[Year]

R ja koneoppiminen

ASIANTUNTIJALUENNOT

BRAGGE JANNE TTM23SAI

[COMPANY NAME] | [Company address]

Sisällysluettelo

Logistinen Regressio

Tässä on yksinkertainen esimerkki lineaarisesta regressiosta R-kielellä, jossa selitetään vaihe vaiheelta, miten se toteutetaan. Projekti analysoi kuvitteellista dataa, jossa yritetään ennustaa myyntiä mainosbudjetin perusteella.

Datan valmistelu

Ensiksi luodaan esimerkkidata, joka sisältää mainosbudjetin (Advertising_Budget) ja myynnin (Sales).

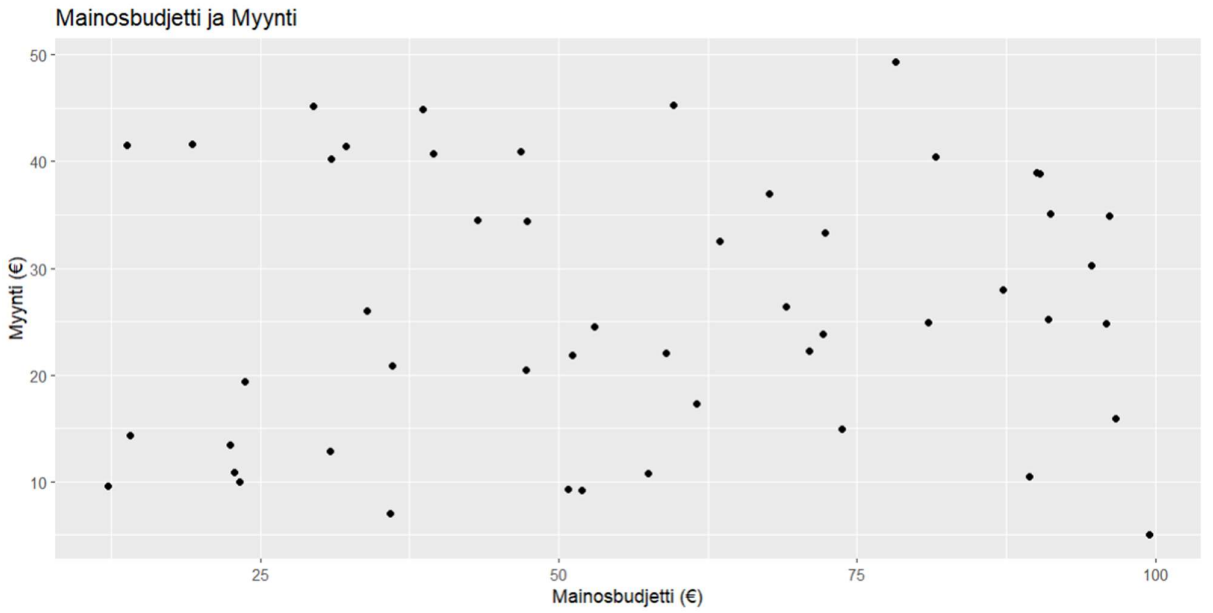
```
# Luo esimerkkidata

set.seed(123) # Aseta satunnaissiementä toistettavuuden varmistamiseksi

data <- data.frame(

  Advertising_Budget = runif(50, 10, 100), # Satunnaisia mainosbudjetteja välillä 10-100

  Sales = runif(50, 5, 50) # Satunnaisia myyntilukuja välillä 5-50

)

# Tarkastele dataa

head(data)
```

```
  Advertising_Budget     Sales
1           35.88198  7.062403
2           80.94746 24.899003
3           46.80792 40.951618
4           89.47157 10.485467
5           94.64206 30.242659
6           14.10008 14.293913
```

Datan visualisointi

Piirretään scatterplot, jotta nähdään mahdollinen yhteys mainosbudjetin ja myynnin välillä.

```
# Asenna ggplot2, jos sitä ei ole
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)


# Scatterplot
ggplot(data, aes(x = Advertising_Budget, y = Sales)) +
 geom_point() +
 labs(title = "Mainosbudjetti ja Myynti",
     x = "Mainosbudjetti (€)",
     y = "Myynti (€)")
```



Mallin rakentaminen

Käytetään lm()-funktiota luodaksemme lineaarisen regressiomallin.

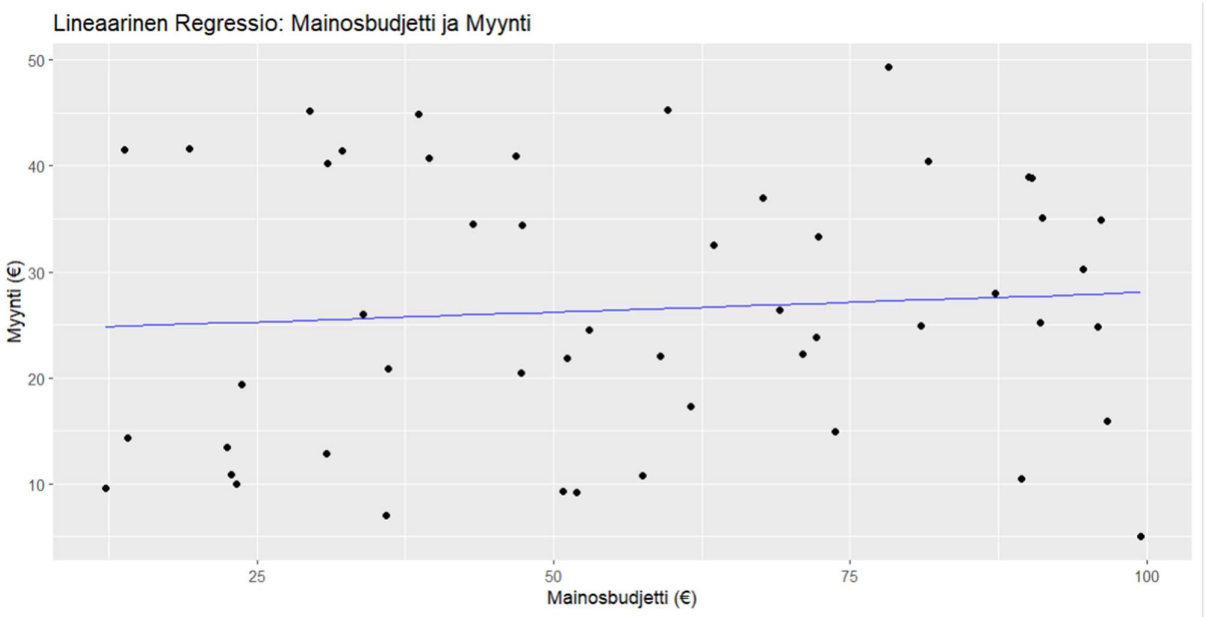
```
# Lineaarinen regressiomalli

model <- lm(Sales ~ Advertising_Budget, data = data)

# Tulosta mallin yhteenveto

summary(model)
```

```
Call:
lm(formula = Sales ~ Advertising_Budget, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-23.012 -11.466  -2.106  10.921  22.066

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       24.37177    4.23211   5.759 5.84e-07 ***
Advertising_Budget  0.03687    0.06764   0.545    0.588
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 12.54 on 48 degrees of freedom
Multiple R-squared:  0.006151,  Adjusted R-squared:  -0.01455
F-statistic: 0.2971 on 1 and 48 DF,  p-value: 0.5882
```

Ennusteet ja visualisointi

Lisätään ennustettu trendiviiva scatterplottiin.

```
# Lisää mallin ennusteet dataan

data$Predicted_Sales <- predict(model)


# Visualisoi malli

ggplot(data, aes(x = Advertising_Budget, y = Sales)) +

 geom_point() +

 geom_line(aes(y = Predicted_Sales), color = "blue") +

 labs(title = "Lineaarinen Regressio: Mainosbudjetti ja Myynti",

    x = "Mainosbudjetti (€)",

    y = "Myynti (€)")
```



Mallin arviointi

Lasketaan mallin suorituskykymetriikoita, kuten R^2 ja RMSE.

```
# R-squared ja RMSE

r_squared <- summary(model)$r.squared

rmse <- sqrt(mean((data$Sales - data$Predicted_Sales)^2))


# Tulosta metriikat

cat("R²:", round(r_squared, 3), "\n")

cat("RMSE:", round(rmse, 3), "\n")
```

r_squared	0.00615143500295936
rmse	12.2873875138606

Päätöspuu

Datan valmistelu

Luodaan esimerkkidata, kuten lineaarisen regressioesimerkin kohdalla.

```
# Luo esimerkkidata

set.seed(123)

data <- data.frame(

 Advertising_Budget = runif(50, 10, 100), # Satunnaisia mainosbudjetteja

 Sales = runif(50, 5, 50) # Satunnaisia myyntilukuja

)


# Tarkastele dataa

head(data)
```

```
> head(data)
  Advertising_Budget     Sales
1           35.88198  7.062403
2           80.94746 24.899003
3           46.80792 40.951618
4           89.47157 10.485467
5           94.64206 30.242659
6           14.10008 14.293913
>
```

Datan jako treeni- ja testijoukkoihin

Jaetaan data osiin mallin kouluttamista ja testaamista varten.

```
# Asenna caret-paketti, jos sitä ei ole
if (!require("caret")) install.packages("caret")
library(caret)


# Jaa data treeni- (70%) ja testijoukkoon (30%)
set.seed(123)
index <- createDataPartition(data$Sales, p = 0.7, list = FALSE)
train_data <- data[index, ]
test_data <- data[-index, ]
```

```
>
> # Jaa data treeni- (70%) ja testijoukkoon (30%)
> set.seed(123)
> index <- createDataPartition(data$Sales, p = 0.7, list = FALSE)
> train_data <- data[index, ]
> test_data <- data[-index, ]
```

Päätöspuun rakentaminen

Käytetään rpart-pakettia päätöspuun luomiseen.

Asenna rpart-paketti, jos sitä ei ole

if (!require("rpart")) install.packages("rpart")

library(rpart)

Luo päätöspuumalli

tree_model <- rpart(Sales ~ Advertising_Budget, data = train_data, method = "anova")

Tulosta puun yhteenveto

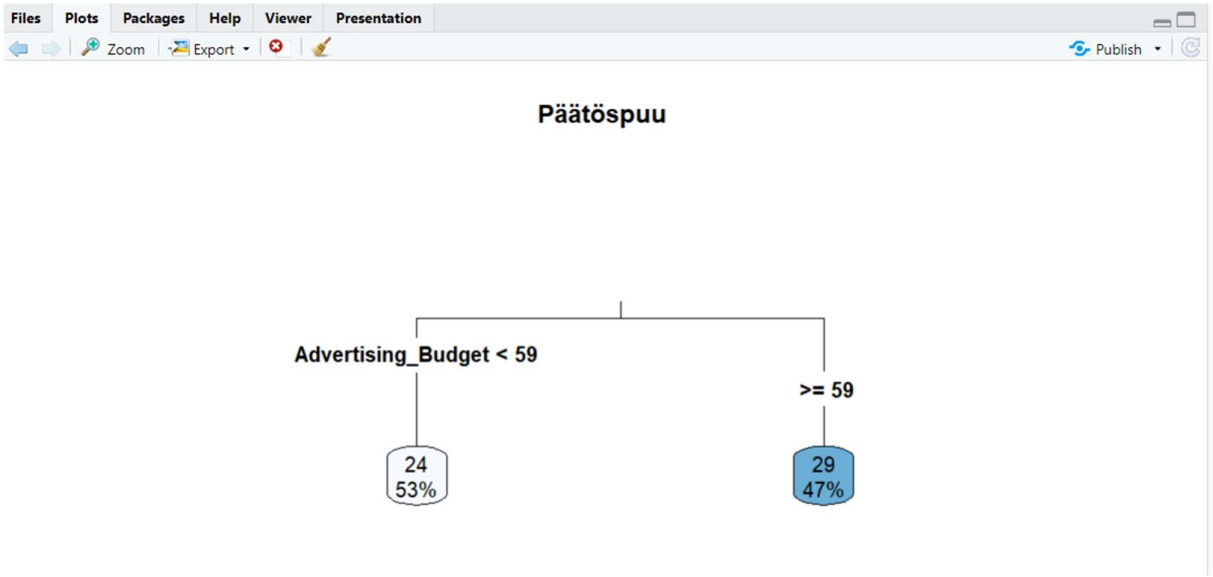
summary(tree_model)

```
>
> # Asenna rpart-paketti, jos sitä ei ole
> if (!require("rpart")) install.packages("rpart")
Loading required package: rpart
> library(rpart)
>
> # Luo päätöspuumalli
> tree_model <- rpart(Sales ~ Advertising_Budget, data = train_data, method = "anova")
>
> # Tulosta puun yhteenveto
> summary(tree_model)
Call:
rpart(formula = Sales ~ Advertising_Budget, data = train_data,
    method = "anova")
  n= 38

         CP nsplit rel error   xerror      xstd
1 0.0502825      0 1.0000000 1.044571 0.1547839
2 0.0100000      1 0.9497175 1.273164 0.2010636

Variable importance
Advertising_Budget
               100

Node number 1: 38 observations,    complexity param=0.0502825
  mean=26.24547, MSE=148.7639
  left son=2 (20 obs) right son=3 (18 obs)
  Primary splits:
      Advertising_Budget < 58.57932 to the left,  improve=0.0502825, (0 missing)

Node number 2: 20 observations
  mean=23.65082, MSE=155.5727

Node number 3: 18 observations
  mean=29.12841, MSE=125.4071

>
```

Päätöspuun visualisointi

Visualisoidaan päätöspuu, jotta ymmärretään sen rakenne.

```
# Asenna rpart.plot-paketti, jos sitä ei ole
if (!require("rpart.plot")) install.packages("rpart.plot")
library(rpart.plot)

# Piirrä päätöspuu
rpart.plot(tree_model, type = 3, fallen.leaves = TRUE, main = "Päätöspuu")
```



Ennusteiden tekeminen

Käytetään testidataa ennusteiden tekemiseen.

Tee ennusteet testidatalle

test_data$Predicted_Sales <- predict(tree_model, newdata = test_data)

Tarkastele testidatan ennusteita

head(test_data)

```
> # Tee ennusteet testidatalle
> test_data$Predicted_Sales <- predict(tree_model, newdata = test_data)
>
> # Tarkastele testidatan ennusteita
> head(test_data)
   Advertising_Budget    Sales Predicted_Sales
10           51.09533 21.85082        23.65082
11           96.11500 34.93018        29.12841
18           13.78536 41.55753        23.65082
23           67.64561 36.95821        29.12841
26           73.76774 14.90535        29.12841
27           58.96594 22.09174        29.12841
>
```

Mallin arviointi

Arvioidaan päätöspuumallin suorituskykyä metriikoilla, kuten RMSE.

Laske RMSE

rmse <- sqrt(mean((test_data$Sales - test_data$Predicted_Sales)^2))

Tulosta RMSE

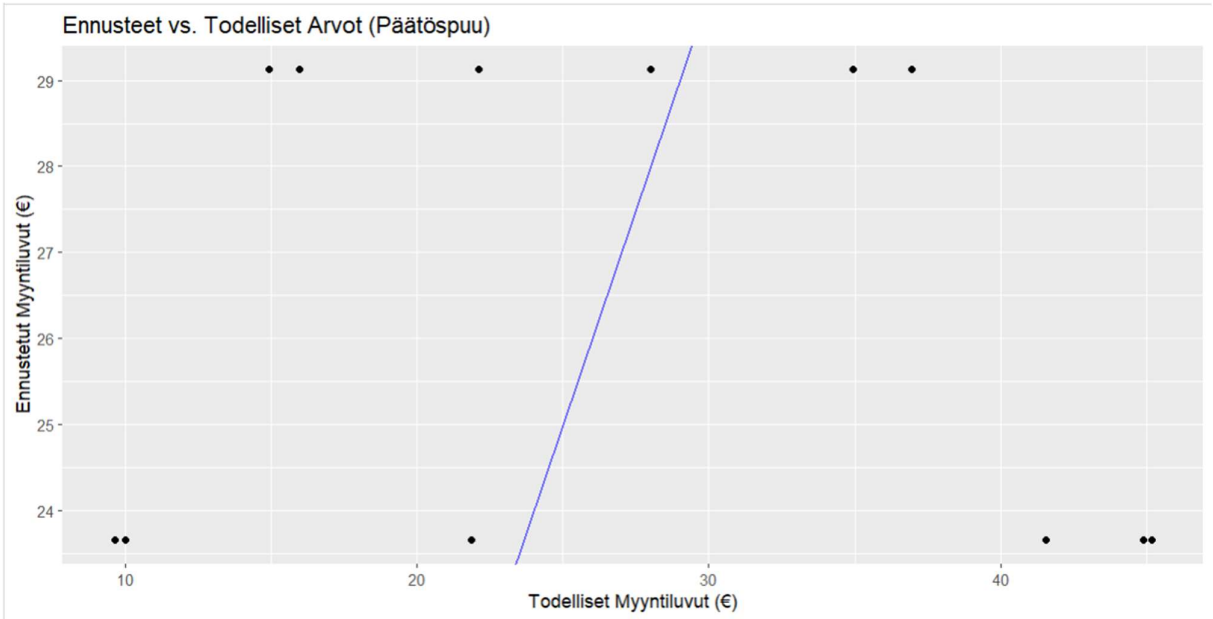
cat("RMSE päätöspuulle:", round(rmse, 3), "\n")

```
>
> # Laske RMSE
> rmse <- sqrt(mean((test_data$Sales - test_data$Predicted_Sales)^2))
>
> # Tulosta RMSE
> cat("RMSE päätöspuulle:", round(rmse, 3), "\n")
RMSE päätöspuulle: 13.364
>
>
```

Visualisointi

Visualisoidaan ennusteet vs. toteutuneet arvot.

```
# Scatterplot todellisten ja ennustettujen arvojen välillä
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)

ggplot(test_data, aes(x = Sales, y = Predicted_Sales)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "blue") +
  labs(title = "Ennusteet vs. Todelliset Arvot (Päätöspuu)",
       x = "Todelliset Myyntiluvut (€)",
       y = "Ennustetut Myyntiluvut (€)")
```



Kommentit ja jatkokehitys

1. **Päätöspuun selitys**: Päätöspuu jakaa datan yksinkertaisiin sääntöihin perustuen ja toimii hyvin, kun data sisältää epälineaarisia yhteyksiä.
2. **Jatkokehitys**: Voit kokeilla muita menetelmiä, kuten satunnaismetsiä (random forest), mikäli päätöspuun suorituskyky ei ole riittävä.
3. **Tuning**: Päätöspuun monimutkaisuutta voi säätää rpart-funktiossa esimerkiksi cp-parametrilla (complexity parameter).

KNN

Datan valmistelu

Luodaan esimerkkidata, jossa on kaksi muuttujaa: Advertising_Budget ja Sales.

```
# Luo esimerkkidata

set.seed(123)

data <- data.frame(

 Advertising_Budget = runif(100, 10, 100), # Mainosbudjetti välillä 10-100

 Sales = runif(100, 5, 50) # Myynti välillä 5-50

)


# Tarkastele dataa

head(data)
```

```
> 
> # Tarkastele dataa
> head(data)
  Advertising_Budget    Sales
1           35.88198 31.99950
2           80.94746 19.97706
3           46.80792 26.98759
4           89.47157 47.95132
5           94.64206 26.73061
6           14.10008 45.06576
> 
> 
```

Datan skaalaus

KNN-menetelmä on herkkä muuttujien mittayksiköille, joten data on hyvä skaalata.

```
# Skaalaa muuttujat välillä 0-1

if (!require("caret")) install.packages("caret")

library(caret)


# Skaalaus

scaled_data <- data
```

scaled_data[, 1:2] <- scale(data[, 1:2])

Tarkasta skaalattu data

head(scaled_data)

```
> 
> # Skaalaa muuttujat välillä 0-1
> if (!require("caret")) install.packages("caret")
> library(caret)
> 
> # Skaalaus
> scaled_data <- data
> scaled_data[, 1:2] <- scale(data[, 1:2])
> 
> # Tarkasta skaalattu data
> head(scaled_data)
  Advertising_Budget       Sales
1         -0.7402982  0.32514147
2          1.0166700 -0.68771329
3         -0.3143283 -0.09709745
4          1.3489993  1.66903427
5          1.5505811 -0.11874712
6         -1.5895088  1.42593432
> 
```

Datan jako treeni- ja testijoukkoihin

Jaetaan data treeni- ja testijoukkoihin (70% treeni, 30% testi).

```
# Jaa data
set.seed(123)
index <- createDataPartition(scaled_data$Sales, p = 0.7, list = FALSE)
train_data <- scaled_data[index, ]
test_data <- scaled_data[-index, ]
```

KNN-mallin rakentaminen

Käytetään kknn-pakettia KNN-mallin rakentamiseen.

```
# Asenna kknn-paketti, jos sitä ei ole
if (!require("kknn")) install.packages("kknn")
library(kknn)

# Rakenna KNN-malli
knn_model <- kknn(Sales ~ Advertising_Budget,
                  train = train_data,
                  test = test_data,
                  k = 5) # Käytetään 5 lähintä naapuria

# Ennusteet testidatalle
test_data$Predicted_Sales <- knn_model$fitted.values
```

Mallin arviointi

Lasketaan KNN-mallin suorituskykymetriikat, kuten RMSE.

RMSE (Root Mean Squared Error)

rmse <- sqrt(mean((test_data$Sales - test_data$Predicted_Sales)^2))

Tulosta RMSE

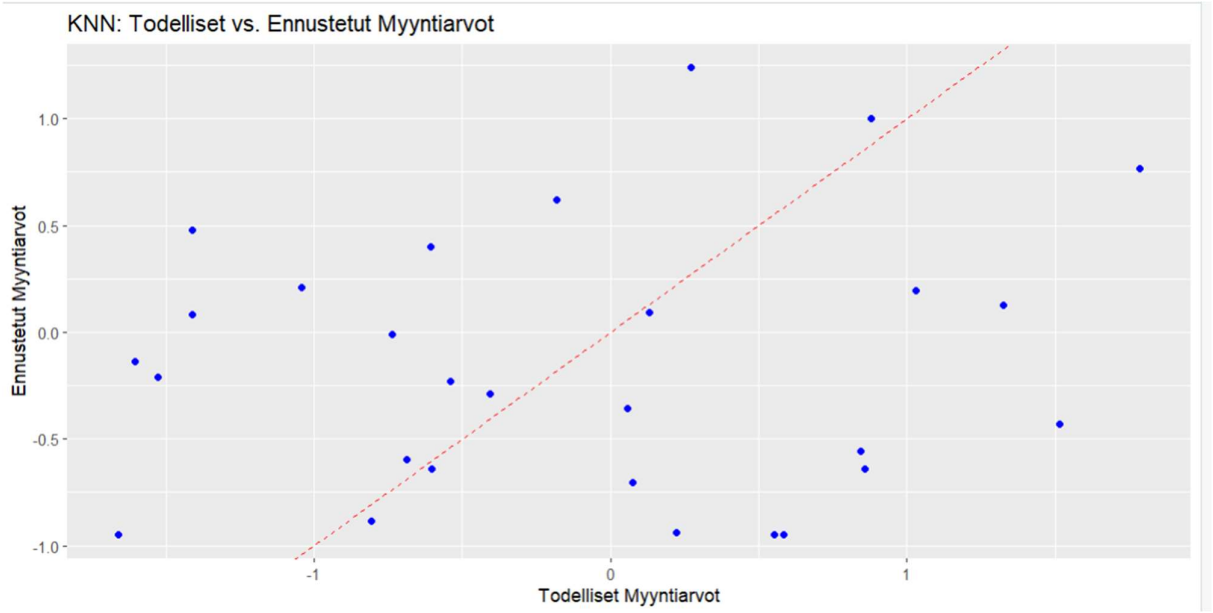
cat("RMSE KNN-mallille:", round(rmse, 3), "\n")

```
>
> # RMSE (Root Mean Squared Error)
> rmse <- sqrt(mean((test_data$Sales - test_data$Predicted_Sales)^2))
>
> # Tulosta RMSE
> cat("RMSE KNN-mallille:", round(rmse, 3), "\n")
RMSE KNN-mallille: 1.085
```

Visualisointi

Visualisoidaan ennusteiden ja toteutuneiden arvojen välinen suhde.

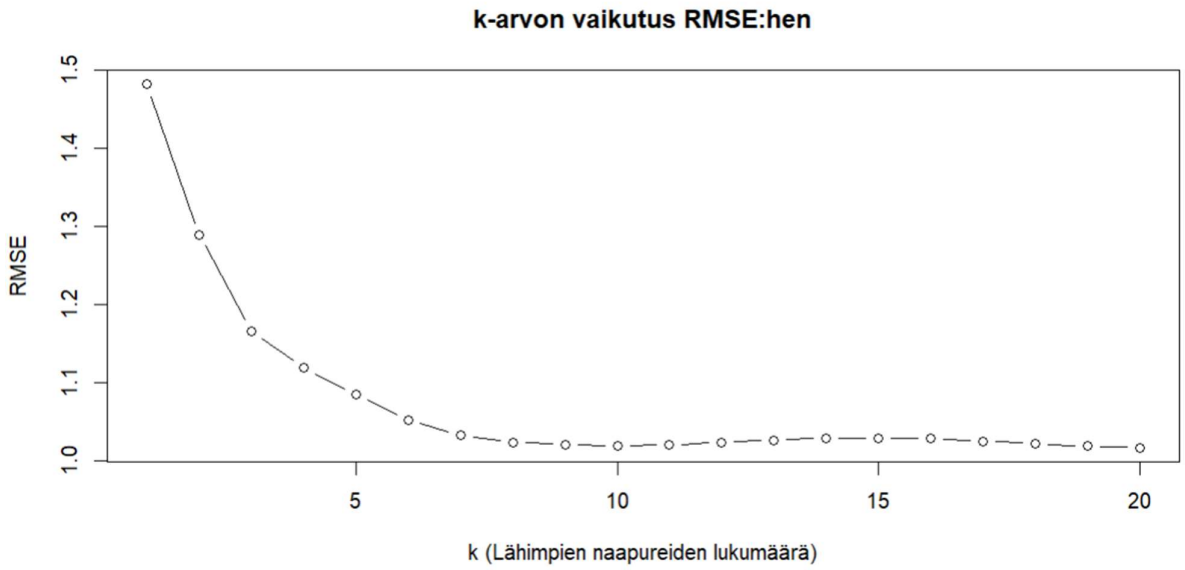
```
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)


# Scatterplot: Todelliset vs. Ennustetut arvot
ggplot(test_data, aes(x = Sales, y = Predicted_Sales)) +
 geom_point(color = "blue") +
 geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
 labs(title = "KNN: Todelliset vs. Ennustetut Myyntiarvot",
     x = "Todelliset Myyntiarvot",
     y = "Ennustetut Myyntiarvot")
```



Hyperparametrien säätäminen

KNN-mallin suorituskyky riippuu suuresti lähimpien naapureiden lukumäärästä (k). Voimme etsiä parhaan arvon käyttämällä esimerkiksi ristiinvalidaatiota.

```
# Kokeile eri k-arvoja ja valitse paras

k_values <- 1:20

rmse_values <- sapply(k_values, function(k) {

  model <- kknn(Sales ~ Advertising_Budget, train = train_data, test = test_data, k = k)
```




```
  sqrt(mean((test_data$Sales - model$fitted.values)^2))

})


# Parhaan k-arvon valinta

best_k <- k_values[which.min(rmse_values)]

cat("Paras k:", best_k, "\n")


# Visualisoi k:n vaikutus RMSE:hen

plot(k_values, rmse_values, type = "b", main = "k-arvon vaikutus RMSE:hen",

    xlab = "k (Lähimpien naapureiden lukumäärä)", ylab = "RMSE")
```

Arviointi

1. **KNN-menetelmä** on tehokas, mutta se vaatii datan skaalauksen ja sopivan k:n valinnan.
2. **Visualisoinnit** auttavat ymmärtämään, kuinka hyvin malli toimii.
3. **Jatkokehitys**: Voit lisätä muuttujia tai kokeilla muita ennustemalleja, kuten regressiota tai päätöspuita, ja vertailla niiden tuloksia.

K-Means

Datan valmistelu

Luodaan esimerkkidata, jossa on kaksi muuttujaa: Advertising_Budget ja Sales.

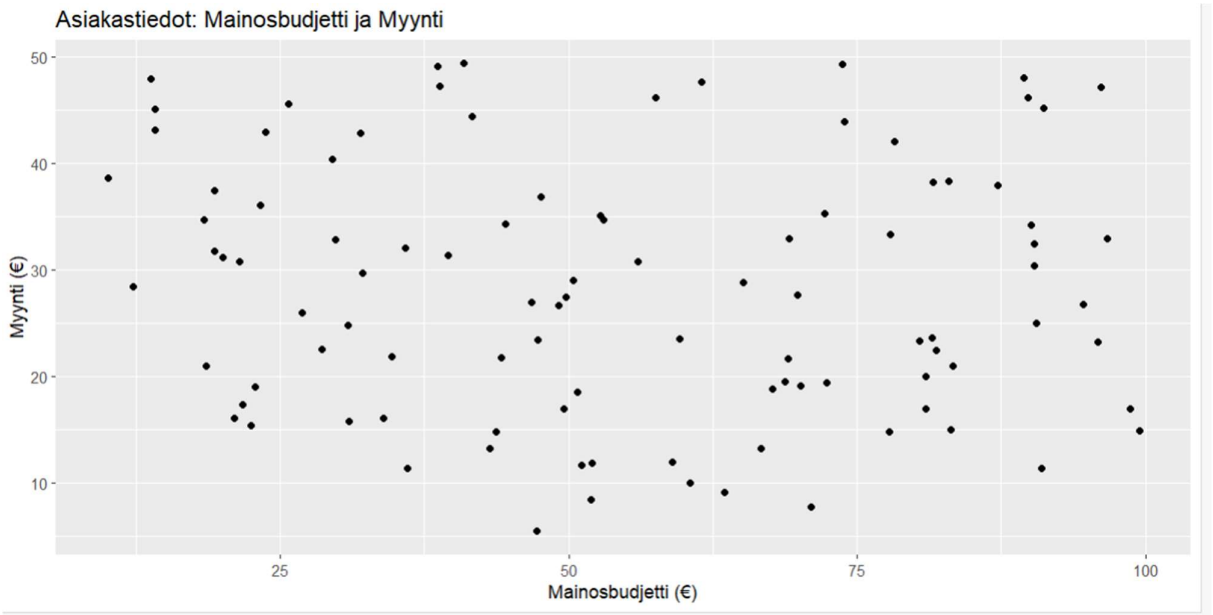
```
# Luo esimerkkidata

set.seed(123)

data <- data.frame(

 Advertising_Budget = runif(100, 10, 100), # Mainosbudjetti välillä 10-100

 Sales = runif(100, 5, 50)            # Myynti välillä 5-50

)


# Tarkastele dataa

head(data)
```

```
> # Tarkastele dataa
> head(data)
  Advertising_Budget    Sales
1           35.88198 31.99950
2           80.94746 19.97706
3           46.80792 26.98759
4           89.47157 47.95132
5           94.64206 26.73061
6           14.10008 45.06576
>
```

Datan visualisointi

Visualisoidaan data, jotta nähdään, miten asiakkaat jakautuvat.

```
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)


# Scatterplot
ggplot(data, aes(x = Advertising_Budget, y = Sales)) +
 geom_point() +
 labs(title = "Asiakastiedot: Mainosbudjetti ja Myynti",
     x = "Mainosbudjetti (€)",
     y = "Myynti (€)")
```



Datan skaalaus

K-Means on etäisyyksiin perustuva menetelmä, joten muuttujat on hyvä skaalata ennen klusterointia.

Skaalaa muuttujat

data_scaled <- scale(data)

Tarkastele skaalattua dataa

head(data_scaled)

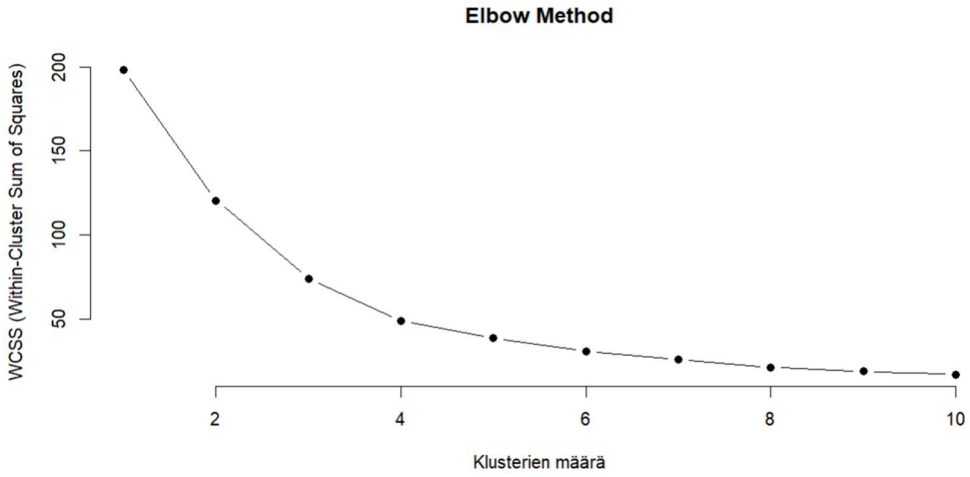
```
>
> # Tarkastele skaalattua dataa
> head(data_scaled)
     Advertising_Budget       Sales
[1,]         -0.7402982  0.32514147
[2,]          1.0166700 -0.68771329
[3,]         -0.3143283 -0.09709745
[4,]          1.3489993  1.66903427
[5,]          1.5505811 -0.11874712
[6,]         -1.5895088  1.42593432
>
```

Optimaalisen klusterimäärän valinta

Käytetään **"Elbow Method"** -menetelmää optimaalisen klusterimäärän arvioimiseen.

```
# Laske WCSS (Within-Cluster Sum of Squares) eri klusterimäärille
wcss <- sapply(1:10, function(k) {
  kmeans(data_scaled, centers = k, nstart = 25)$tot.withinss
})

# Visualisoi WCSS
plot(1:10, wcss, type = "b", pch = 19, frame = FALSE,
    xlab = "Klusterien määrä",
    ylab = "WCSS (Within-Cluster Sum of Squares)",
    main = "Elbow Method")
```

Optimaalinen klusterimäärä valitaan kohdasta, jossa WCSS:n lasku hidastuu ("kyynärpää").



K-Means Klusterointi

Valitaan esimerkiksi **3 klusteria** ja suoritetaan K-Means.

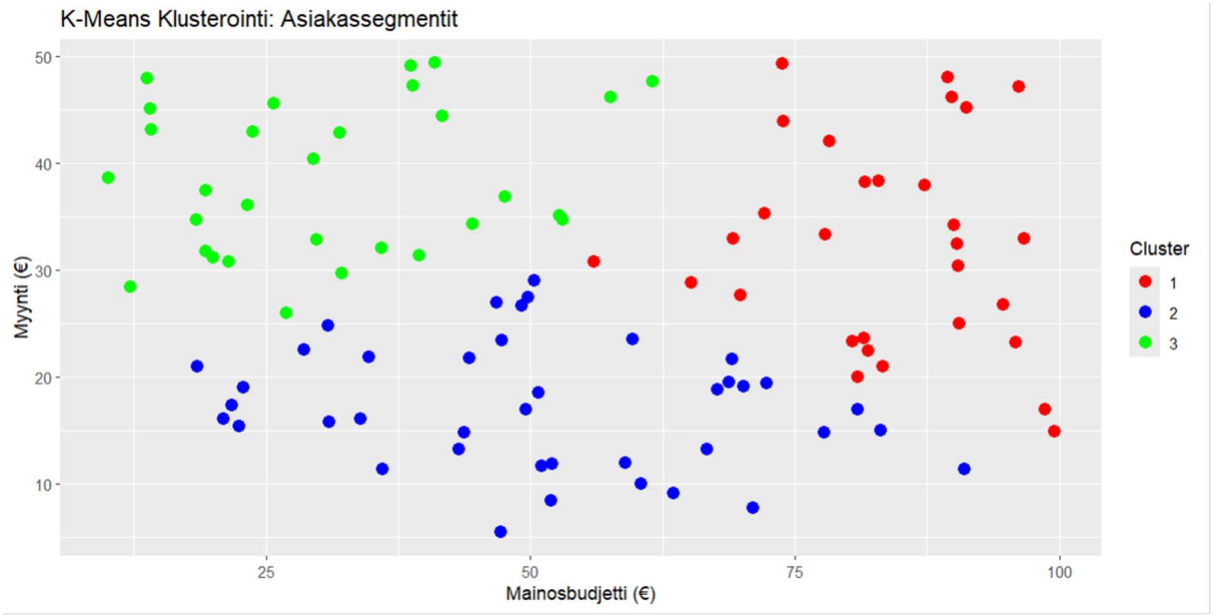
```
# Suorita K-Means klusterointi

set.seed(123)

kmeans_model <- kmeans(data_scaled, centers = 3, nstart = 25)


# Lisää klusterit alkuperäiseen dataan

data$Cluster <- as.factor(kmeans_model$cluster)


# Tarkastele klusteroitua dataa

head(data)
```

```
> # Suorita K-Means klusterointi
> set.seed(123)
> kmeans_model <- kmeans(data_scaled, centers = 3, nstart = 25)
>
> # Lisää klusterit alkuperäiseen dataan
> data$Cluster <- as.factor(kmeans_model$cluster)
>
> # Tarkastele klusteroitua dataa
> head(data)
  Advertising_Budget    Sales Cluster
1           35.88198 31.99950       3
2           80.94746 19.97706       1
3           46.80792 26.98759       2
4           89.47157 47.95132       1
5           94.64206 26.73061       1
6           14.10008 45.06576       3
>
>
```

Klusteroinnin visualisointi

Visualisoidaan klusterit eri väreillä.

```
# Scatterplot klustereille
ggplot(data, aes(x = Advertising_Budget, y = Sales, color = Cluster)) +
 geom_point(size = 3) +
 labs(title = "K-Means Klusterointi: Asiakassegmentit",
    x = "Mainosbudjetti (€)",
    y = "Myynti (€)") +
 scale_color_manual(values = c("red", "blue", "green"))
```



Tulosten tulkinta

- **Klusterit**: Jokainen klusteri edustaa ryhmää, jolla on samanlaisia ominaisuuksia. Esimerkiksi:
    - Klusteri 1: Asiakkaat, joilla on pieni mainosbudjetti ja pieni myynti.
    - Klusteri 2: Asiakkaat, joilla on suuri mainosbudjetti ja korkea myynti.
    - Klusteri 3: Asiakkaat, joilla on keskitasoinen budjetti ja myynti.
- **Sovellus**: Klusterien perusteella yritys voi kohdentaa markkinointistrategioita eri asiakasryhmille.

Jatkokehitys

- **Hyperparametrien säätäminen**: Testaa erilaisia klusterimääriä ja valitse optimaalinen.
- **Lisää muuttujia**: Voit lisätä muita muuttujia, kuten asiakastyytyväisyyden tai tuoteryhmän.
- **Vertailu muihin klusterointimenetelmiin**: Kokeile esim. hierarkkista klusterointia.

Naive Bayes

Datan valmistelu

Luodaan esimerkkidata, jossa asiakkaat jaetaan kahteen ryhmään (High ja Low) myyntilukujen perusteella.

```
# Luo esimerkkidata

set.seed(123)

data <- data.frame(

 Advertising_Budget = runif(100, 10, 100), # Mainosbudjetti välillä 10-100

 Sales = runif(100, 5, 50)            # Myynti välillä 5-50

)


# Luo luokkamuuttuja (High tai Low myynnin perusteella)

data$Category <- ifelse(data$Sales > median(data$Sales), "High", "Low")

data$Category <- as.factor(data$Category)


# Tarkastele dataa

head(data)
```

```
> # Luo esimerkkidata
> set.seed(123)
> data <- data.frame(
+     Advertising_Budget = runif(100, 10, 100), # Mainosbudjetti välillä 10-100
+     Sales = runif(100, 5, 50)                 # Myynti välillä 5-50
+ )
>
> # Luo luokkamuuttuja (High tai Low myynnin perusteella)
> data$Category <- ifelse(data$Sales > median(data$Sales), "High", "Low")
> data$Category <- as.factor(data$Category)
>
> # Tarkastele dataa
> head(data)
  Advertising_Budget    Sales Category
1           35.88198 31.99950     High
2           80.94746 19.97706      Low
3           46.80792 26.98759      Low
4           89.47157 47.95132     High
5           94.64206 26.73061      Low
6           14.10008 45.06576     High
>
>
```

Datan jako treeni- ja testijoukkoihin

Jaetaan data treeni- (70%) ja testijoukkoon (30%).

```
if (!require("caret")) install.packages("caret")

library(caret)


# Jaa data

set.seed(123)

index <- createDataPartition(data$Category, p = 0.7, list = FALSE)

train_data <- data[index, ]

test_data <- data[-index, ]
```

test_data	30 obs. of 3 variables
train_data	70 obs. of 3 variables

Naive Bayes -mallin rakentaminen

Käytetään **e1071**-pakettia Naive Bayes -mallin rakentamiseen.

```
# Asenna e1071-paketti, jos sitä ei ole

if (!require("e1071")) install.packages("e1071")

library(e1071)


# Rakenna Naive Bayes -malli

nb_model <- naiveBayes(Category ~ Advertising_Budget + Sales, data = train_data)


# Tarkastele mallia

print(nb_model)
```

```
>
> # Rakenna Naive Bayes -malli
> nb_model <- naiveBayes(Category ~ Advertising_Budget + Sales, data = train_data)
>
> # Tarkastele mallia
> print(nb_model)

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
High  Low
 0.5  0.5

Conditional probabilities:
      Advertising_Budget
Y          [,1]     [,2]
  High 49.82179 27.59648
  Low  57.88624 24.02247

      Sales
Y          [,1]     [,2]
  High 37.10214 6.665279
  Low  18.08305 5.641949
```

Ennusteiden tekeminen

Käytetään testijoukkoa ennusteiden tekemiseen ja lisätään ennusteet dataan.

Tee ennusteet testidatalle

test_data$Predicted_Category <- predict(nb_model, test_data)

Tarkastele ennusteita

head(test_data)

```
>
> # Tee ennusteet testidatalle
> test_data$Predicted_Category <- predict(nb_model, test_data)
>
> # Tarkastele ennusteita
> head(test_data)
   Advertising_Budget    Sales Category Predicted_Category
1            35.88198 31.99950     High               High
2            80.94746 19.97706      Low                Low
4            89.47157 47.95132     High               High
5            94.64206 26.73061      Low                Low
9            59.62915 23.48104      Low                Low
11           96.11500 47.08849     High               High
>
>
```

Mallin suorituskyvyn arviointi

Arvioidaan mallin tarkkuutta vertaamalla todellisia luokkia ennustettuihin.

Confusion Matrix

conf_matrix <- confusionMatrix(test_data$Predicted_Category, test_data$Category)

Tulosta tulokset

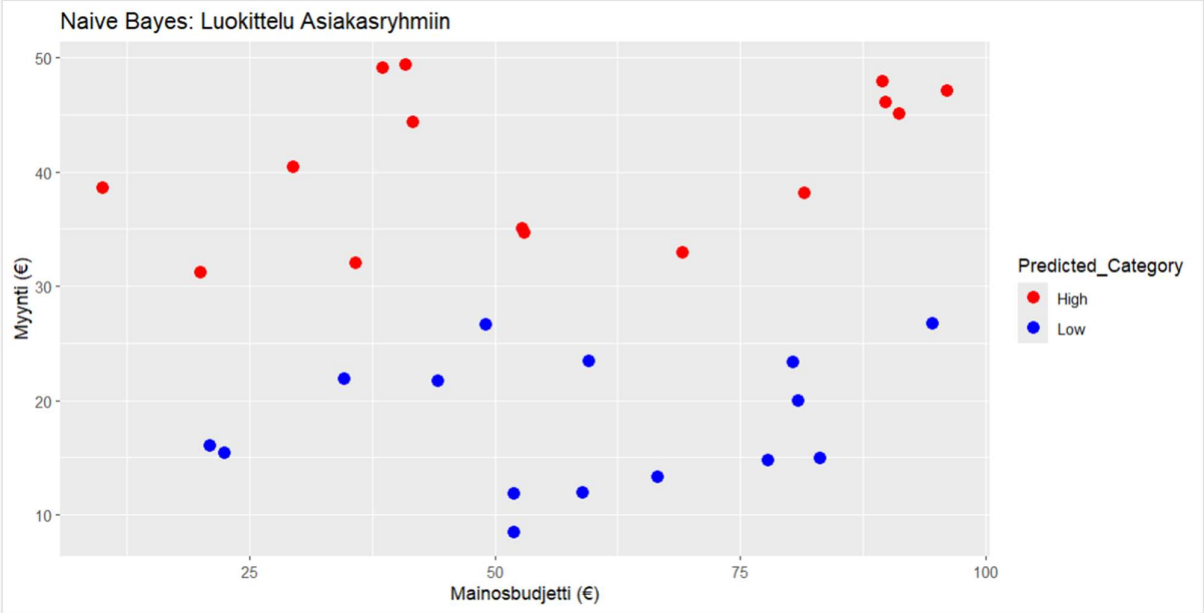
print(conf_matrix)

```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction High Low
      High   15   0
      Low     0  15

               Accuracy : 1
                 95% CI : (0.8843, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 9.313e-10

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : High

>
>
```

Visualisointi

Visualisoidaan, miten luokat jakautuvat päätösrajalla.

```
# Scatterplot ennustetuille luokille
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)

ggplot(test_data, aes(x = Advertising_Budget, y = Sales, color = Predicted_Category)) +
  geom_point(size = 3) +
  labs(title = "Naive Bayes: Luokittelu Asiakasryhmiin",
       x = "Mainosbudjetti (€)",
       y = "Myynti (€)") +
  scale_color_manual(values = c("red", "blue"))
```



Tulosten tulkinta

1. **Confusion Matrix**:
    - Näyttää mallin tarkkuuden, herkkyyden ja spesifisyyden.
    - Esim. "Accuracy" kertoo, kuinka suuri osa ennusteista on oikein.
2. **Visualisointi**:
    - Visualisointi auttaa näkemään, miten mainosbudjetti ja myynti vaikuttavat asiakkaan kategoriaan.

Jatkokehitys

- **Hyperparametrien säätö**: Säädä mallin parametreja, kuten prioreja, parantaaksesi suorituskykyä.
- **Lisää muuttujia**: Kokeile lisätä datan muita piirteitä, kuten asiakastyytyväisyys tai alue.
- **Vertaa muihin malleihin**: Kokeile esim. logistista regressiota tai SVM:ää ja vertaa tuloksia.