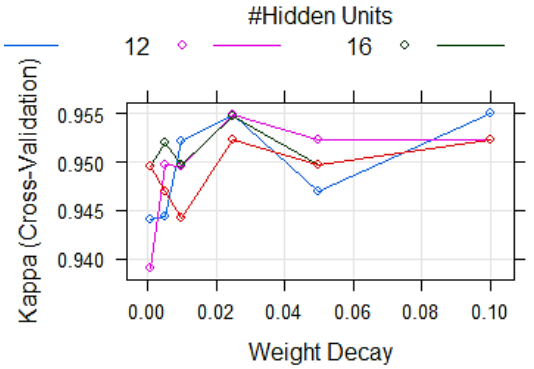
```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             14              0            0               0            0      0            2      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            7               0            0      0            0      0            0
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                1              0            0               0            0      3            2      0            0
  South.Apulia          0              0            0               0            0      0           40      0            0
  Umbria                0              0            0               0            0      0            0     11            0
  West.Liguria          0              0            2               0            0      0            0      0           11

Overall Statistics

               Accuracy : 0.9386
                 95% CI : (0.8776, 0.975)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9238
```

- **treenimisandmetel**
- **testandmetel**

Närvivõrgu parameetrite häälestus paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5 ,search = "random",classProbs =
TRUE,summaryFunction = multiClassSummary)
nnet_grid <-  expand.grid(size = c(8,12,16,20),
                          decay = c(0.1,0.05,0.025,0.01,0.005,0.001))
RNGkind(sample.kind = "Rounding")
set.seed(123)
cv_nn <- train(area~.,data=olive.train,
               method = "nnet",linout = FALSE,skip=T,trace = F,
               metric = "Kappa",
               trControl = train.control, tuneGrid = nnet_grid)
plot(cv_nn)
head(cv_nn$results[order(-cv_nn$results$Kappa),],1)
```

```
  size decay  Accuracy     Kappa AccuracySD    KappaSD
8    8   0.1 0.9630856 0.9549798 0.01778169 0.02161567
```




```
confusionMatrix(as.factor(predict(cv_nn$finalModel,type="class")),olive.train$area,mode
= "everything")
confusionMatrix(as.factor(predict(cv_nn$finalModel,newdata= olive.test, type="class")),
olive.test$area,mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             39              0            0               0            0      1            0      0            0
  Coast.Sardinia        0             24            0               0            0      0            0      0            0
  East.Liguria          0              0           41               0            0      0            0      0            0
  Inland.Sardinia       0              1            0              55            0      0            0      0            0
  North.Apulia          0              0            0               0           22      0            0      0            0
  Sicily                1              0            0               0            0     30            1      0            0
  South.Apulia          1              0            0               0            0      2          161      0            0
  Umbria                0              0            0               0            0      0            0     40            0
  West.Liguria          0              0            0               0            0      0            0      0           39

Overall Statistics

               Accuracy : 0.9847
                 95% CI : (0.9688, 0.9938)
    No Information Rate : 0.3537
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9813
```

- **treenimisandmetel**

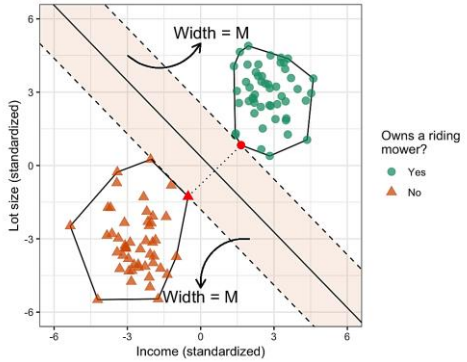
```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      0            1      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            6               0            0      0            0      0            0
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      3            3      0            0
  South.Apulia          0              0            0               0            0      0           40      0            0
  Umbria                0              0            2               0            0      0            0     11            0
  West.Liguria          0              0            1               0            0      0            0      0           11

Overall Statistics

               Accuracy : 0.9386
                 95% CI : (0.8776, 0.975)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9238
```

- **testandmetel**

8.4. Tugivektormasinad (*Support Vector Machines*, SVM)

- **Tugivektormasinad kuuluvad suurima eraldusvööga klassifikaatorite (*maximum margin classifiers*) hulka.** https://bookdown.org/mpfoley1973/data-sci/maximal-margin-classifier.html
- **SVM-meetod otsib tunnuste ruumi jagavat tasapinda, millele lähimate andmepunktide (tugivektorite) omavaheline kaugus on kõige suurem. Klasse eraldava tasapinna asend sõltub vaid tugivektoritest.**
- **Tugivektorid on klasside eralduspinnale kõige lähemal olevad andmepunktid.**
- **Tugivektormasinad on algselt leiutatud nominaalse funktsioontunnuse ja pidevate seletavate tunnuste jaoks. Nominaalsete argumenttunnuste puhul moodustatakse *dummy variables*. Pidevad argumendid normaliseeritakse.**
- **Tugivektormasinad on rakendatavad ka pideva muutuja väärtuste prognoosimiseks ehk regressiooniülesandeks.** https://koalatea.io/r-svm-regression/

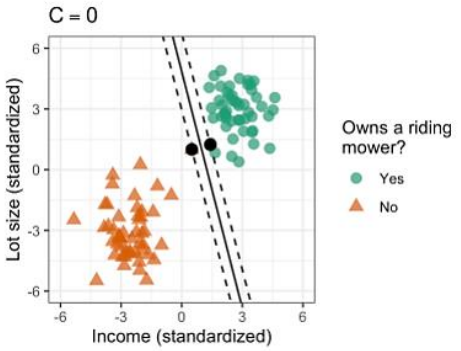
- **HMC (*hard margin classifiers*): valida kaks paralleelset hüpertasandit, mis eraldavad kaks andmeklassi, nii et nende vaheline kaugus M oleks võimalikult suur. Nende kahe hüpertasandi piiratud piirkonda nimetatakse „eraldusvööndiks„ (*margin*) ja *maximum-märgin* hüpertasand on nende vahel poolel teel asuv hüpertasapind**

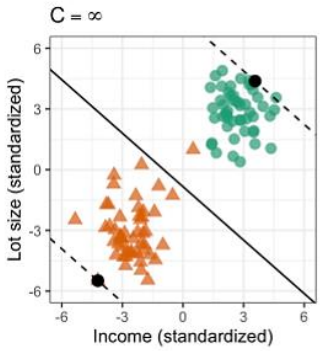


$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \begin{cases} \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i \left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) \geq M, \quad i = 1, 2, \ldots, n \end{cases}$$

- **SMC (*soft margin classifiers*): erineb HMC-st selles, et lubab objekti olemasolu eraldusvööndis, *C* on SVM meetodi tuunimisparameeter**





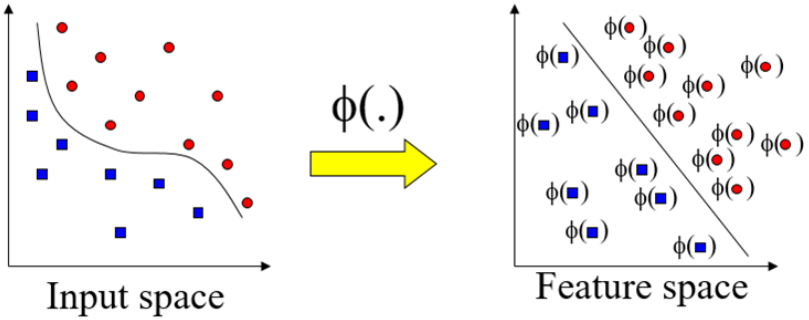
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \begin{cases} \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i \left(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}\right) \geq M\left(1 - \xi_i\right), \quad i = 1, 2, \ldots, n \\ \xi_i \geq 0, \\ \sum_{i=1}^{n} \xi_i \leq C \end{cases}$$

tuning parameter C, also known as Cost, that determines the possible misclassifications. It essentially imposes a penalty to the model for making an error: the higher the value of C, the less likely it is that the SVM algorithm will misclassify a point

https://bradleyboehmke.github.io/HOML/svm.html#prerequisites-12

- ***Kernel trick:*** **ruumi teisendus ϕ**

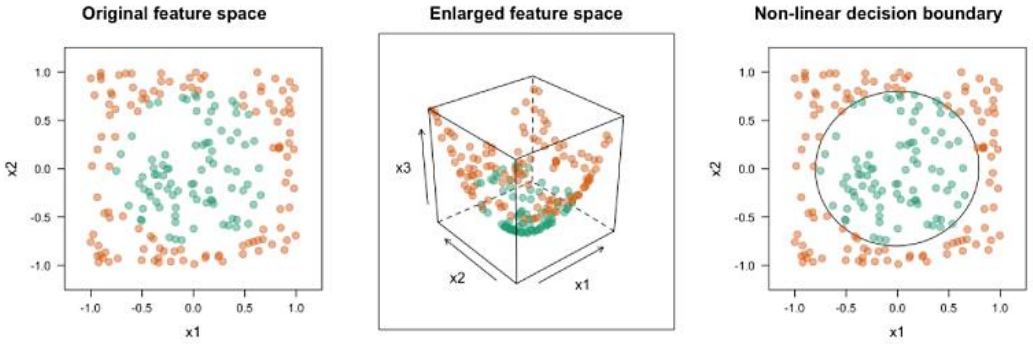


- ***Kernel function:*** $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

- **Populaarsemad kernel-funktsioonid SVM meetodil**

  - *d*-th degree polynomial: $K(x, x') = \gamma(1 + \langle x, x' \rangle)^d$
  - Radial basis function: $K(x, x') = \exp(\gamma \|x - x'\|^2)$
  - Hyperbolic tangent: $K(x, x') = \tanh(k_1 \|x - x'\| + k_2)$

- **Igale kernel-funktsioonile vastavad oma häälestatavad parameetrid**



Teisendus $X_3=X_1^2+X_2^2$ polünomiaalkerneliga d=2

SVM R-is

Kasutame R-i funktsiooni ksvm paketist kernlab

```
library(kernlab)
```

ksvm {kernlab} R Documentation

Support Vector Machines

Description

Support Vector Machines are an excellent tool for classification, novelty detection, and regression. `ksvm` supports the well known C-svc, nu-svc, (classification) one-class-svc (novelty) eps-svr, nu-svr (regression) formulations along with native multi-class classification formulations and the bound-constraint SVM formulations.
`ksvm` also supports class-probabilities output and confidence intervals for regression.

Usage

```
## S4 method for signature 'formula'
ksvm(x, data = NULL, ..., subset, na.action = na.omit, scaled = TRUE)
```

`type` `ksvm` can be used for classification , for regression, or for novelty detection. Depending on whether `y` is a factor or not, the default setting for `type` is `C-svc` or `eps-svr`, respectively, but can be overwritten by setting an explicit value.
Valid options are:

- `C-svc` C classification
- `nu-svc` nu classification
- `C-bsvc` bound-constraint svm classification
- `spoc-svc` Crammer, Singer native multi-class
- `kbb-svc` Weston, Watkins native multi-class
- `one-svc` novelty detection
- `eps-svr` epsilon regression
- `nu-svr` nu regression
- `eps-bsvr` bound-constraint svm regression

`kernel` the kernel function used in training and predicting. This parameter can be set to any function, of class kernel, which computes the inner product in feature space between two vector arguments (see `kernels`).
kernlab provides the most popular kernel functions which can be used by setting the kernel parameter to the following strings:

- `rbfdot` Radial Basis kernel "Gaussian"
- `polydot` Polynomial kernel
- `vanilladot` Linear kernel
- `tanhdot` Hyperbolic tangent kernel
- `laplacedot` Laplacian kernel
- `besseldot` Bessel kernel
- `anovadot` ANOVA RBF kernel
- `splinedot` Spline kernel
- `stringdot` String kernel

Setting the kernel parameter to "matrix" treats `x` as a kernel matrix calling the `kernelMatrix` interface.

The kernel parameter can also be set to a user defined function of class kernel by passing the function name as an argument.

`kpar` the list of hyper-parameters (kernel parameters). This is a list which contains the parameters to be used with the kernel function. For valid parameters for existing kernels are :

- `sigma` inverse kernel width for the Radial Basis kernel function "rbfdot" and the Laplacian kernel "laplacedot".
- `degree, scale, offset` for the Polynomial kernel "polydot"
- `scale, offset` for the Hyperbolic tangent kernel function "tanhdot"
- `sigma, order, degree` for the Bessel kernel "besseldot".
- `sigma, degree` for the ANOVA kernel "anovadot".
- `length, lambda, normalized` for the "stringdot" kernel where length is the length of the strings considered, lambda the decay factor and normalized a logical parameter determining if the kernel evaluations should be normalized.

Hyper-parameters for user defined kernels can be passed through the kpar parameter as well. In the case of a Radial Basis kernel function (Gaussian) kpar can also be set to the string "automatic" which uses the heuristics in `sigest` to calculate a good `sigma` value for the Gaussian RBF or Laplace kernel, from the data. (default = "automatic").

`C` cost of constraints violation (default: 1) this is the 'C'-constant of the regularization term in the Lagrange formulation.

❑ SVM R-is

NÄIDE: kasutame andmestiku olive paketist dslabs

```
library(kernlab)
olive.ksvm = ksvm(area~.,data=olive.train,cross=5)
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.185374140325592

Number of Support Vectors : 193

Objective Function Value : -2.3216 -5.604 -3.1965 -7.0842 -20.1885 -13.9204 -3.4147 -3.1398 -2.5556
 -10.2563 -2.1744 -2.5396 -3.6441 -1.919 -2.0458 -4.9526 -4.8609 -5.8839 -4.5266 -9.7304 -8.0733 -3.
0435 -3.4035 -4.4822 -2.5006 -2.6668 -13.4514 -4.6628 -3.8983 -2.7443 -16.2974 -3.6477 -3.5127 -3.17
4 -3.4797 -2.9593
Training error : 0.026201
Cross validation error : 0.030578
```

❑ Tulemus testandmetel

```
confusionMatrix(predict(olive.ksvm,newdata= olive.test, type="response"), olive.test$area,
mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      1            1      0            0
  Coast.Sardinia        0              7            0               0            0      0            0      0            0
  East.Liguria          0              0            6               0            0      0            0      0            1
  Inland.Sardinia       0              1            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            1               0            0      2            2      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            1               0            0      0            0     11            0
  West.Liguria          0              0            1               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9211
                 95% CI : (0.8554, 0.9633)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9016
```

Parameetri C häälestus

```
olive.ksvm = ksvm(area~.,data=olive.train,cross=5,C=15)
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 15

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.180937609718721

Number of Support Vectors : 171

Objective Function Value : -2.3158 -6.0281 -3.1983 -10.7657 -76.2714 -43.4095 -3.4136 -3.1087 -2.5594 -27.8989 -2.1693 -2.5343 -3.8073
 -1.9082 -2.0362 -5.0306 -5.0193 -6.2555 -4.475 -14.0068 -10.7377 -3.0489 -3.3977 -4.5753 -2.4959 -2.665 -28.4399 -4.8419 -3.9643 -2.70
81 -65.5709 -3.6427 -3.464 -3.1354 -3.433 -2.9725
Training error : 0.002183
Cross validation error : 0.048017
```

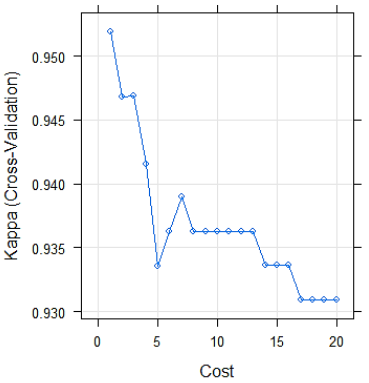
Tulemus testandmetel

```
confusionMatrix(predict(olive.ksvm,newdata= olive.test, type="response"), olive.test$area,
mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      1            1      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            7               0            0      0            0      0            1
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            1               0            0      2            2      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            1               0            0      0            0     11            0
  West.Liguria          0              0            0               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9386
                 95% CI : (0.8776, 0.975)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9235
```

SVM parameetrite tuunimine paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5)
RNGkind(sample.kind = "Rounding")
set.seed(123)
svm1 <- train(area~.,data=olive.train, method = "svmLinear", trControl =
train.control,tuneGrid = expand.grid(C=0:20),metric = "Kappa")
plot(svm1)
head(svm1$results[order(- svm1$results$Kappa),],1)
```




```
  C  Accuracy     Kappa AccuracySD    KappaSD
1 0.9606987 0.9519006 0.02632914 0.03221578
```

Tulemus testandmetel

```
confusionMatrix(predict(svm1,newdata= olive.test), olive.test$area, mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      0            1      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            6               0            0      0            0      0            1
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      3            2      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            1               0            0      0            0     11            0
  West.Liguria          0              0            2               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9386
                 95% CI : (0.8776, 0.975)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9235
```

SVM parameetrite häälestus paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5 ,search = "random")
RNGkind(sample.kind = "Rounding")
set.seed(123)
svm2 <- train(area~.,data=olive.train, method = "svmRadial", trControl = train.control,
tuneLength = 20,metric = "Kappa")
plot(svm2)
head(svm2$results[order(- svm2$results$Kappa),],1)
```

```
     sigma        C  Accuracy    Kappa AccuracySD    KappaSD
0.0980105 11.57506 0.9672698 0.959978 0.01081871 0.01327759
```

Tulemus testandmetel

```
confusionMatrix(predict(svm2,newdata= olive.test), olive.test$area, mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      1            1      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            8               0            0      0            0      0            1
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      2            2      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            1               0            0      0            0     11            0
  West.Liguria          0              0            0               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.889, 0.9804)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9344
```

SVM parameetrite häälestus paketiga caret

```
library(caret)
train.control <- trainControl(method = "cv",number = 5 ,search = "random)
RNGkind(sample.kind = "Rounding")
set.seed(123)
svm3 <- train(area~.,data=olive.train, method = "svmPoly", trControl = train.control,
tuneLength = 20,metric = "Kappa")
plot(svm3)
head(svm3$results[order(- svm3$results$Kappa),],1)
```

```
  degree     scale         C  Accuracy     Kappa AccuracySD    KappaSD
       2 0.1930864 0.6236088 0.9650959 0.9572086 0.02374813 0.02925871
```

Tulemus testandmetel

```
confusionMatrix(predict(svm3,newdata= olive.test), olive.test$area, mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      1            1      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            8               0            0      0            0      0            2
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      2            2      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            1               0            0      0            0     11            0
  West.Liguria          0              0            0               0            0      0            0      0            9

Overall Statistics

               Accuracy : 0.9386
                 95% CI : (0.8776, 0.975)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9235
```

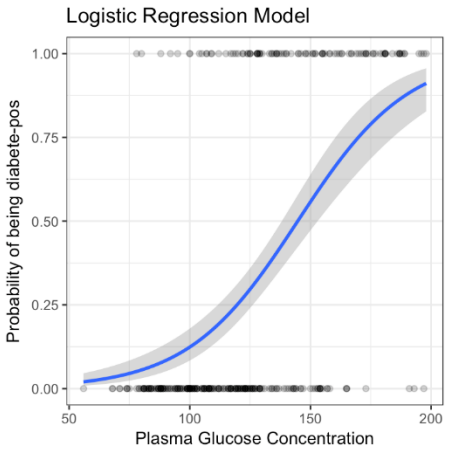
8.5. Logistiline regressioon

- **Logistilse regressiooni puhul prognoositakse $\theta(\boldsymbol{X}) = P(Y = 1|\boldsymbol{X})$.**
- **Klassifitseerimise reegel:**

If $\hat{\theta}(\boldsymbol{X}) = P(Y = 1|\boldsymbol{X}) \geq p$ then classify Y as being from class 1, i.e. "Yes" or "Success".
If $\hat{\theta}(\boldsymbol{X}) = P(Y = 1|\boldsymbol{X}) < p$ then classify Y as being from class 0, i.e. "No" or "Failure".

- **Tõenäosuste prognoosimiseks kasutatakse üldistatud lineaarne mudel (*generalized linear model*, GLM) $\theta(\boldsymbol{X})$ *logit* teisendusest**

$$L = ln\left(\frac{\theta(\boldsymbol{X})}{1-\theta(\boldsymbol{X})}\right) = \beta_o + \beta_1 U_1 + \beta_2 U_2 + \cdots + \beta_{k-1} U_{k-1}$$



- **Omakorda tõenäosuste prognoosimiseks kasutatakse valemit**

$$\hat{\theta}(\boldsymbol{X}) = \frac{e^{\hat{L}}}{1+e^{\hat{L}}} = \frac{1}{1+e^{-\hat{L}}}, \quad \textbf{kus } L = \beta_o + \beta_1 U_1 + \beta_2 U_2 + \cdots + \beta_{k-1} U_{k-1}$$

- **Nagu MLR mudeli korral, mudeli argumentide kohal saab proovida polünomiaaltegurid ja tegurite koosmõjusid.**

https://bookdown.org/mpfoley1973/data-sci/logistic-regression.html

Logistiline regressioon R-is: binaarklassifitseerimine

Binaarklassifitseerimise korral kasutame baaspaketi funktsiooni glm

```
glm(mudeli valem,data=andmestik,family="binomial")
```

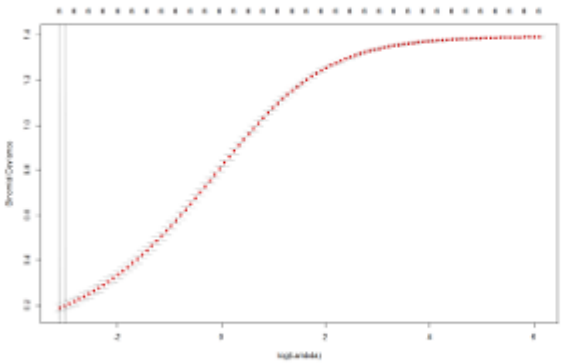
- **On võimalik rakendada ka regulariseerimist nagu MLR mudeli puhul**

Ridge Logistic: $ln\left(\frac{\theta(x)}{1-\theta(x)}\right) = \beta_o + \sum_{j=1}^{k}\beta_j u_j + \lambda\sum_{j=1}^{k}\beta_j^2$

Lasso Logistic: $ln\left(\frac{\theta(x)}{1-\theta(x)}\right) = \eta_o + \sum_{j=1}^{k}\beta_j u_j + \lambda\sum_{j=1}^{k}|\beta_j|$

Elastic Net Logistic: $ln\left(\frac{\theta(x)}{1-\theta(x)}\right) = \eta_o + \sum_{j=1}^{k}\beta_j u_j + \lambda_1\sum_{j=1}^{k}|\beta_j| + \lambda_2\sum\beta_j^2$

```
> ridge.cv = cv.glmnet(X,y,alpha=0,family="binomial")
> lasso.cv = cv.glmnet(X,y,alpha=1,family="binomial")
> plot(ridge.cv)
```



```
> ridge.lam = ridge.cv$lambda.min
> ridge.lam
[1] 0.04476108
```

```
> ypred.ridge = predict(forg.ridge,newx=X,s=ridge.lam,type="response")
> ypred.lasso = predict(forg.lasso,newx=X,s=lasso.lam,type="response")

> table(ypred.ridge>.5,y)
             y
           0   1
  FALSE  100   1
  TRUE     0  99

>table(ypred.lasso>.5,y)
             y
           0   1
  FALSE  100   0
  TRUE     0 100
```

http://www.sthda.com/english/articles/36-classification-methods-essentials/151-logistic-regression-essentials-in-r/

https://rpubs.com/jkylearmstrong/logit_w_caret

❑ Logistiline regressioon R-is: mitme klassi probleem

NÄIDE: kasutame andmestiku olive **paketist** dslabs

Kasutame R-i funktsiooni multinom **paketist** nnet

```
library(nnet)
olive.logreg = multinom(area ~ ., data = olive.train)
olive.logreg
```

```
Call:
multinom(formula = area ~ ., data = olive.train)

Coefficients:
               (Intercept)    palmitic palmitoleic    stearic      oleic  linoleic  linolenic  arachidic  eicosenoic
Coast.Sardinia   -689.77354  -471.80532   -22.40131  145.09978  335.97442 2559.7291 -261.51813  767.58459 -1031.45350
East.Liguria     -315.74730   -46.22965   134.97086  -50.78624  303.69043  245.6318 -625.36266  697.68685 -2248.84729
Inland.Sardinia   718.42352 -1309.34432   138.87780 -316.05679 -607.34789  496.9249 -647.30516  621.85036 -1200.78590
North.Apulia    -1461.07970  1634.80283   277.16975 -481.87591 4938.86556 1812.4357  249.69744  618.76999   296.83326
Sicily             71.83472   816.20448   304.73555  292.32462 2218.38967 1359.3080 -380.25922  673.05959   228.77351
South.Apulia     1177.60533  1992.33268  1317.94564  631.96835 6083.74275 4853.7885 -698.29453  279.53126   297.71604
Umbria             53.77488  -821.19029   158.27505 -779.75170   25.85475  211.8562  -43.75252   40.14769 -1193.08956
West.Liguria     -212.50558  -878.91881   889.21881  -15.37689  758.98033 1423.1793 -987.83062 -707.77933    66.83135

Residual Deviance: 3.169948
AIC: 147.1699
```

❑ Tulemus testandmetel

```
confusionMatrix(predict(log.reg,newdata= olive.test), olive.test$area, mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      0            0      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            8               0            0      0            0      0            0
  Inland.Sardinia       0              0            0              10            0      0            0      0            1
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      3            3      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            0               0            0      0            0     11            0
  West.Liguria          0              0            1               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9561
                 95% CI : (0.9006, 0.9856)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9455
```

https://ashgreat.github.io/analyticsAppBook/multinomial-logistic-regression-mnl.html

Ristvalideerimine paketiga caret

```
library(caret)
RNGkind(sample.kind = "Rounding")
set.seed(123)
train.control <- trainControl(method = "cv",number = 5)
fit <- train(area~.,data=olive.train, method = "multinom", trControl = train.control, trace
= FALSE,tuneLength = 20,metric = "Kappa")
head(fit$results[order(- fit$results$Kappa),],1)
```

```
      decay  Accuracy     Kappa AccuracySD    KappaSD
0.03162278 0.9521194 0.9415574 0.01608578 0.01957019
```

Tulemus testandmetel

```
confusionMatrix(predict(fit,newdata= olive.test), olive.test$area, mode = "everything")
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Calabria Coast.Sardinia East.Liguria Inland.Sardinia North.Apulia Sicily South.Apulia Umbria West.Liguria
  Calabria             15              0            0               0            0      0            0      0            0
  Coast.Sardinia        0              8            0               0            0      0            0      0            0
  East.Liguria          0              0            7               0            0      0            0      0            1
  Inland.Sardinia       0              0            0              10            0      0            0      0            0
  North.Apulia          0              0            0               0            3      0            0      0            0
  Sicily                0              0            0               0            0      3            3      0            0
  South.Apulia          0              0            0               0            0      0           41      0            0
  Umbria                0              0            2               0            0      0            0     11            0
  West.Liguria          0              0            0               0            0      0            0      0           10

Overall Statistics

               Accuracy : 0.9474
                 95% CI : (0.889, 0.9804)
    No Information Rate : 0.386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9346
```

https://ashgreat.github.io/analyticsAppBook/multinomial-logistic-regression-mnl.html