Statistical learning og programmering

1. Semesterprojekt. Antal ord:

Af Kenneth Gottfredsen, Eva Rauff og Sanne Sørensen

2022-12-28

Indhold

options(scipen = 100)

Problemfelt

Videnskabsteori og metode

Metodevalg

Den statistiske metode som anvendes i denne undersøgelse, kaldes for superviseret metode, fordi den tager udgangspunkt i én afhængig variabel. Den konkrete metode er en multibel lineær regression. Den vil vi anvende til og forudsige efterspørgslen på koldskål i liter fremadrettet på baggrund af effekten af nogle uafhængige vejr-variabler. Når vi først har trænet vores model på træningsdata og fået beregnet de nødvendige koefficienter, får alle variablerne i ligningen smidt en hat på toppen - dette indikerer prædiktion (Hastie et.al 2021). Den generelle formel bliver beskrevet nedenunder:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 ... + \hat{\beta}_p x_p + \epsilon$$

$\hat{y}$ er den forudsagte værdi af Y og $\hat{f}$ er et estimat for f.

$\hat{y}$ er desuden den afhængige variabel efterspørgsel i liter. x_i er udvalgte uafhængige variabler. $\hat{y} = f(X)$ indeholder variation som vi kan reducere ved, at bruge den korrekte SL-metode til, at beregne f med. Dog vil det aldrig være en fejlfri model vi ender ud med. Fordi estimatet ϵ er tilfældige fejl eller støj, man ikke kan gøre noget ved og den type fejl vil altid være til stede (Hastie et.al 2021).

Først indlæses `pacman::load()`:

```
# Vi bruger pacman til at installere og indhente relevante
# pakker på samme tid.
pacman::p_load("tidyverse", "magrittr", "nycflights13", "gapminder",
               "Lahman", "maps", "lubridate", "pryr", "hms", "hexbin",
               "feather", "htmlwidgets", "broom", "pander", "modelr",
               "XML", "httr", "jsonlite", "lubridate", "microbenchmark",
               "splines", "ISLR2", "MASS", "testthat", "leaps", "caret",
               "RSQLite", "class", "babynames", "nasaweather",
               "fueleconomy", "viridis", "readxl", "timeDate", "tinytex",
               "ggbeeswarm", "palmerpenguins", "hms", "RColorBrewer",
               "boot", "openxlsx", "writexl", "PerformanceAnalytics", "car", "pscl",
```

Import

I første omgang vil vi importere datasættet:

```
# Indlæser datasæt og gemmer det nye datasæt i et objekt.
data1 <- read_excel("data/stud_exam_data.xlsx")
 # Dernæst undersøges strukturen i datasættet.
str(data1)
```

```
## tibble [152 x 4] (S3: tbl_df/tbl/data.frame)
##  $ date             : POSIXct[1:152], format: "2022-04-01" "2022-04-02" ...
##  $ efterspørgsel    : num [1:152] 367 361 376 47 367 402 416 355 283 454 ...
##  $ kammerjunkere    : chr [1:152] "0" "0" "0" "1" ...
##  $ forventet_l_lager: chr [1:152] "3" "3" "3" "3" ...
```

Tidying og transformering af datasæt

Nu har vi fået indlæst datasættet. Det næste skridt er gøre strukturen i vores dataframe nemmere, at arbejde med og mere læsevenlig. Denne proces kaldes for tidy data, det betyder

at hver variabel har en kolonne, hver observation en række, samt hver observationsenhed er i en tabel (Wickham 2022). Dette vil gøre analysearbejdet væsentlig nemmere. Vi starter ud med at rekode nogle af variablerne, så de stemmer overens med hvad der står i opgavebesvarelsen.

I nedestående kode-chunk rekodes og transformeres de udvalgte variabler så de stemmer overens med eksamensbesvarelsen. Hele kodestumpen vil blive kædet sammen med 'pipe' funktionen' fra dplyr pakken. Omkodningerne bliver til sidst gemt i en ny dataframe som kaldes data1.

Derefter bruges `mutate()` til at lave en ny kolonne ud fra data1. Først laves der en date-variabel, som bliver kodet om til et date objekt med `ymd()` fra lubridate pakken.

I den næste del anvendes mutate igen til, at lave en kolonne der hedder dag, som bliver omkodet til en faktor. Dernæst koder vi date til et objekt med `ymd()` funktionen fra lubridate pakken. "lubridate.week.start",1=mandag, istedet for søndag som er standardindstillingerne i R.

Dernæst bruger vi `mutate()` igen til at danne en ny weekend-variabel der hedder weekend_1. I denne sammenhæng vælger vi at fredag, lørdag, søndag og fire andre helligdage er 1, ellers er de andre værdier 0. Dette kaldes for en dummyvariabel.

Måned, dag, kamjunk, forvent_lager og weekend_1 er alle kategoriske faktorer. For at gøre det nemmere at forstå hvad de forskellige værdier udtrykker, navngiver vi disse med `fct_recode()` funktionen.

```
data1 <- data1 %>%
  mutate(date = ymd(date), måned = factor(month(date)),
         kamjunk = factor(kammerjunkere), forvent_lager =
           factor(forventet_l_lager)) %>%
  mutate(dag = as.factor(wday(date, week_start =
                                getOption("lubridate.week.start", 1)))) %>%
  mutate(weekend_1 = as.integer(dag %in% c("5", "6", "7")| date %in%
                                  ymd("2022-04-14", "2022-04-18", "2022-05-26",
                                      "2022-06-06"))) %>%
  mutate(weekend = factor(weekend_1)) %>%
```

```
mutate(data1, kamjunk = fct_recode(kammerjunkere, "ja" = "0",
                                   "nej" = "1")) %>%
mutate(data1, forvent_lager = fct_recode(forventet_l_lager, "lav" = "1",
                                         "mellem" = "2", "høj" = "3")) %>%
mutate(data1, måned = fct_recode(måned, "april" = "4", "maj" = "5",
                                 "juni" = "6", "juli" = "7",
                                 "august" = "8")) %>%
mutate(data1, dag = fct_recode(dag, "mandag" = "1", "tirsdag" = "2",
                               "onsdag" = "3", "torsdag" = "4",
                               "fredag" = "5", "lørdag" = "6",
                               "søndag" = "7")) %>%
dplyr::select(date, måned, dag, efterspørgsel, kamjunk, forvent_lager,
              weekend_helligdag = weekend)
```

I denne kodechunk vil vi lave en HTTP GET-anmodning til en API fra DMI. Vi skal bruge adgangen til at få de relevante vejr-variable som vi senere skal bruge i vores analyse. API'en leverer til slut et objekt i JSON format som bliver transformeret om til en dataframe i stedet for en liste.

Først bruger vi base_url og info_url til at anmode om vejrdata fra DMI's API. req_url bruges til at udvælge specifikke parametre fra API´en.

I denne kodechunk vil vi transformere den data vi har hentet fra vores API-kald til nogle mere brugbare data.

Først bruger vi base_url og info_url til at anmode om vejrdata fra DMI's API. req_url bruges til at udvælge specifikke parametre fra API´en.

Derefter bruger vi `pivot_wider()`-funktionen til at sprede variablerne ud i deres egne separate kolonner.

Vi bruger derefter Mutate-funktionen til at konvertere kolonnen 'målingstidspunkt' til en datoformat Separate-funktionen bruges til at opdele kolonnen 'målingstidspunkt' i to separate kolonner som vi navngiver 'date' og 'time'.

`Filter()`funktionen udvælger rækker, der indeholder de første fire characters: "12:0".

```
data2 <- as.data.frame(do.call(cbind, list_dmi))
data2 <- dplyr::select(data2, features.properties.observed,
                       features.properties.value,
                       features.properties.parameterId) %>%
  rename(værdi = features.properties.value, parameter =
                 features.properties.parameterId,
  målingstidspunkt = features.properties.observed) %>%
  pivot_wider(names_from = parameter, values_from = værdi) %>%
  mutate(målingstidspunkt = as_datetime(målingstidspunkt)) %>%
  separate(målingstidspunkt, into = c('date', 'time'), sep = " ") %>%
  filter(str_sub(time, 1, 4) == "12:0") %>%
  mutate(date = as_date(date)) %>%
  mutate(time = as_hms(time)) %>%
  dplyr::select(-(temp_max_past12h:temp_min_past12h))
```

I nedestående kode-chunk merger vi data1 og data2 til data3 for at beholde alle observationer i x.

Vi bruger derefter mutate til at oprette fire nye variabler i data3 kaldet temp_gt25_3_dage. `Lag()`-funktionen er brugt til at lave variablerne, som har opfanget forsinkede værdier fra temp1, temp2 og temp3. Afslutningsvis dannes variablen 'temp_gt25_3_dage', som måler de dage hvor der har været mere end 3 dage i træk med >= 25 grader. Det er en dummyvariabel fordi vi bruger `if_else`. Da vi har et begrænset antal ord og tegn med mellemrum til rådighed, vil vi ikke lave en udtømmende variabelbeskrivelse. Relevante variabler bliver beskrevet når vi tolker på de forskellige parametre i regressionsanalysen.

```
data3 <- data1 %>%
left_join(data2, data1, by = c("date" = "date"))
dplyr::select(data3, date, time, weekend_helligdag, everything())
```

```
data3 <- data3 %>%
  mutate(temp1 = lag(temp_max_past1h, 1),
         temp2 = lag(temp_max_past1h, 2),
```

```
        temp3 = lag(temp_max_past1h, 3),
        temp1 = if_else(is.na(temp1), 0, temp1),
        temp2 = if_else(is.na(temp2), 0, temp2),
        temp3 = if_else(is.na(temp3), 0, temp3),
        temp_gt25_3_dage = if_else(temp1 >= 25 & temp2 >= 25 & temp3 >= 25,
                                   1, 0))
```

Datavisualisering og eksplorativ analyse

Nu er vores data blevet gjort tidy, det næste skridt er at undersøge hvilke vejr-mønstre der hænger sammen med butikkernes efterspørgslen af koldskål i det sammenkoblede datasæt, som vi har kaldt data3. Vi går derfor i gang med den eksplorative del af analysen.

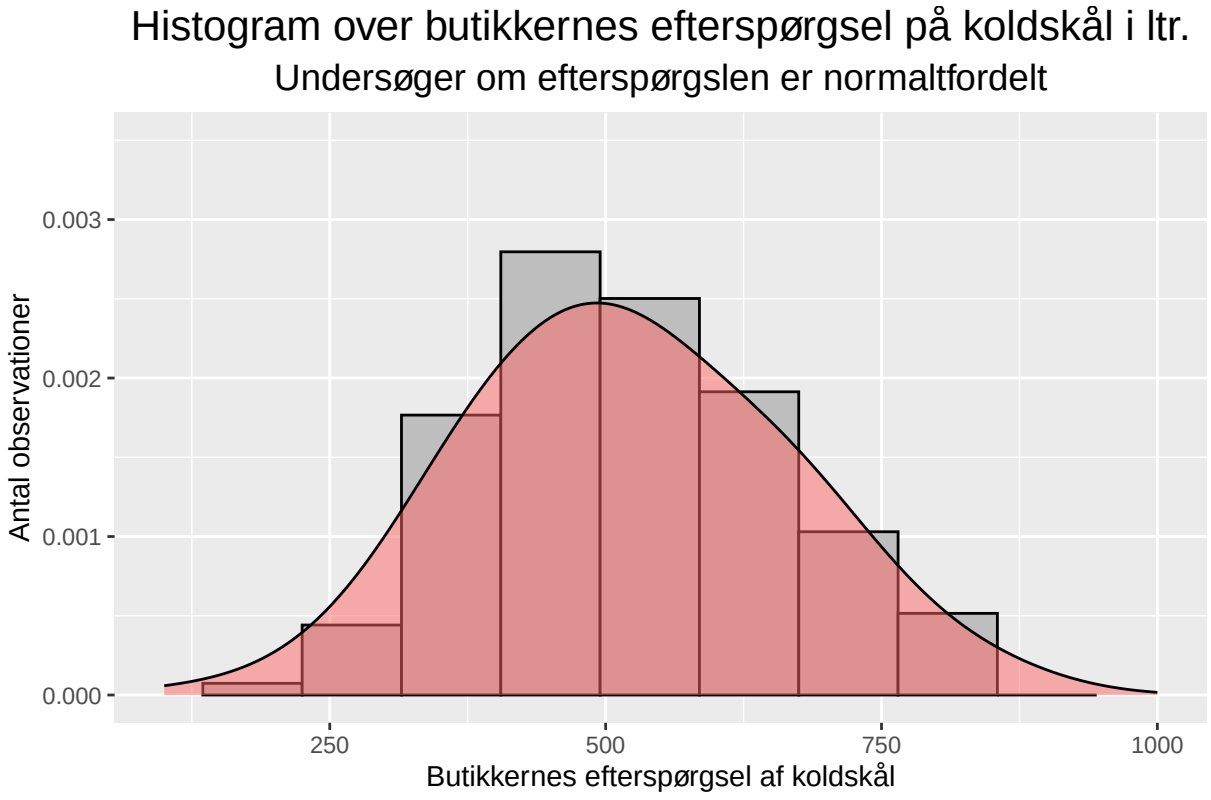
Først identificeres potentielle outliers i vores dataset. Derefter fjerner vi 1 outlier som er 47, da den skiller sig væsentligt ud i forhold til de andre observationer. Umiddelbart vurderes det ikke, at der er mange outliers i vores data som kan have indflydelse på den samlede varians, hvorfor vi kun har fjernet den ene. Tilbage er der n = 151 i vores datasæt.

Derefter laves der et ggplot for at se fordelingen af efterspørgselen af koldskål i form af simpelt histogram, fordi efterspørgslen er en kontinuert variabel. Det er derfor muligt, at beregne spredningen mellem observationerne.

`geom_density()` funktionen bruges til, at forstå fordelingen og til at forudsige den forventede fordeling af efterspørgslen på koldskål. Man kan se at at spredningen af observationerne er størst omkring 500. Endvidere kan det ses, at efterspørgslen af koldskål er tilnærmelsesvis normalfordelt, og at sandsynlighedskurven er symmetrisk klokkeformet.

Dog kan man også se, at nogle af observationerne falder udenfor, hvilket kan skyldes tilfældig variation eller systematiske fejl. Man ved derfor, at ca. 50% af observationerne befinder sig til venstre og højre af midtpunktet, dette er middelværdien. At vores data er normalfordelt er en fordel, fordi den lineære regressionsmodel er en parametrisk test, hvor kravet er at data er normaltfordelt.

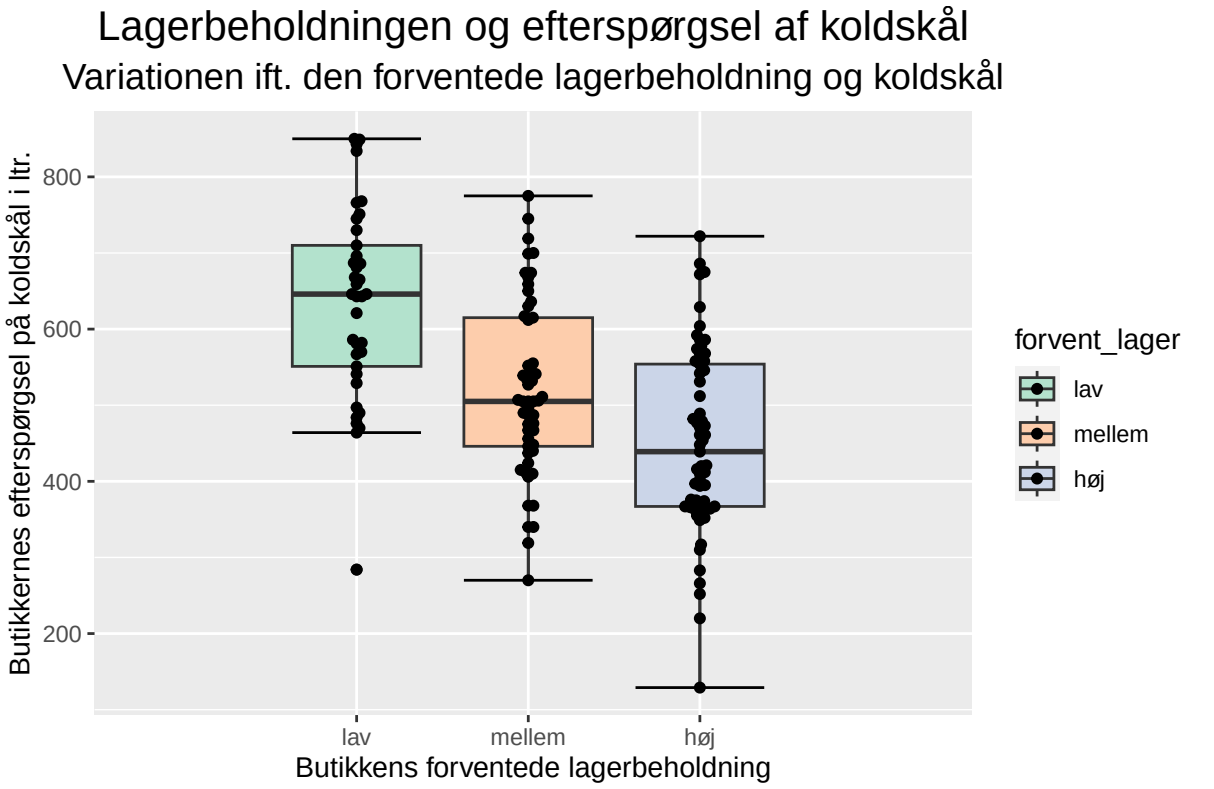
```r
data3 <- data3 %>%
  filter(efterspørgsel > 47) # Fjerner obs. 47 fra datasættet.


ggplot(data3, aes(x = efterspørgsel)) +
 geom_histogram(aes(y = ..density..), color = "black",
                fill = "grey", binwidth = 90) +
 geom_density(alpha = 0.5, fill="#FF6666", adjust = 1.6) +
  labs(title = "Histogram over butikkernes efterspørgsel på koldskål i ltr.",
       subtitle = "Undersøger om efterspørgslen er normaltfordelt",
       y = "Antal observationer",
       x = "Butikkernes efterspørgsel af koldskål",
       caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22") +
  ggeasy::easy_center_title() + # Centrerer titlen.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 1, face = "italic", size = 10)) +
  xlim(100, 1000) + ylim(0, 0.0035)
```



I følgende kode-chunk har er der lavet et boxplot til at vise den statistiske variationen ift.butikkens forventede lagerbeholdning og efterspørgslen på koldskål.

Her kan man se at median-efterspørgslen stiger når man går fra høj til lav forventet lagerbeholdning af koldskål. Dette tyder også på at der er en signifikant sammenhæng mellem de 2 variabler. Man kan også se, at der er fx. ved en høj forventet lagerbeholdning er en relativ stor usikkerhed i forhold til mellem og lav lagerbeholdning, dette indikerer at datapunkterne er en del spredt ud.

```
ggplot(data = data3, mapping = aes(x = forvent_lager, y = efterspørgsel, fill =
                                     forvent_lager)) +
  stat_boxplot(geom = 'errorbar') + # Undersøger usikkerheden og spredningen.
  geom_boxplot() +
  labs(title = "Lagerbeholdningen og efterspørgsel af koldskål",
       subtitle = "Variationen ift. den forventede lagerbeholdning og koldskål",
       caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
       y = "Butikkernes efterspørgsel på koldskål i ltr.",
       x = "Butikkens forventede lagerbeholdning") +
  geom_beeswarm(dodge.width=3, cex = 1, color = "black") + # undgår overplot.
  ggeasy::easy_center_title() +
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 2.4, face = "italic",size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```



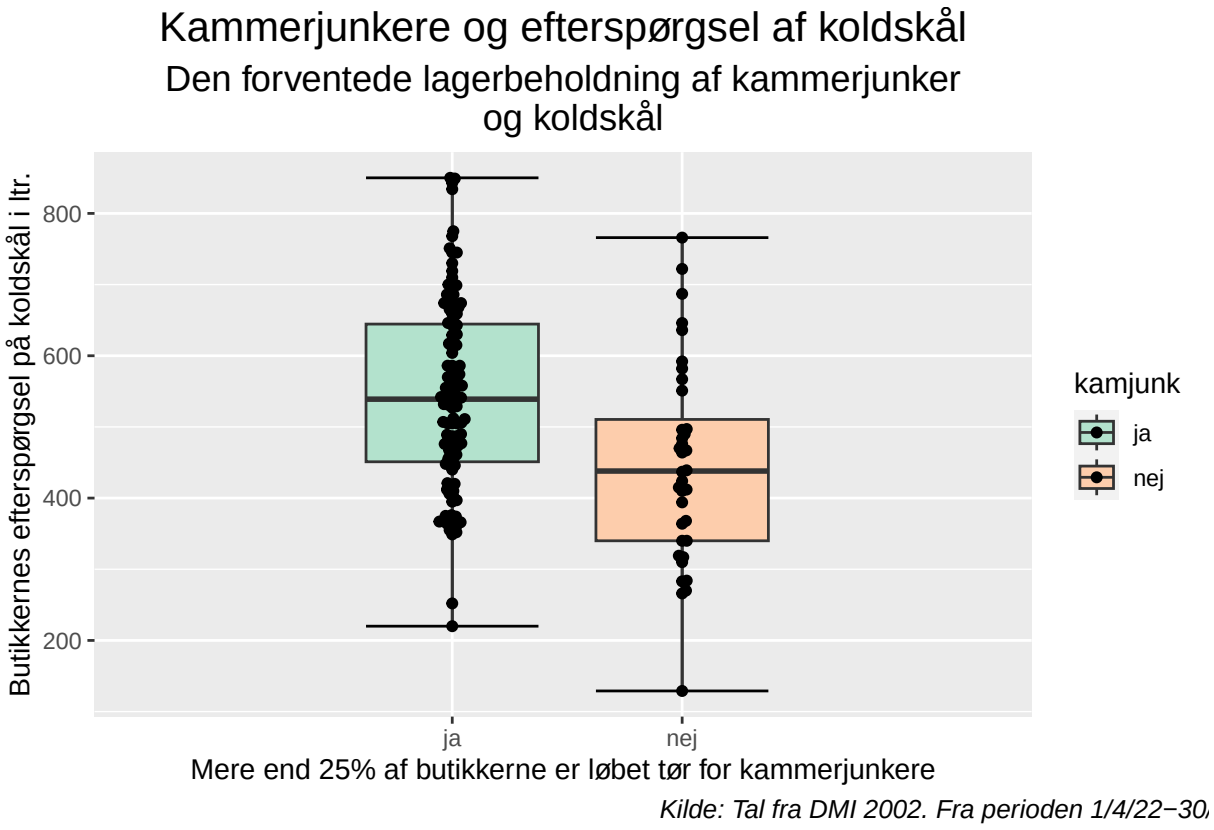
Kilde: Tal fra DMI 2002. Fra perioden 1/4

I næste kode-chunk er der lavet et boxplot som viser fordelingen af efterspørgslen i forhold til om 25% af butikkerne er løbet tør for kammerjunkere eller ej.

På baggrund af plottet kan man se at hvis butikkerne ikke har kammerjunkere på lageret så falder efterspørgslen. Det betyder at Efterspørgslen på koldskål stiger når de er løbet tør for kammerjunkere.

```
ggplot(data = data3, mapping = aes(x = kamjunk, y = efterspørgsel, fill =
                                     kamjunk)) +
  stat_boxplot(geom = 'errorbar') +
  geom_boxplot() +
  labs(title = "Kammerjunkere og efterspørgsel af koldskål",
  subtitle = "Den forventede lagerbeholdning af kammerjunker
  og koldskål",
  caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
  y = "Butikkernes efterspørgsel på koldskål i ltr.",
  x = "Mere end 25% af butikkerne er løbet tør for kammerjunkere") +
  ggeasy::easy_center_title() + # Centrerer titlen.
```

```
  geom_beeswarm(dodge.width=3,cex=1, color = "black") + # Justerer boksbredden.
  theme( plot.title = element_text(hjust = 0.5, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 1.7, face = "italic",
                                     size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```

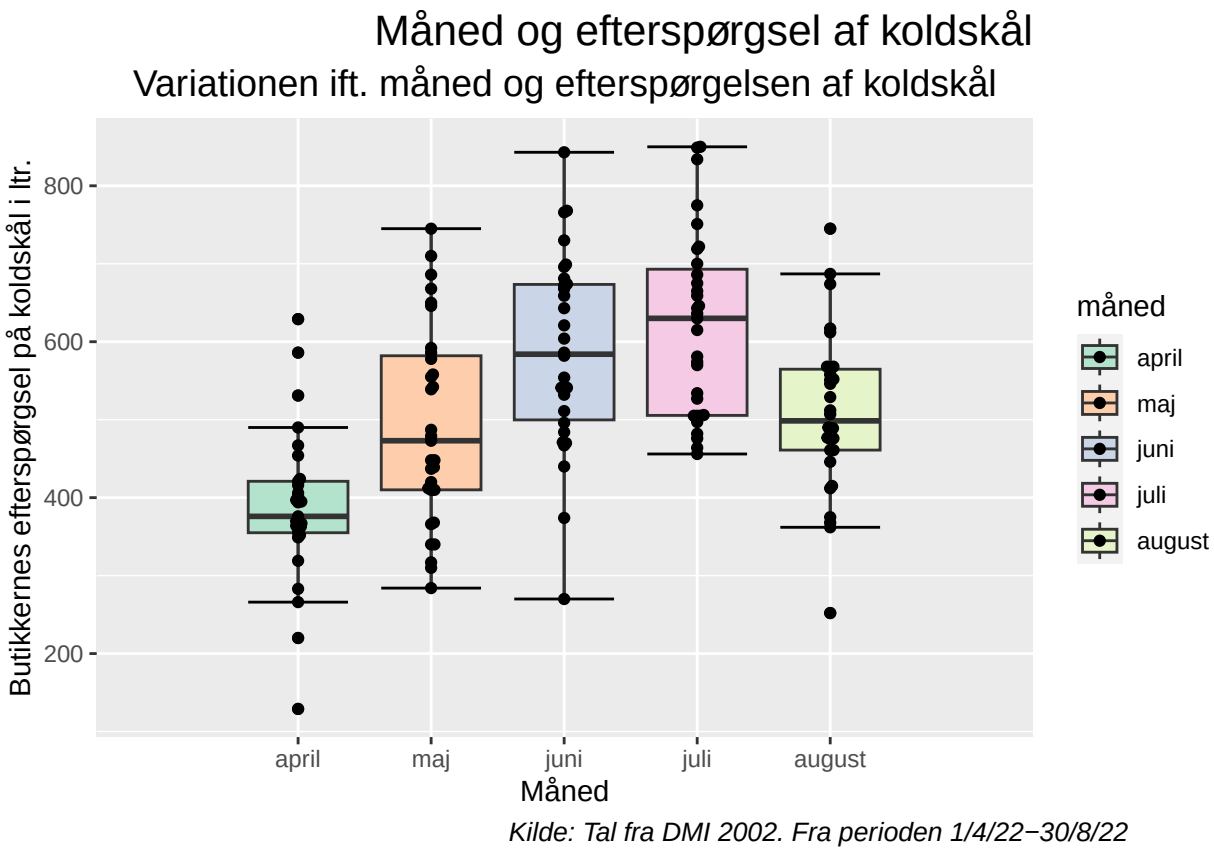


Forneden har vi lavet et boxplot som viser sammenhængen mellem måned og efterspørgslen af koldskål. Det er tydeligt at se at median-efterspørgslen stiger fra april-juli hvorefter den falder efterspørgslen i august.

Denne observation stemmer også overens med påstande fra vores kilder i problemfeltet, og udsagn fra vores interviews med medarbejderne hos Thise Mejeri. Hvilket indikerer at efterspørgselen på koldskål hænger moderat sammen med årstiden, dvs. selve sommerperioden.

```
ggplot(data = data3, mapping = aes(x = måned, y = efterspørgsel,
                                   fill = måned)) +
  stat_boxplot(geom = 'errorbar') +
```

```
  geom_boxplot() +
  labs(title = "Måned og efterspørgsel af koldskål",
       subtitle = "Variationen ift. måned og efterspørgelsen af koldskål",
       caption = "Kilde: Tal fra DMI 2002. Fra perioden 1/4/22-30/8/22",
       y = "Butikkernes efterspørgsel på koldskål i ltr.",
       x = "Måned") +
  ggeasy::easy_center_title() + # Centrerer titlen.
  geom_beeswarm(dodge.width = 3,cex = 0.5, color = "black") + # Justerer boksbredden.
  theme( plot.title = element_text(hjust = 1, size = 16),
         plot.subtitle = element_text(hjust = 0.5, size = 14),
         plot.caption = element_text(hjust = 1.3, face =
                                       "italic", size = 10 )) +
  scale_fill_brewer(palette = "Pastel2")
```



I nedestående kodechunk er der udvalgt en række kontinuerte variabler, fordi ønskes er, at undersøge om disse korrelerer med hinanden, og om deres indbyrdes korrelation er statistisk signifikant. Der anvendes en `chart.Correlation()` funktion til, at foretage en

korrelationsanalyse.

Efterspørgel og humidity er ikke korreleret med hinanden, og dermed ikke statistisk signifikant. Beslutningen er derfor, at humidity ikke vil blive inkluderet i analysen. Efterspørgsel og den gennemsnitlige temperatur per time har en korrelations koefficient på 0.38. P-værdien er meget lav med tre stjerner, det betyder at sammenhængen er signifikant, og det er derfor usandsynligt at opnå et mere ekstremt resultat, hvis man indsamlede en ny stikprøve igen og igen.

Alle temperatur variablerne er tæt på 1, hvilket betyder at de har stærk samvariation. Dette kaldes for multikolinearitet. Det vil sige, hvis de blev brugt i den endelige model ville det være vanskeligt, at fortolke på koefficienterne. Fordi en Model med høj multikolinearitet bliver mindre præcis og mindre pålidelig.

```
cor_matrice <- data3 %>%
  dplyr::select(efterspørgsel,
                temp_min_past1h,
                temp_dry,
                temp_max_past1h,
                humidity)
chart.Correlation(cor_matrice, histogram = TRUE, method = "pearson")
```

Som førnævnt var der en moderat signifikant sammenhæng mellem efterspørgslen og gennemsnits-temperaturen. Derfor har vi valgt at bruge denne variabel som den uafhængige effekt i næste kodechunk. Som førnævnt var der en moderat signifikant sammenhæng mellem efterspørgslen og gennemsnits-temperaturen.

Efterspørgslen af koldskål bliver prædiktet ud fra den gennemsnitlige temperatur hver time i °C. Først ved 10, 20 og 30 grader. Den grå linje omkring tendenslinjen referer til konfidensintervallet af den gennemsnitlige efterspørgsel på koldskål ved en given temperatur. Mange af observationerne er placeret udenfor dette bånd, hvorfor det er besluttet at anvende et prædiktionsinterval i stedet - der er den røde stiplede linje. Formålet er med andre ord, at indfange usikkerheden omkring de individuelle værdier og ikke usikkerheden omkring gennemsnittet.

Når den gennemsnitlige temperatur hver time er 10 °C, forudsiger vi at efterspørgslen på

koldskål for én ny observation være 440.64 liter. Ved samme temperatur vil efterspørgslen med 95% sikkerhed være [181.78:699.51].

Når den gennemsnitlige temperatur hver time er 20 °C, forudsiger vi at efterspørgslen på koldskål for én ny observation være 535.25 liter. Ved samme temperatur vil butikkernes efterspørgslen af koldskål med 95% sikkerhed være [278.23:792.28].

Når den gennemsnitlige temperatur hver time er 30 °C, forudsiger vi at efterspørgslen på koldskål for én ny observation være 629.87 liter. Ved samme temperatur vil butikkernes efterspørgslen af koldskål med 95% sikkerhed være [369.30:890.43].

Man kan på baggrund af ovenstående tydeligt se, at når den gennemsnitlige temperatur i °C stiger, så stiger butikkernes efterspørgsel på koldskål tilsvarende.

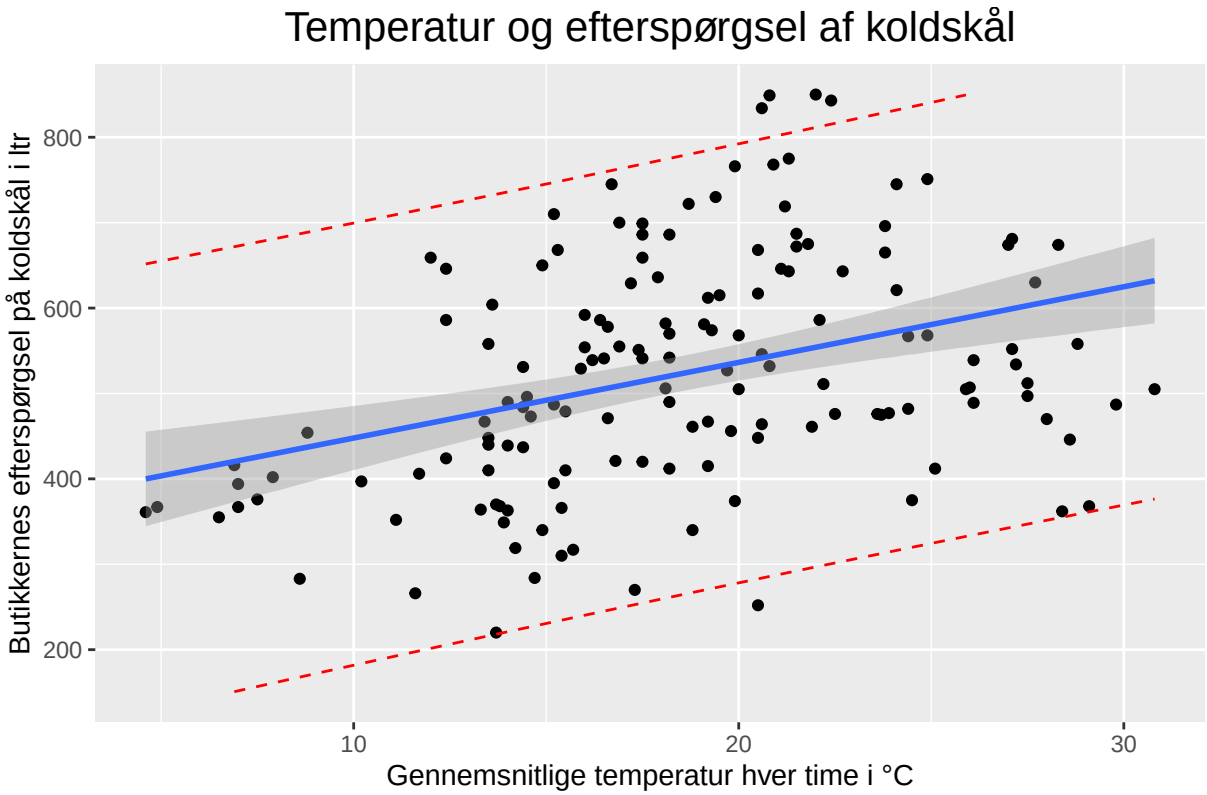
```
model1 <- lm(efterspørgsel ~ temp_mean_past1h, data = data3)
predict(model1,data.frame(temp_mean_past1h = (c(10,20,30))), interval = "prediction",
```

```
##        fit      lwr      upr
## 1 440.6455 181.7817 699.5094
## 2 535.2564 278.2290 792.2838
## 3 629.8672 369.3044 890.4301
```

```
prædiktion <- predict(model1, interval = "prediction", level = 0.95)
ny_df <- cbind(data3, prædiktion)
ggplot(ny_df, aes(temp_mean_past1h, efterspørgsel)) +
    geom_point() +
    geom_line(aes(y=lwr), color = "red", linetype = "dashed") +
    geom_line(aes(y=upr), color = "red", linetype = "dashed") +
    geom_smooth(method = lm, se = TRUE) +
    labs(title = "Temperatur og efterspørgsel af koldskål",
    caption = "Kilde: tal fra DMI 2002 fra perioden 1/4/22-30/8/22",
    y = "Butikkernes efterspørgsel på koldskål i ltr",
    x = "Gennemsnitlige temperatur hver time i °C") +
ggeasy::easy_center_title() + # Centrerer titlen.
theme( plot.title = element_text(hjust = 0.5, size = 16),
```

```
plot.subtitle = element_text(hjust = 0.5, size = 10),
plot.caption = element_text(hjust = 1, face = "italic", size = 10 ))+
xlim(4.6, 30.8) + ylim(150, 850)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Kilde: tal fra DMI 2002 fra perioden 1/4/22−30/8/22

Træning på træningsdata

Da antallet af variabler i det samlede datasæt er mindre end antallet af observationer, benyttes backward-selection til, at udvælge de uafhængige variabler som fremadrettet skal indgå i modellerne. Det vil sige, at vi tilføjer alle variable ind på højre side af ligningen, og fjerner dem med den højeste p-værdi indtil der kun er signifikante uafhængige variable tilbage (Hastie et.al 2021).

Vi træner først vores model på vores træningsdata, fordi vi gerne vil tilpasse modelparametrene. Vi bruger træningsdata til, at fintune vores regressionsmodel. Når modellen er blevet trænet godt igennem, bliver den afprøvet på testdata, da vi gerne vil undersøge hvor god modellen er til, at forudsige en så præcis efterspørgsel på koldskål som mulig. Vurderingen af modelpræcisionen bestemmes ud fra den laveste MSE værdi. MSE måler hvor langt den

forudsagte værdi for en observation er fra den faktiske værdi for en observation. Er MSE lille er der den forudsagte værdi tæt på den faktiske værdi, er MSE stor er den forudsagte værdi langt fra den faktiske værdi. MSE er således et udtryk for, hvor præcis den udvalgte model er til, at forudsige efterspørgslen af koldskål (Hastie et.al 2021).

Baseline modellen er den simpleste uden uafhængige variabler. Her er den gennemsnitlige efterspørgsel den afhængige variabel. Baselinemodellen har MSE = 19.376. Det er meget langt fra 0. Den simple model har en MSE på 16.576. Den er bedre end baselinemodellen fordi vi har øget kompleksiteten ved, at tilføje temp_mean_past1h.

```
# Baseline model


lm.fit_træning <- lm(efterspørgsel ~ 1, data = data3) # Baseline model
lm_fit1.1_summary <- summary(lm.fit_træning)
mean(lm_fit1.1_summary$residuals^2) # MSE 19376.55
```

```
## [1] 19376.55
```

```
rmse(lm.fit_træning, data = data3)  # RMSE = 139.19
```

```
## [1] 139.1997
```

```
lm.fit_træning
```

```
##
## Call:
## lm(formula = efterspørgsel ~ 1, data = data3)
##
## Coefficients:
## (Intercept)
##       520.2
```

```r
# Simpel model

lm.fit2_træning <- lm(efterspørgsel ~ temp_mean_past1h, data = data3)
lm_fit2_summary <- summary(lm.fit2_træning)
mean(lm_fit2_summary$residuals^2) #  MSE = 16576.45
```

```
## [1] 16576.45
```

```r
rmse(lm.fit2_træning, data = data3)  # RMSE = 128.74
```

```
## [1] 128.7495
```

```r
lm_fit2_summary
```

```
##
## Call:
## lm(formula = efterspørgsel ~ temp_mean_past1h, data = data3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -305.02  -92.77  -11.28   84.39  306.18
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       346.035     36.288   9.536  < 2e-16 ***
## temp_mean_past1h    9.461      1.886   5.017 1.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 129.6 on 149 degrees of freedom
## Multiple R-squared:  0.1445, Adjusted R-squared:  0.1388
## F-statistic: 25.17 on 1 and 149 DF,  p-value: 1.476e-06
```

```r
# Mellem model

lm.fit3_træning = lm(efterspørgsel ~ forvent_lager + weekend_helligdag + kamjunk + te
lm_fit3_summary <- summary(lm.fit3_træning)
mean(lm_fit3_summary$residuals^2) # MSE = 6740.342
```

```
## [1] 6740.342
```

```r
rmse(lm.fit3_træning, data = data3) # RMSE = 82.09959
```

```
## [1] 82.09959
```

```r
lm_fit3_summary
```

```
##
## Call:
## lm(formula = efterspørgsel ~ forvent_lager + weekend_helligdag +
##     kamjunk + temp_gt25_3_dage + måned + I(temp_mean_past1h),
##     data = data3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.464  -57.822    6.436   62.564  202.505
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          379.923     40.788   9.314 2.35e-16 ***
## forvent_lagermellem  -76.565     19.482  -3.930 0.000133 ***
## forvent_lagerhøj     -82.672     22.579  -3.661 0.000355 ***
## weekend_helligdag1   113.010     15.007   7.530 5.66e-12 ***
## kamjunknej           -71.890     17.507  -4.106 6.80e-05 ***
## temp_gt25_3_dage     -81.992     34.237  -2.395 0.017951 *
```

```
## månedmaj              84.369     24.541   3.438 0.000773 ***
## månedjuni            129.512     30.656   4.225 4.29e-05 ***
## månedjuli            155.947     32.790   4.756 4.85e-06 ***
## månedaugust           85.101     34.768   2.448 0.015616 *
## I(temp_mean_past1h)    4.249      2.096   2.028 0.044475 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.26 on 140 degrees of freedom
## Multiple R-squared:  0.6521, Adjusted R-squared:  0.6273
## F-statistic: 26.25 on 10 and 140 DF,  p-value: < 2.2e-16
```

```
# Kompleks model


lm.fit4_træning = lm(efterspørgsel ~ forvent_lager + weekend_helligdag + kamjunk + te
lm_fit4_summary <- summary(lm.fit4_træning)
mean(lm_fit4_summary$residuals^2) # MSE = 6923.087
```

```
## [1] 6923.087
```

```
rmse(lm.fit4_træning, data = data3) # RMSE = 83.20509
```

```
## [1] 83.20509
```

```
lm_fit4_summary
```

```
##
## Call:
## lm(formula = efterspørgsel ~ forvent_lager + weekend_helligdag +
##     kamjunk + temp_gt25_3_dage + måned + I(temp_mean_past1h^22),
##     data = data3)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -228.715  -55.834    9.064   60.411  195.816
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              4.348e+02  3.118e+01  13.946  < 2e-16 ***
## forvent_lagermellem     -7.456e+01  1.988e+01  -3.751 0.000257 ***
## forvent_lagerhøj        -8.700e+01  2.279e+01  -3.818 0.000202 ***
## weekend_helligdag1       1.073e+02  1.511e+01   7.103 5.66e-11 ***
## kamjunknej              -7.652e+01  1.763e+01  -4.340 2.71e-05 ***
## temp_gt25_3_dage        -6.674e+01  3.429e+01  -1.946 0.053643 .
## månedmaj                 1.017e+02  2.326e+01   4.373 2.38e-05 ***
## månedjuni                1.627e+02  2.625e+01   6.199 5.98e-09 ***
## månedjuli                1.974e+02  2.648e+01   7.457 8.45e-12 ***
## månedaugust              1.329e+02  2.659e+01   4.999 1.69e-06 ***
## I(temp_mean_past1h^22) -7.542e-32  1.359e-31  -0.555 0.579766
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 86.41 on 140 degrees of freedom
## Multiple R-squared:  0.6427, Adjusted R-squared:  0.6172
## F-statistic: 25.18 on 10 and 140 DF,  p-value: < 2.2e-16
```

Test på testdata

I fornævnte afsnit blev den gennemsnitlige MSE beregnet for hver af de fire modeller på træningsdata. Vi er egentlig ligeglade med disse MSE værdier, det er mere interessant at se hvor præcise forudsigelserne er på testdata. Træningsdata anvendes som førnævnt til, at udvælge signifikante uafhængige variable og tilpasse modelparametrene.

Dog er det værd at nævne, at den data undersøgelsen er baseret på er simuleret. Det vil sige, at f på forhånd er kendt. Den virkelige og yderst komplekse sandhed om efterspørgslen af koldskål ved vi ikke. Men hvis der bliver udtrukket nogle testdata ud fra data3, kan man validere hvor godt modellerne performer på disse testdata, såfremt modelkompleksiteten

øges. Kompleksiteten kan øges ved, at de kontinuerte variable opløftes i flere potenser, eller ved og inkludere flere uafhængige variabler i modellerne.

Man kan producere testdata på flere måder. Der anvendes en LOOCV metode, fordi data3 kun indeholder 151 observationer i alt. Det gode ved denne fremgangsmåde er, at den træner på alle datapunkterne undtagen ét datapunkt. Processen gentages i dette tilfælde 150 gange. Derefter beregnes en gennemsnitlig score er beregnet, denne udtrykker hvor god modellen klare sig (dvs. modellen med lavest MSE vælges) (Hastie et.al 2021).

Problemet med metoden er, at den kræver stor computerkraft. Det tog denne bærbare computer ca. 2 minutter hver gang koden stumpen blev kørt. Det skyldes, at modellen bliver trænet k gange (ibid).

Modeludvælgelse og test på testdata

	Baseline	Simpel	Mellem	Kompleks
Forvent_lager(mellem)			−76.57***	−74.55***
			{19.82}	
Forvent_lager(høj)			−82.67***	−87.00***
			{22.58}	
Weekend_helligdag(ja)			113.01***	107.33***
			{15.00}	
Kamjunk(nej)			−71.89***	−76.52***
			{17.50}	
Temp_gt25_3_dage			−82.00*	−66.74*
			{34.23}	
Måned(maj)			84.37***	101.69***
			{24.54}	
Måned(juni)			129.51***	162.71***
			{32.79}	
Måned(juli)			155.95***	155.947***
			{32.79}	
Måned(august)			85.10*	197.44*
			{34.76}	

	Baseline	Simpel	Mellem	Kompleks
Temp_mean_past1h		9.46***	4.249*	−0.0
		{1.89}	{2.07}	
Uafhængige variable	0	1	6	6
Skæring	520.23***	346.04***	379.93***	434.78***
Model P-værdi	* * *	* * *	* * *	* * *
MSE_træning	139.20	128.74	82.10	83.21
MSE_test	139.20	130.36	88.55	89.37
R^2		0.15	0.65	0.64
RSE	139.7	129.6	85.26	86.41
Obs	151	151	151	151

Tabel 1. Summeret modelreferat fra testdata. Referencegrupper () for faktorerne er: kamjunkja, forvent_lagerlav, månedapril. {} referer til standardfejlen. Note:$* = P < 0.1; ** = P < 0.05; *** = P < 0.01$

```
# Baseline model

lm.fit_træning <- lm(efterspørgsel ~ 1, data = data3) # Baseline model
lm_fit1.1_summary <- summary(lm.fit_træning)
mean(lm_fit1.1_summary$residuals^2) # MSE 19376.55
```

```
## [1] 19376.55
```

```
rmse(lm.fit_træning, data = data3)  # RMSE = 139.19
```

```
## [1] 139.1997
```

```
lm.fit_træning
```

```
##
## Call:
```

```
## lm(formula = efterspørgsel ~ 1, data = data3)
##
## Coefficients:
## (Intercept)
##       520.2
```

```
# Simpel model

ctrl <- trainControl(method = "LOOCV") # Udvælger cross-validation metode
model1_test <- train(efterspørgsel ~ temp_mean_past1h, data = data3, method = "lm", t
model1_test # RMSE 130.36
```

```
## Linear Regression
##
## 151 samples
##   1 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   130.3607  0.1238588  105.8906
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
summary(model1_test)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -305.02  -92.77  -11.28   84.39  306.18
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       346.035     36.288   9.536  < 2e-16 ***
## temp_mean_past1h    9.461      1.886   5.017 1.48e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 129.6 on 149 degrees of freedom
## Multiple R-squared:  0.1445, Adjusted R-squared:  0.1388
## F-statistic: 25.17 on 1 and 149 DF,  p-value: 1.476e-06
```

```
# Mellem model

model2_test <- train(efterspørgsel ~ forvent_lager + weekend_helligdag + kamjunk + te
model2_test # RMSE 88.55
```

```
## Linear Regression
##
## 151 samples
##   6 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   88.55084  0.5967607  72.56083
##
```

```
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
summary(model2_test)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.464  -57.822    6.436   62.564  202.505
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          379.923     40.788   9.314 2.35e-16 ***
## forvent_lagermellem  -76.565     19.482  -3.930 0.000133 ***
## forvent_lagerhøj     -82.672     22.579  -3.661 0.000355 ***
## weekend_helligdag1   113.010     15.007   7.530 5.66e-12 ***
## kamjunknej           -71.890     17.507  -4.106 6.80e-05 ***
## temp_gt25_3_dage     -81.992     34.237  -2.395 0.017951 *
## månedmaj              84.369     24.541   3.438 0.000773 ***
## månedjuni            129.512     30.656   4.225 4.29e-05 ***
## månedjuli            155.947     32.790   4.756 4.85e-06 ***
## månedaugust           85.101     34.768   2.448 0.015616 *
## temp_mean_past1h       4.249      2.096   2.028 0.044475 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85.26 on 140 degrees of freedom
## Multiple R-squared:  0.6521, Adjusted R-squared:  0.6273
## F-statistic: 26.25 on 10 and 140 DF,  p-value: < 2.2e-16
```

```r
# Kompleks model

model3_test <- train(efterspørgsel ~ forvent_lager + weekend_helligdag + kamjunk + ter
model3_test # RMSE 89.61
```

```
## Linear Regression
##
## 151 samples
##   6 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   89.56104  0.5875517  73.06976
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
summary(model3_test)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -221.256  -56.598    6.328   62.528  200.854
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)             429.119609  31.672090  13.549  < 2e-16 ***
## forvent_lagermellem     -76.537449  19.725445  -3.880 0.000160 ***
## forvent_lagerhøj        -85.329967  22.818829  -3.739 0.000268 ***
## weekend_helligdag1      110.172053  15.161385   7.267 2.36e-11 ***
## kamjunknej              -74.260703  17.719015  -4.191 4.89e-05 ***
## temp_gt25_3_dage        -75.688080  34.908239  -2.168 0.031833 *
## månedmaj                 99.704921  23.368071   4.267 3.63e-05 ***
## månedjuni               154.527876  27.871703   5.544 1.42e-07 ***
## månedjuli               185.559489  29.127768   6.371 2.53e-09 ***
## månedaugust             117.611689  30.962316   3.799 0.000216 ***
## `I(temp_mean_past1h^3)`   0.001376   0.001584   0.869 0.386471
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 86.27 on 140 degrees of freedom
## Multiple R-squared:  0.6438, Adjusted R-squared:  0.6184
## F-statistic: 25.31 on 10 and 140 DF,  p-value: < 2.2e-16
```

På baggrund af vores MSE værdier, har vi opstillet et histogram der visuelt viser, hvordan MSE'en bliver mindre i takt med at modelkompleksiteten stiger. MSE er somsagt et udtryk for hvor godt en model yder på træningsdata. Men den MSE der bliver beregnet ud fra træningsdata kan være biased, derfor beregnes der en MSE på testdata vha. LOOCV metoden. Nævn bias variance-tradeoff her.

```
x <- data3$temp_mean_past1h
y <- data3$efterspørgsel
data <- data.frame(y, x)

ggplot(data3, mapping = aes(x=x, y=y)) +
  geom_point(alpha=1/3) +
  geom_smooth(method="glm", formula = y ~ poly(x, 1, raw=TRUE), se=FALSE, colour="blu
  geom_smooth(method="glm", formula = y ~ poly(x, 3, raw=TRUE), se=FALSE, colour="gre
  geom_smooth(method="glm", formula = y ~ poly(x, 22, raw=TRUE), se=FALSE, colour="re
```

```
geom_point(data=data3, mapping = aes(x=x, y=y), alpha=1/3) +
  labs(title = "Skæringen ved de tre polynomiske regressionsmodeller",
caption = "Kilde: Tal fra DMI 2002 fra perioden 1/4/22-30/8/22",
y = "Butikkernes efterspørgsel på koldskål i Ltr.",
x = "Gennemsnitlige temperatur pr.time i °C.") +
ggeasy::easy_center_title() + # Centrerer titlen.
theme( plot.title = element_text(hjust = 0.5, size = 16),
plot.subtitle = element_text(hjust = 0.5, size = 14),
plot.caption = element_text(hjust = 1, face = "italic", size = 10 ))+
xlim(5, 31) + ylim(220, 900) +
  theme_gray()
```

Tidy

Transformer

Visualiser

Model

Kommunikér/analyse

Sessioninformation

For at højne gennemsnigtigheden printes der en udskrift om den nuværende R session:

```
SI <- sessionInfo(package = NULL) # Udskriver en liste om denne R session.
```

Litteratur

Bilag