



Bistader

62527 Big data E20

Armandas Rokas(s185144)

22/11 2020

Indholdsfortegnelse

1	Problemformulering	1
2	Data beskrivelse	1
2.1	Import af data	1
2.2	Variabler	1
2.3	Databehandling	3
2.3.1	Vægt	3
2.3.2	Temperatur	5
3	Sammenligning af bistader	6
3.1	Formål	6
3.2	Hypotesetest	8
3.3	Konklusion på sammenligningen	8
4	Forudsigelse af vægten i vinterperioden	9
4.1	Formål	9
4.2	Modellen	9
4.3	Forudsigelse	9
4.4	Validering	10
5	Klassificering af vægttilvækst	11
5.1	Formål	11
5.2	Dataklargøring	11
5.3	Beslutningstræ	11
5.4	Validering	14
6	Konklusion	14
7	Referencer	15
8	Bilag	16
8.1	Bilag A: Opsummering af alle variabler i datasættet	16
8.2	Bilag B: Mine funktioner	16
8.3	Bilag C: Dataklargøring til klassificering af vægttilvækst	19

1 Problemformulering

I dette projekt skulle der arbejdes på dataen fra bistadet. Et helt overordnet formål ved dette er at hjælpe biavlens med driften af hans bifamilier. Mere præcist kommer projektet at udbrede sig i tre forskellige spektre. Først skal der sammenlignes to forskellige bistader fra den same have til at finde ud om den ene bifamilie præstere bedre end den anden. Bagefter forudsiges der vægten i vinterperiode for at forebygge evt. sult i bifamilien. Og til sidst klassificeres der væggtivækst i sommerperioden med et formål at advare biavlens, når en bifamilien evt. er syg.

2 Data beskrivelse

Før kommer vi i gang med datanalen, skal der lige først beskrive, hvordan dataen ser ud. Så for det første samles Dataen via Raspberry Pi, som er tilknyttet til forskellige sensorer, som måler bistadets vægt, fugtighed, lys osv. (hele listen af variabler kan man se i Variabler afsnit), og gemmes både lokalt på SQLite databasen og sendes til skyen på HiveTool.net platformen.

2.1 Import af data

Der blev valgt at bruge SQLite til at indlæse dataen til R, da det er lidt svært at få fat i dataen på HiveTool.net, især hvis man skal bruge en lang tidsperiode. For at indlæse dataen fra SQLite blev der skrevet en `import_hive_data` funktion, hvis koden kan findes i Bilag B.

2.2 Variabler

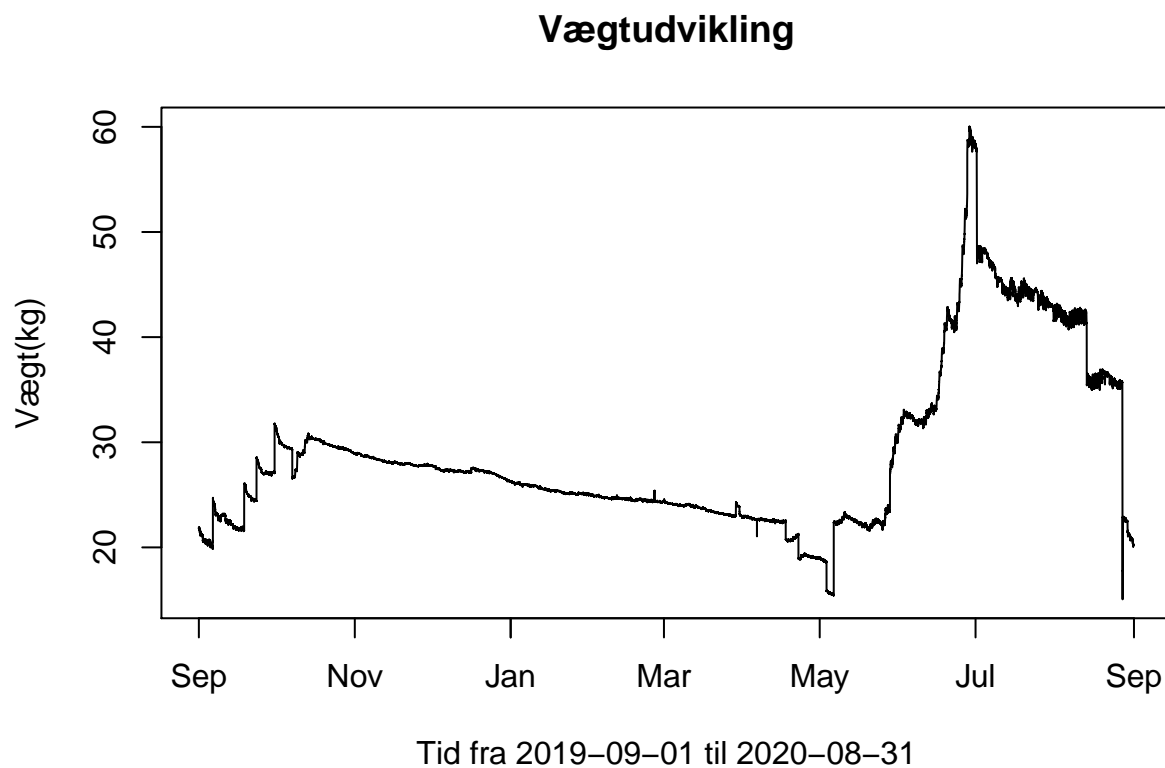
Observationer fra bistadet bliver taget hver 5. minut og der er i alt 105064 observationer (Bilag A). Dataen er dog ikke helt konsistent, da der er nogle huller i datasættet, hvor der mangler målinger, og der er nogle målinger, som har forkerte værdier. Dataen går helt tilbage til 2018-03-07, men den bliver mere og mere inkonsistente jo mere i fortiden man går, så derfor blev der valgt at tage udgangspunkt i et års data, dvs. fra 2019-09-01 til 2020-09-01.

Der er i alt 43 søjler i datasættet, men ud fra opsummering af variabler, som har nogen observationer, i Bilag A kan man konstatere at disse variabler kun er i brug og relevante: `hive_observation_time_local`, `hive_weight_kgs`, `hive_temp_c`, `hive_humidity`, `ambient_temp_c`, `ambient_humidity` og `ambient_luminance`. Nedenfor på Tabel 1 opsummeres der disse variabler:

Tabel 1: Opsummering af stadel data fra 2019-09-01 to 2020-09-01

	N	Missing	Mean	SD	Min	Q1	Median	Q3	Max
hive_weight_kgs	105064	0	28.94	7.98	15.06	23.65	26.97	30.36	60.04
hive_temp_c	105064	0	25.32	8.47	2.10	20.10	25.00	34.20	37.10
hive_humidity	105064	0	62.10	10.82	36.00	57.10	60.30	67.50	91.00
ambient_temp_c	101787	3277	9.50	6.21	-4.30	4.90	8.00	13.60	31.60
ambient_humidity	101787	3277	99.00	4.17	55.10	99.90	99.90	99.90	99.90
ambient_luminance	105064	0	14.65	56.36	0.00	0.00	0.00	5.00	2070.00

Vægten er den centrale variable i datasættet. Nedenfor på Figur 1 kan man se en vægtudvikling fra i perioden fra 2019-09-01 til 2020-09-01, hvor man kan se tydeligt to sæsoner: om vinteren vægten bliver gradvis mindre, fordi bierne spiser føder, mens om sommer stiger vægten kraftigt, fordi der er der, hvor bierne indsamler nektar.



Figur 1: Vægtudvikling på bistade1 fra 2019-09-01 til 2020-09-01

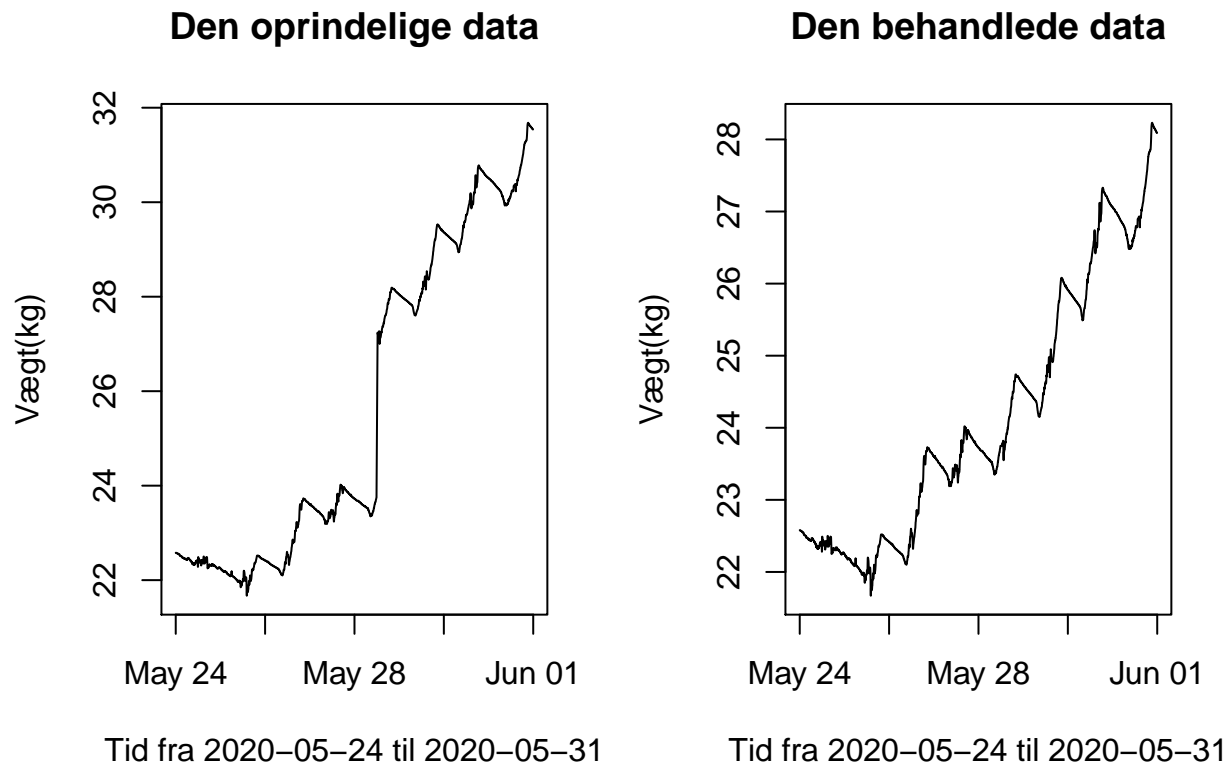
2.3 Databehandling

2.3.1 Vægt

Manual indgreb

Først og fremmest vægtmålinger skulle renses op for forkerte målinger. De er forkerte, fordi der blev udført flere gange “manual indgreb” på bistadet. Manuelt indgreb medfører at vægten pludselig stiger kraftig eller falder. Oftest det er fordi biavlens skal påsætte en ny magasin eller fjerne én. For at visualisere problemet blev der plottet vægten videre omkring 2020-05-28 12:25:01, hvor vægten har øget med 3.49, fordi der blev sat en nye magasin ind. Dette kan man se på den venstre graf på Figur 2.

Så derfor blev skrevet en `manipulate_weight_deltas` funktion, som udligner disse manualle indgreb (hele koden af funktionen kan findes i Bilag B). Funktionen tager imod et bistade data og perioderne, som skal udlignes. Den returnerer bagefter den behandlede data, som man kan se på den højre graf på Figur 2.

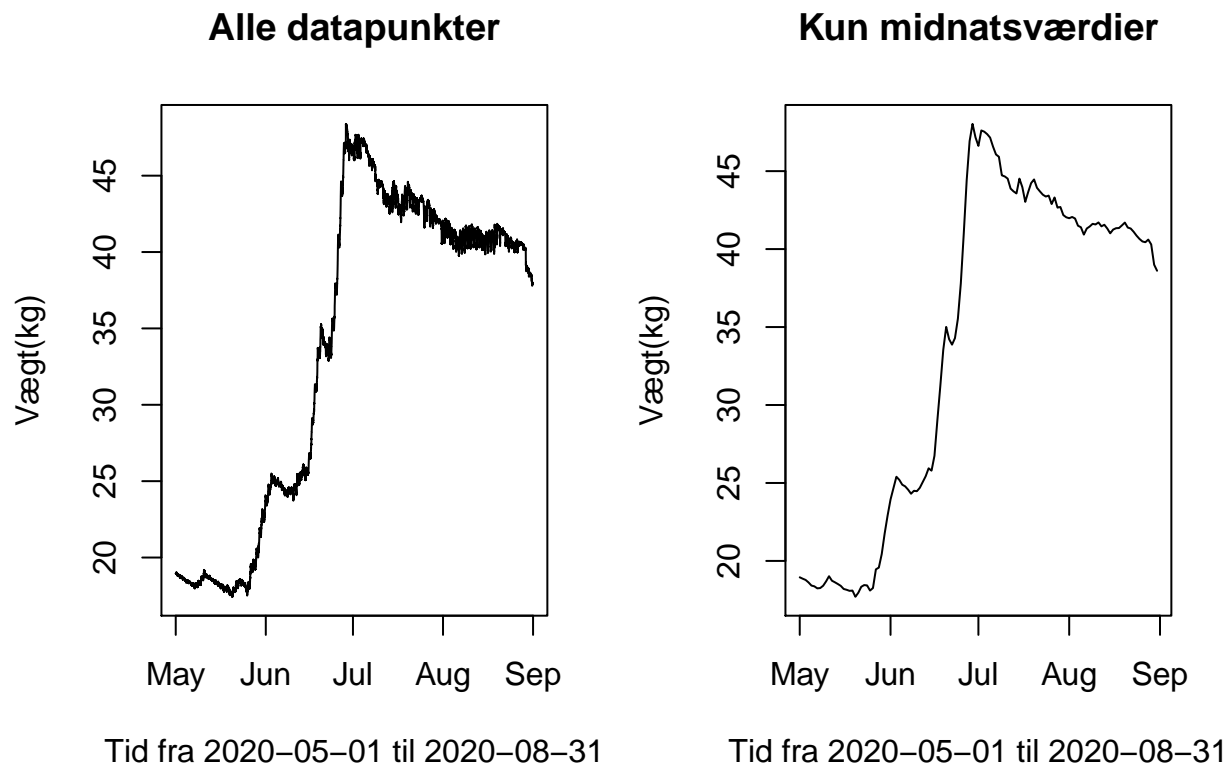


Figur 2: Vægtudvikling på bistade1 fra 2020-05-24 til 2020-06-01, hvor til venstre er den oprindelige data, og til højre den behandlede data, hvor et manuelt indgreb er fjernet.

Midnats vægtværdier

Den anden støj i vægtmålinger var, at der var alt for mange målinger, nemlig hvert 5. minut. Bierne følger den daglige rutine. Det flyver ud om morgen, indsamler nektar i løbet af dagen, om natten fordamper den. Det medfører at vægten svinger op og ned i løbet af dagen og for at se den ægte vægt tilvækst pr. dag er det bedst at kigge på midnatsværdier. Så der blev skrevet en `'import_hive_data_daily'` funktion, som trækker ud midnatsværdier (hele koden kan findes i Bilag B).

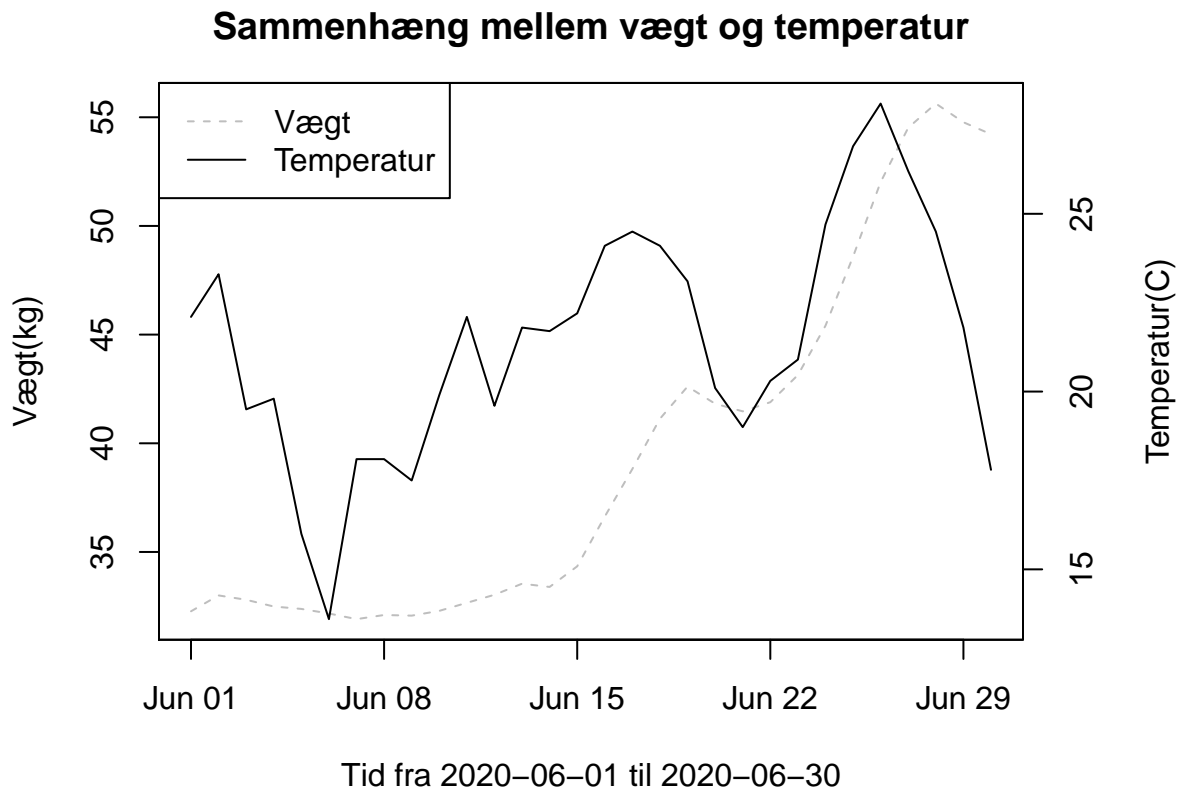
Nedenfor på Figur 3 er der to grafer. Til venstre er der alle data punkter, dvs. hvert 5. minut. Til højre er der kun midnatsværdier. Man kan se at det er nemmere aflæse grafen kun med midnatsværdier især i august måned, hvor vægten svinger mest i løbet af dagen.



Figur 3: Vægtudvikling på bistade1 fra 2020-05-01 til 2020-09-01, hvor til venstre er der en graf med alle datapunkter, og til højre kun med midnatsværdier.

2.3.2 Temperatur

Nu når der blev fået døgnvægtværdier, skulle der også findes døgntemperaturværdier, da de oprindelige målinger som nævnt er taget hvert 5. minut. Først blev der prøvet at finde den gennemsnitlige temperatur i døgnet, men værdierne var ikke så informative, da de lave nattemperaturer og de høje dagtemperaturer udlignede hinanden. Så der blev besluttet at finde den højeste værdi i døgnet. Hvordan det blev gjort, kan man se i Bilag B under `import_hive_data_daily` funktionen. På grafen i Figur 4 nedenfor kan man se de daglige højeste temperaturværdier og dets sammenhæng med midnatsvægtværdierne, hvor man kan fornemme noget forbindelse mellem disse to variabler, men dette undersøges videre i sektionen Klassificering af vægttilvækst.

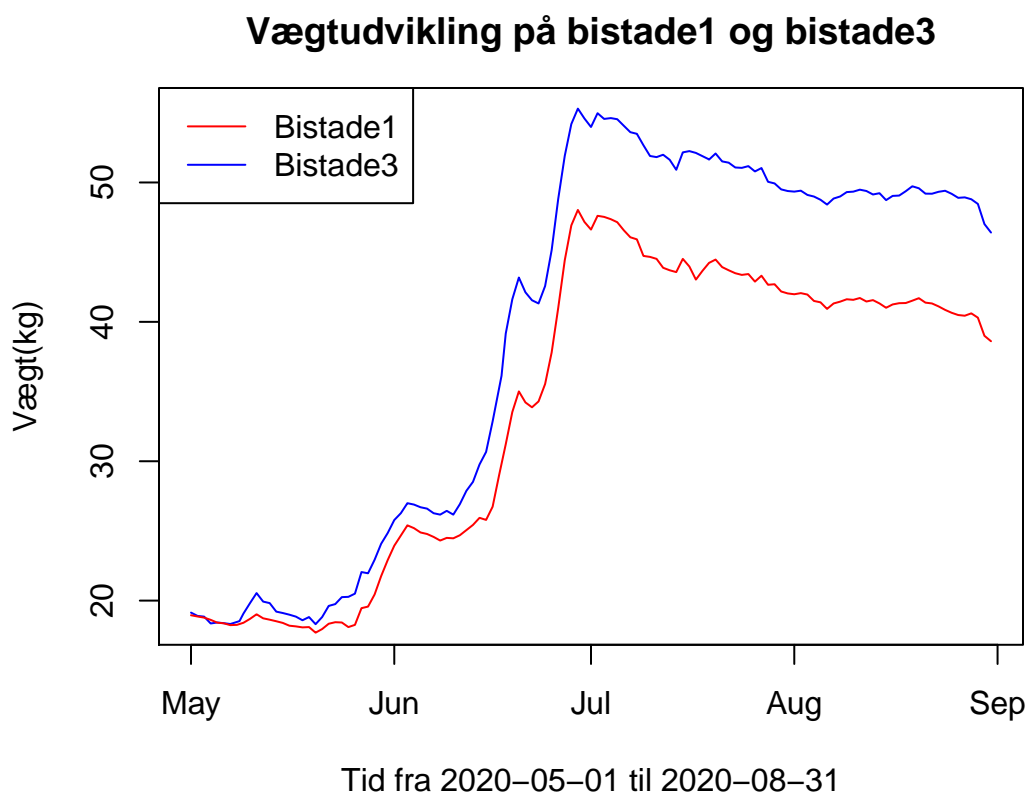


Figur 4: Sammenhæng mellem vægt og temperatur

3 Sammenligning af bistader

3.1 Formål

I dette afsnit skal der sammenlignes to bistader fra den samme have, altså Bistade1 og Bistade3. Formålet med sammenligning kunne være at vurdere om den ene familie præstere bedre end den anden. Nedenfor på Figur 5 kan man se vægtudvikling på disse bistader.



Figur 5: Vægtudvikling på bistade1 og bistade3 fra 2020-05-01 til 2020-08-31

Den daglige vægttilvækst

For at gøre det nemmere at sammenligne bistader, blev der taget de daglige vægttilvækst af disse bistader. Graffen af dette kan man se på Figur 6. Gennemsnittene af de dagligt vægttilvækst for de respektive bistader er:

- $\mu_1 = 0.1598374$ (Bistade1)
- $\mu_3 = 0.2188468$ (Bistade3)

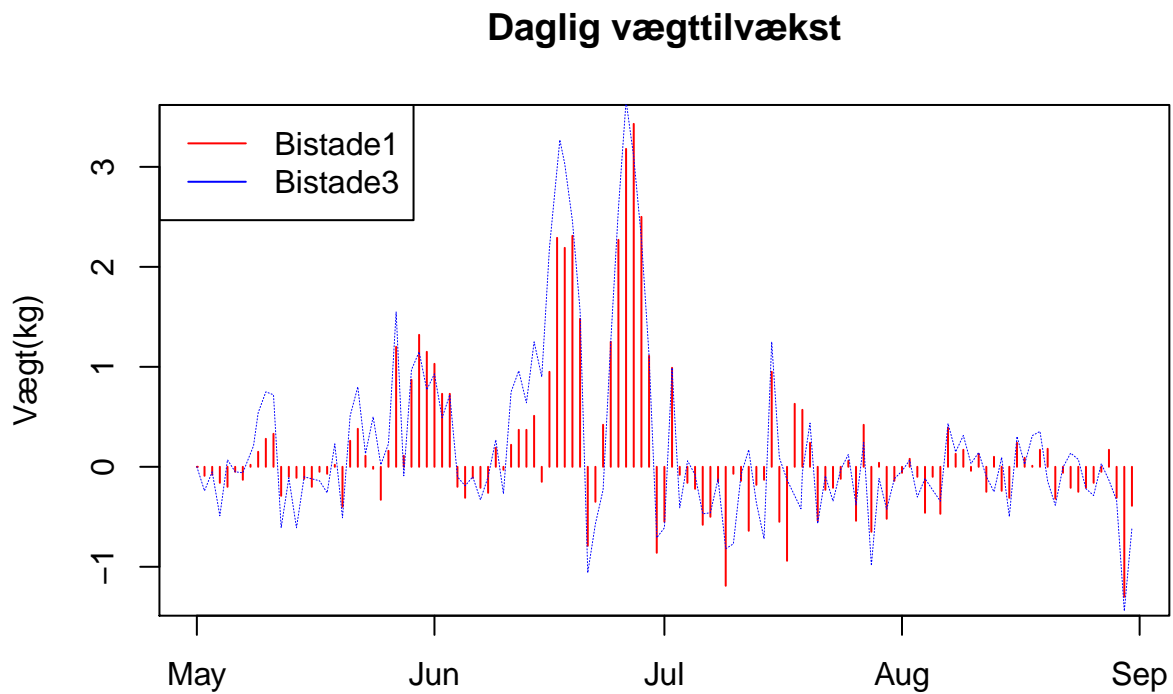
Disse blev udregnet med R koden nedenfor:

```
mean(hive1$daily_weight_delta)
```

```
## [1] 0.1598374
```

```
mean(hive3$daily_weight_delta)
```

```
## [1] 0.2216829
```



Figur 6: Daglig vægttilvækst af bistade1 og bistade3 fra 2020-05-01 til 2020-08-31

3.2 Hypotesetest

I den forudgående sektion blev der fundet at bistade3 umiddelbart har lidt højere gennemsnit, nemlig $\mu_3 - \mu_1 = 0.2188468 - 0.1598374 = 0.0590$, men er det nok at konkludere at der er en forskel mellem disse stader? For at undersøge det blev der udført et hypotesetest.

$$H_0 : \mu_1 = \mu_3$$

$$H_1 : \mu_1 \neq \mu_3$$

Welch Two Sample t-test

Nedenfor i outputtet fra `t.test` kan der ikke konstateres en signifikant forskel i den daglige vægttilvækst mellem Bistade1 og Bistade3, fordi p-value (sandsynlighed at vi har fået denne forskel tilfældigt) er ret stor 0.5763 og 95% konfidensinterval inkluderer 0. Dvs. vi ikke kan afvise H_0 , som er, at de daglige vægttilvækst er ens. (Brockhoff, Møller, Andersen, Bacher & Christiansen, 2018)

```
t.test(hive1$daily_weight_delta, hive3$daily_weight_delta)

##
##  Welch Two Sample t-test
##
## data:  hive1$daily_weight_delta and hive3$daily_weight_delta
## t = -0.58405, df = 240.5, p-value = 0.5597
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2704368  0.1467457
## sample estimates:
## mean of x mean of y
## 0.1598374 0.2216829
```

3.3 Konklusion på sammenligningen

Selv om der ikke kunne lade sig gøres at bevise signifikant forskellen på bistader ved at bruge `t.test`, er den forskellen som man kan sige på grafer er nok til biavlens til at konkludere, at der er en tydeligt eksempel på at bifamilier er forskellige, da stade3 producerer mest honning, og derfor ville han vælge Bistade3, som “mor” til en evt. ny familie.

4 Forudsigelse af vægten i vinterperioden

4.1 Formål

Der er to formål, hvorfor biavlens skal overvåge og evt. forudsige vægten i en vinter periode. Det første formål er, at det er ret vigtigt for biavlens, at vægten ikke falder under en vist niveau. Hvis den gør alligevel, så kunne dette betyde at bierne ikke har mere foder og derfor vil de begynde at sulte. Så det kunne være en fordel til biavlens, at han kunne forudsige, hvornår den falder under dette niveau, så biavlens kunne fylde foderdepoter op. For eksempel hvis vi tager Bistadel, så ligger dens minimumvægt på 15kg, da magasin, tavler og bier vejer omkring 12 kg.

Den anden formål kunne være at overvåge om familien har det godt om vinteren. Familien er god stand, hvis vægten falder konstant og som forventet, så det betyder at familien forbruger foder, men hvis vægten stopper med at falde, så det kan være en tegn, at familien er syg eller i det værste tilfælde uddødet.

4.2 Modellen

Som man kan sige i R kodeudklippet nedenfor blev der benyttet `ets` funktion (Hyndman & Khandakar, 2020) i `forecast` pakke til at lave en modellen, hvor der blev givet daglige vægtværdier fra 2019-11-01 til 2020-03-01. `model="ZZZ"` betyder, at funktionen selv skal find ud, om fejlene er additive eller multiplikative, om der er tendenser og til sidst om der er sæsoner i tidserien. Fra outputtet nedenfor kan man se, at der blev udregnet M,A,N, hvilket betyder, at fejlene er multiplikative, at tendensen er additiv, og at der er ingen sæsoner i denne tidsserie (Kabacoff, 2014, 354).

```
# Create timeseries object
thive1 <- ts(hive1_winter_weights, frequency=365, start=c(2019,305))
# Create exponential forecasting model
fit <- ets(thive1, model="ZZZ")
print(fit$method)
```

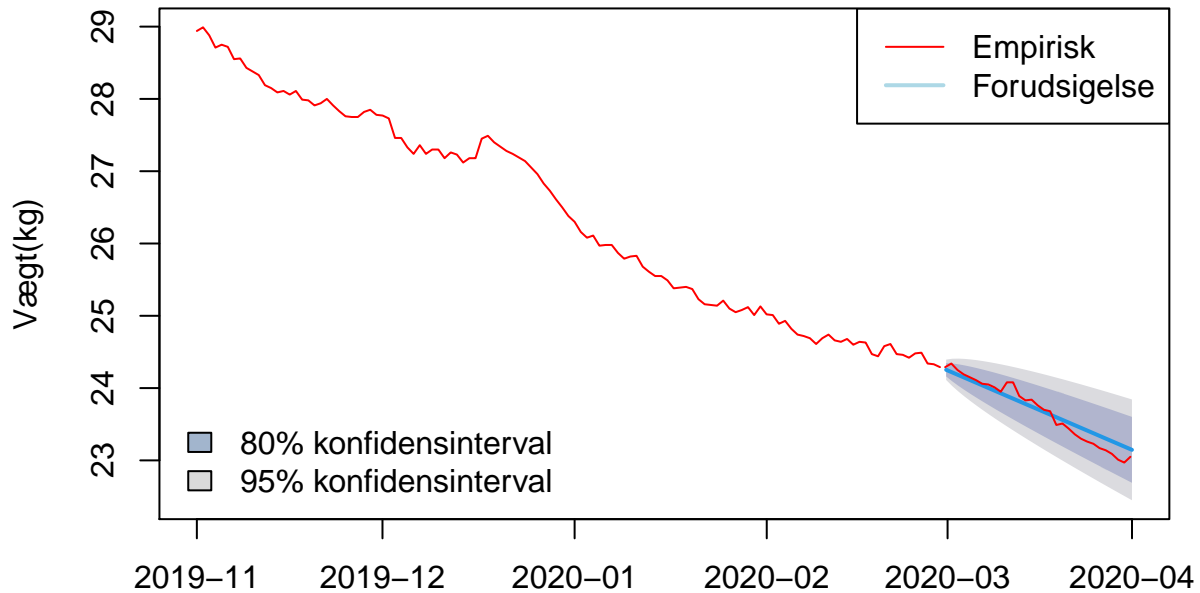
```
## [1] "ETS(M,A,N)"
```

4.3 Forudsigelse

Modellen som blev lavet i forudgående asnit blev brugt i `forecast` funktionen (Hyndman & Khandakar, 2020) til at forudsige vægten i hele marts måned, altså 31 dage. På Figur 7 nedenfor kan man se de forudsagte værdier. En blå linje betyder "Point Forecast", den mørke gråzone markerer 80% konfidensintervall, og den lyse gråzone markerer 95% konfidensinterval.

```
forecast <- forecast(fit,31)
```

Forudsigelse af vægtudvikling på Bistade1



Figur 7: Forudsigelse af vægtudvikling på Bistade1 i 2020 marts måned. De forudsagte værdier sammenlignes også med de empiriske værdier for at validere modellens nøjagtighed.

4.4 Validering

I valideringsfasen blev der først og fremmest sammenlignet de forudsagde værdier og de empiriske værdier ved at plote de empiriske værdier for 2020 marts måned på den samme graf i Figur 7. De empiriske værdier ligger pænt ind i 80% konfidensinterval, hvilket tyder på at modellen forudsiger ret præcist. Det skal bare bemærkes, at modellen kun kan forudsige vægtværdier indtil bierne faktisk begynder at indsamle nektar, da modellen som nævnt er egnet kun til vinterhalvår (sommerhalvår er dog håndteret i det næste afsnit).

For det andet blev der også undersøgt modellens **Root mean squared error (RMSE)**, som er udregnet ved $\sqrt{\text{mean}(e_t^2)}$ (Hyndman & Athanasopoulos, 2020 (afsnit 3.4)). I outputtet nedenfor kan man se at modellen i forhold til målestok har rimeligt lavt **RSME** værdi (Astur, 2013), som er 0.077.

```
forecast::accuracy(fit)
```

```
##               ME      RMSE      MAE      MPE      MAPE  MASE
## Training set -0.00198362 0.07700837 0.06158356 -0.006668392 0.2335366  NaN
##               ACF1
## Training set 0.02305504
```

Alt i alt kan man derfor konkludere at modellen er ret nøjagtigt.

5 Klassificering af vægttilvækst

5.1 Formål

I denne sektion skal der klassificeres, om vægten skulle gå OPAD eller NEDAD ved at bruge vejrparametre som “predictors”. Formålet med dette kunne være at advare biavlere, når bistadets vægten falder, selv om vejrforholdene er tilstrækkelige for at bierne skulle indsamle nektar.

Det kan forklares med, at når vægten stiger, så bierne henter nektar, hvilket kan tyde på at bierne fungerer som de skal. Men hvis vægten falder, så det kan enten betyde, at vejrforholdene er for dårligt, eller der er problemer med bierne.

Så i denne sektion skulle der defineres en model, som kunne klassificere, om vejrforholdene er gode nok for at vægten skulle stige.

5.2 Dataklargøring

Først og fremmest var der ikke nok vejrparametre i et given bistadets datasæt, så derfor blev der hentet et datasæt med flere vejr parametre fra <https://www.dmi.dk/vejrarkiv/> . Vejrdatasættet består af disse variabler: nedbør, vindhastighed, tørkeindex, luftfugtighed, lufttryk og solskin timer. Nedenfor på Figur 8 kan man også se grafer over disse variabler i juni måned 2020.

Desuden blev der også udført disse forberedelse/modifikationer på datasættet for at gøre det muligt at klassificere:

- Vægttilvæksten, som oprindeligt var kvantitativ variabel, blev lavet om til kvalitative variable. Vægttilvæksten blev opdelt i to kategorier, nemlig UP og DOWN. UP er når vægttilvæksten er positiv og DOWN er når vægttilvæksten er negativ.
- Der blev valgt at benytte data kun fra juni måned, fordi bierne trækker kun, når der er nektar, nemlig i slutning af foråret og i starten af sommeren, så derfor vil det ikke give mening at inkludere måneder, hvor der ikke sker noget træk.
- Og sidst benyttes der 2020 juni måned til at lave modellen og 2019 juni måned til at validere modellen.

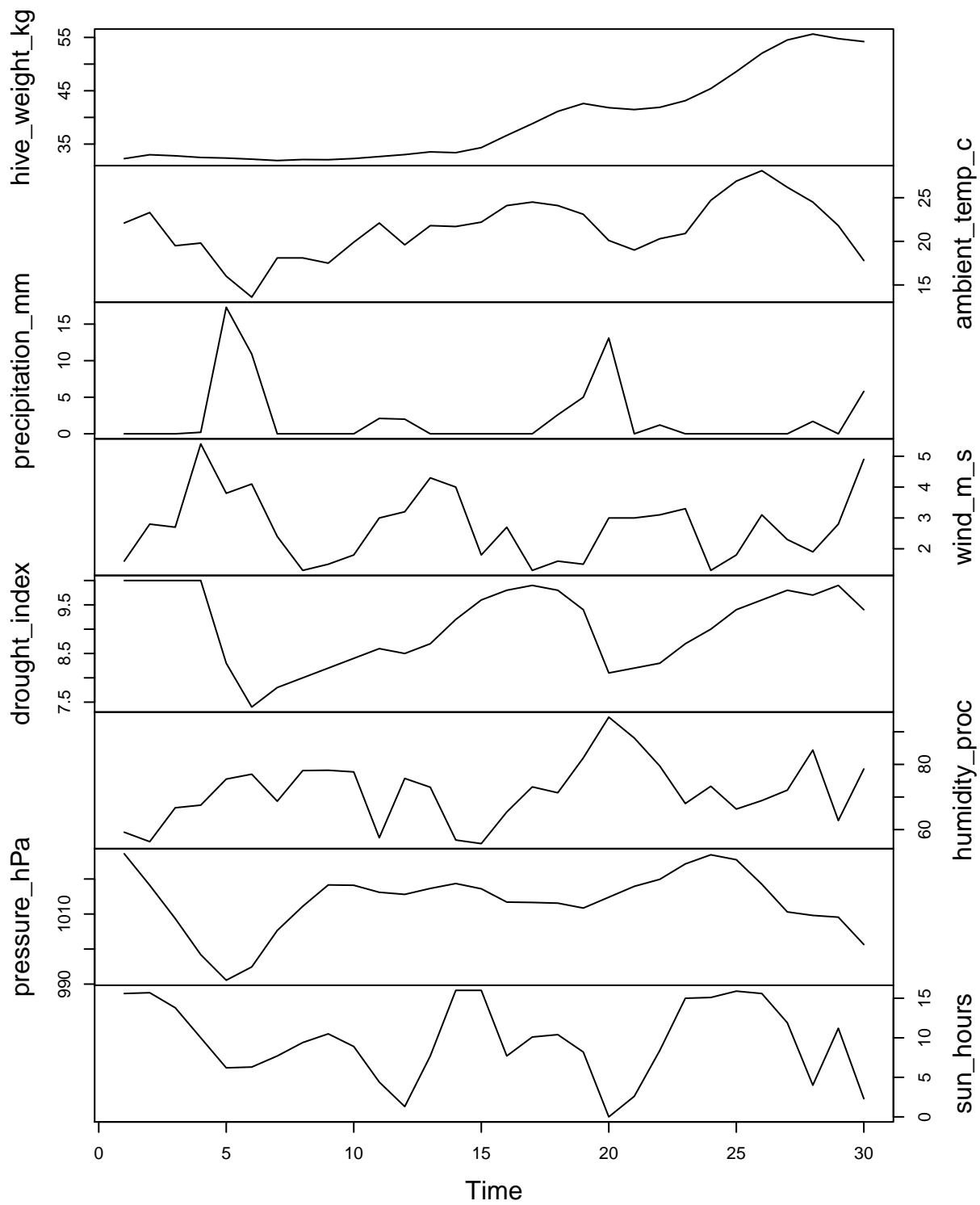
Disse forberedelser på træningssættet kan man se i R kodeudklippet i Bilag C.

5.3 Beslutningstræ

For at klassificere om vægttilvækst skulle være opadgående eller nedadgående, blev der valgt at bruge beslutningstræer, fordi træer er nemmere at fortolke og at visualisere på grafen end de andre modeller bl.a. lineær regression eller logistisk regression. På en anden side kan beslutningstræer være mindre præcise og de er heller ikke så robuste, da små ændringer i datasættet kan medføre stor ændringer i den endelige resultat. (Sørensen, 2020)

Der blev desuden valgt at bruge “Conditional inference tree” frem for traditionelle træer, fordi “Conditional inference tree” vælger variabler og skillelinje på baggrund af signifikans i stedet af homogenitet som er i traditionelle træer (Kabacoff, 2014, 397), hvilket har givet bedre resultater i valideringsfasen.

I R koden nedenfor kan man se at der blev brugt `ctree` funktion (Hothorn, Hornik & Zeileis, 2020) til at lave “Conditional inference tree”. `weight_delta_direction` er sat til at være “outcome variable” i modellen, og alle resterende variabler i datasættet, altså `ambient_temp_c_day_max`, `precipitation_mm`, `wind_m_s`, `drought_index`, `humidity_proc`, `pressure_hPa` og `sun_hours` , skulle bruges som “predictors”.



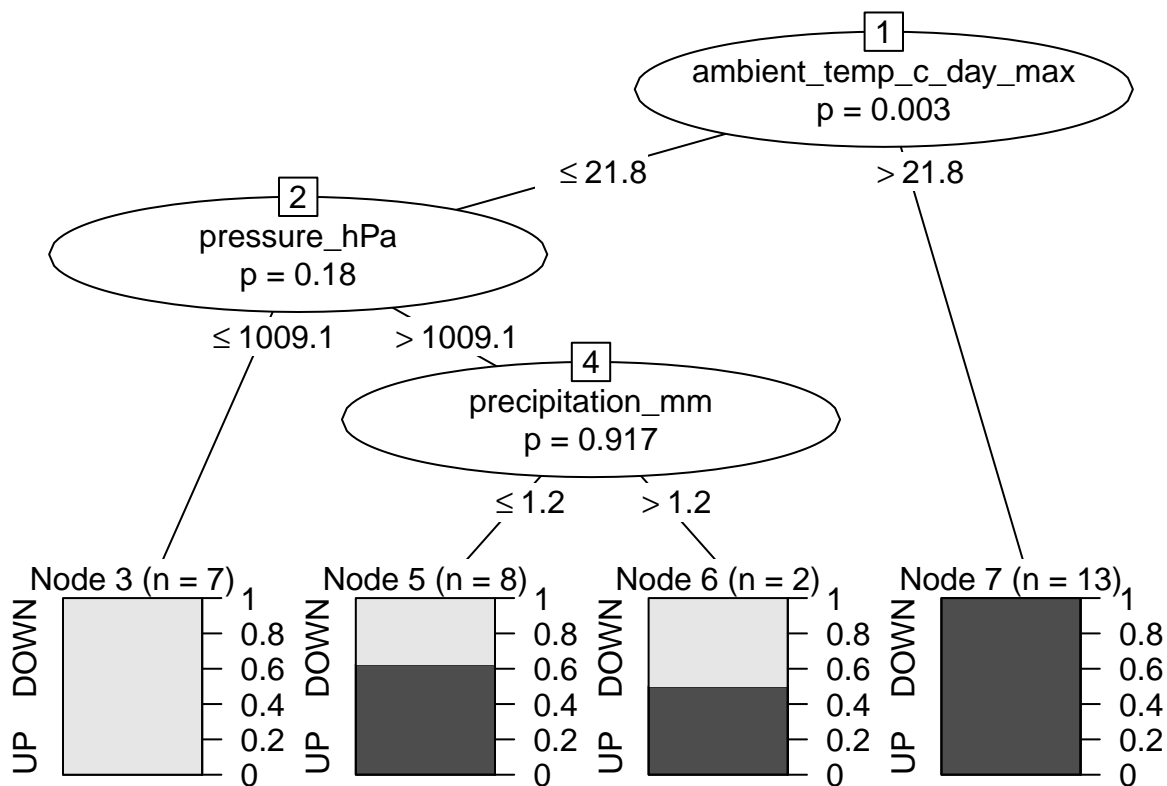
Figur 8: Data fra bistadet sammensat med eksternt vejrdato fra DMI i juni måned 2020.

```
fit.ctree <- ctree(weight_delta_direction ~ ., data=hive_data_2020_jun.train,
  controls = ctree_control(mincriterion = 0.01, minsplit= 1, minbucket = 1))
```

På Figur 9 kan man se dog, at `ctree` udvalgte kun disse tre parametre, nemlig `ambient_temp_c_day_max`, `pressure` og `precipitation`, selv om der blev givet et relativt lavt `mincriterion`, som kan udregnes ved $1 - p\text{-value}$. Man kan også se, at `precipitation` har meget lille p-value, men uden `precipitation` får man dårligere resultat i valideringen.

`minsplit` og `minbucket` er også blevet sat ned til 1 fra 20 (som var default), da træet maksimum kunne deles kun én gang med default værdierne, før grenene blev alt for små til at blive delt videre, da træningsdatasættet indeholder kun 30 observationer.

Det selve træ visualiserer, hvordan modellen klassificere om vægten skulle gå opad eller nedad. F.eks. hvis en given lufttemperatur er højere end 21.8, så modellen estimere at den skulle gå op. Til gengæld hvis temperatur er mindre end 21.8, så der kigges både på lufttryk og nedbør for at klassificere væggtilvæksten.



Figur 9: 'Conditional inference tree' for at klassificere væggtilvæksten.

5.4 Validering

I valideringsfasen blev der sammenlignet de forudsagte værdier med de observerede, hvor der blev brugt datasættet fra 2019 juni måned. Resultaterne kan man se i `confusionMatrix`(Kuhn, 2008) nedenunder. Fra outputtet nedenfor kan man se, at der blev forudsagt kun 4 gange forkert ud af de 30 dage, hvilket svare til 0.8667 “accuracy”. Man kan dog se, at “No information rate” er rimelig stor, hvilket påstår, at man kunne få den samme resultat, hvis man gættede tilfældigt(inkhorn82, 2014). Så derfor er det svært at begrunde at modellen er pålideligt nok, selv om den var ret præcis i valideringsfasen.

```
set.seed(123)
ctree.pred <- predict(fit.ctree, hive_data_2019_jun.validate, type="response" )
confusionMatrix(hive_data_2019_jun.validate$weight_delta_direction, ctree.pred)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction DOWN UP
##      DOWN      6  4
##      UP        0 20
##
##              Accuracy : 0.8667
##              95% CI : (0.6928, 0.9624)
##      No Information Rate : 0.8
##      P-Value [Acc > NIR] : 0.2552
##
##              Kappa : 0.6667
##
##  McNemar's Test P-Value : 0.1336
##
##              Sensitivity : 1.0000
##              Specificity : 0.8333
##              Pos Pred Value : 0.6000
##              Neg Pred Value : 1.0000
##              Prevalence : 0.2000
##              Detection Rate : 0.2000
##      Detection Prevalence : 0.3333
##              Balanced Accuracy : 0.9167
##
##              'Positive' Class : DOWN
##
```

6 Konklusion

I dette projekt blev der arbejdede på dataen fra bistaderne. Først blev dataen oprenset og klargjort, bagefter blev der fundet ud af at der ikke en signifikant forskel mellem bistader fra den samme have. Og til sidst blev der lavet modellerne som kunne forudsige vægtudvikling både om sommeren og om vinteren. Selv om modellerne var ret præcise i valideringsfasen, er der stadig plads til mere validering og finjustering. Personligt synes jeg også at projektet var givende, da jeg har afprøvet adskillige R værktøjer i praksis. I fremtiden kunne det være dog spændende implementere modellerne i produktionen, så at det rent faktisk kunne bidrage biavlens med at passe på hans bifamilier. Dette var dog ikke en del af dette kursus.

7 Referencer

- Astur, R. (2013). *What are good RMSE values*. Lokaliseret fra <https://stats.stackexchange.com/questions/56302/what-are-good-rmse-values>
- Brockhoff, P. B., Møller, J. K., Andersen, E. W., Bacher, P. & Christiansen, L. E. (2018). *Introduction to Statistics at DTU*.
- Hothorn, T., Hornik, K. & Zeileis, A. (2020). *Conditional Inference Trees*. Lokaliseret fra <https://cran.r-project.org/web/packages/partykit/vignettes/ctree.pdf>
- Hyndman, R. & Athanasopoulos, G. (2020). *Forecasting: Principles and Practice*. Lokaliseret fra <https://otexts.com/fpp2/index.html>
- Hyndman, R. & Khandakar, Y. (2020). *Automatic Time Series Forecasting: the forecast Package for R*. Lokaliseret fra <https://cran.r-project.org/web/packages/forecast/vignettes/JSS2008.pdf>
- inkhorn82 (2014). *Predictive modelling fun with the caret package*. Lokaliseret fra <https://www.r-bloggers.com/2014/12/predictive-modelling-fun-with-the-caret-package>
- Kabacoff, R. I. (2014). *R in action*. Manning.
- Kuhn, M. (2008). *Building Predictive Models in R Using the caret Package*. Lokaliseret fra <http://www.jstatsoft.org/article/view/v028i05/v28i05.pdf>
- Sørensen, J. A. (2020). *R based Resampling and CrossValidation exemplified by Random Forest some Tool Elements*.

8 Bilag

8.1 Bilag A: Opsummering af alle variabler i datasættet

Tabel 2: Opsummering af stade1 fra 2019-09-01 to 2020-09-01

	N	Missing	Mean	SD	Min	Q1	Median	Q3	Max
hive_id	105064	0	1.00	0.00	1.00	1.00	1.00	1.00	1.00
hive_weight_lbs	105064	0	63.81	17.60	33.20	52.14	59.46	66.93	132.37
hive_weight_kgs	105064	0	28.94	7.98	15.06	23.65	26.97	30.36	60.04
hive_temp_f	105064	0	77.57	15.24	35.78	68.18	77.00	93.56	98.78
hive_temp_c	105064	0	25.32	8.47	2.10	20.10	25.00	34.20	37.10
hive_humidity	105064	0	62.10	10.82	36.00	57.10	60.30	67.50	91.00
hive_battery_voltage	105064	0	2.13	0.00	2.13	2.13	2.13	2.13	2.13
ambient_temp_f	101787	3277	49.09	11.18	24.26	40.82	46.40	56.48	88.88
ambient_temp_c	101787	3277	9.50	6.21	-4.30	4.90	8.00	13.60	31.60
ambient_humidity	101787	3277	99.00	4.17	55.10	99.90	99.90	99.90	99.90
ambient_luminance	105064	0	14.65	56.36	0.00	0.00	0.00	5.00	2070.00
wx_temp_f	35863	69201	48.52	42.93	21.60	41.00	45.30	53.60	999.00
wx_temp_c	35863	69201	9.41	25.33	0.00	5.00	7.40	12.00	572.80
wx_relative_humidity	35863	69201	93.14	40.98	46.00	88.00	94.00	98.00	999.00
wx_wind_degrees	105064	0	38.59	82.62	0.00	0.00	0.00	37.00	999.00
wx_wind_mph	35863	69201	2.84	44.07	0.00	0.00	0.20	1.30	999.00
wx_wind_gust_mph	35863	69201	3.36	44.07	0.00	0.00	1.10	2.20	999.00
wx_pressure_mb	35863	69201	1002.44	10.93	973.50	995.80	1003.30	1009.70	1035.80
wx_dewpoint_f	35863	69201	45.96	42.89	20.80	38.80	43.30	50.70	999.00
wx_dewpoint_c	35863	69201	8.08	25.29	0.00	3.90	6.30	10.40	572.80
wx_precip_1hr_in	35863	69201	1.95	44.09	0.00	0.00	0.00	0.00	999.00
wx_precip_today_in	35863	69201	2.00	44.09	0.00	0.00	0.00	0.03	999.00
quality	105064	0	5.00	0.00	5.00	5.00	5.00	5.00	5.00

8.2 Bilag B: Mine funktioner

```
import_hive_data <- function(from, to){
  library(DBI)
  con <- dbConnect(RSQLite::SQLite(), "data/stade1.db")
  table_name <- dbListTables(con)
  fields <- dbListFields(con, table_name)
  select_period_with_intervention_query <- paste("SELECT * from", table_name, "WHERE hive_observation_t",
                                                "hive_observation_time_local < ", to, sep=" ")

  sendQuery <- dbSendQuery(con, select_period_with_intervention_query )
  # hive_data <- dbFetch(sendQuery)
  hive_data <- dbFetch(sendQuery)
  hive_data$hive_observation_time_local <- strptime(hive_data$hive_observation_time_local, format = "%Y-
  return (hive_data)
}

import_hive_data_csv <- function(filename){
  hive_data <- read.table(file=filename, sep=",")
}
```

```

hive_data <- hive_data[ c("V1", "V2")]
colnames(hive_data) <- c("hive_observation_time_local", "hive_weight_kgs")
hive_data$hive_observation_time_local <- strptime(hive_data$hive_observation_time_local, format = "%Y-%m-%d %H:%M:%S")
return (hive_data)
}

extract_rows_given_weightdelta <- function(hive_data, weightdelta){
  hive_data <- hive_data %>% mutate(weight_delta = hive_weight_kgs -
                                   dplyr::lag(hive_weight_kgs)) %>%
    mutate(weight_delta = ifelse(is.na(weight_delta), 0, weight_delta))

  hive_data[which(abs(hive_data[, "weight_delta"]) > weightdelta),
            c("hive_observation_time_local", "weight_delta")]
}

extract_rows_given_timedelta <- function(hive_data, timedelta){
  hive_data <- hive_data %>% mutate(timestamp_delta = hive_observation_time_local -
                                   dplyr::lag(hive_observation_time_local)) %>%
    mutate(timestamp_delta = ifelse(is.na(timestamp_delta), 0, timestamp_delta))
  hive_data[which(abs(hive_data[, "timestamp_delta"]) > timedelta),
            c("hive_observation_time_local", "timestamp_delta")]
}

plot_time_weight <- function(hive_data, title){
  if(missing(title)){
    title <- "Vægtudvikling"
  }
  min <- as.Date(head(hive_data, 1)[, "hive_observation_time_local"])
  max <- as.Date(tail(hive_data, 1)[, "hive_observation_time_local"])
  plot(hive_data$hive_observation_time_local, hive_data$hive_weight_kgs, type = 'l', xlab = paste("Tid fra", min, "til", max, "separat"),
       # , at=seq(as.Date(min), as.Date(max), by=(13*7))
  )
}

plot_time_weight_temp <- function(hive_data){
  min <- as.Date(head(hive_data, 1)[, 4])
  max <- as.Date(tail(hive_data, 1)[, 4])
  par(mar = c(5, 5, 3, 5))
  plot(hive_data$hive_observation_time_local, hive_data$hive_weight_kgs, type = "l", ylab = "Vægt",
       main = "Sammenhæng mellem vægt og temperatur", xlab = paste("Tid fra", min, "til", max, "separat"),
       col = "blue")
  par(new = TRUE)
  plot(hive_data$hive_observation_time_local, hive_data$ambient_temp_c, type = "l", xaxt = "n", yaxt = "n",
       ylab = "", xlab = "", col = "red") # , lty = 2
  axis(side = 4)
  mtext("Temperatur", side = 4, line = 3)
  legend("topleft", c("Vægt", "Temperatur"),
       col = c("blue", "red"), lty = c(1, 1))
}

manipulate_weight_deltas <- function(hive_data, periods){
  library(dplyr)
  # Add column weight_delta

```

```

hive_data <- hive_data %>% mutate(weight_delta = hive_weight_kgs - dplyr::lag(hive_weight_kgs)) %>%
mutate(weight_delta = ifelse(is.na(weight_delta), 0, weight_delta))

# Manipulate weight deltas
for(row in 1:nrow(periods)){
  hive_data$weight_delta[hive_data$hive_observation_time_local > periods[row, "from"] & hive_data$hiv
}

# Produce cumulative sums of weight deltas
hive_data <- hive_data %>% mutate(cum_delta=cumsum(weight_delta))

# Produce new hive_weight_kgs from calculated cumulative sums
hive_data <- hive_data %>% mutate(hive_weight_kgs = hive_weight_kgs[1] + cum_delta)

# Remove the produced columns
drops <- c("weight_delta", "cum_delta")
hive_data <- hive_data[, !(names(hive_data) %in% drops)]
return(hive_data)
}

extract_midnight_weights <- function(hive_data){

  hive_data <- hive_data %>%
    mutate( dt = as.Date(hive_observation_time_local)) %>%
    group_by(dt) %>%
    filter(hive_observation_time_local == min(hive_observation_time_local)) %>%
    ungroup() %>%
    select(!dt)

  return(data.frame(hive_data))
}

return_period <- function(hive_data, from ,to){
  from <- strptime(from, format = "%Y-%m-%d %H:%M:%S")
  to <- strptime(to, format = "%Y-%m-%d %H:%M:%S")
  return(hive_data[hive_data$hive_observation_time_local > from & hive_data$hive_observation_time_local
}

```

8.3 Bilag C: Dataklargøring til klassificering af vægttilvækst

```
# Prepare train data
hive_data_2020_jun.train <- import_hive_data_daily(from = "'2020-05-31 00:00:00'",
                                                  to="'2020-07-01 00:00:00'")

# Extract weight deltas
hive_data_2020_jun.train <- hive_data_2020_jun.train %>% mutate(weight_delta =
  hive_weight_kgs_daily - dplyr::lag(hive_weight_kgs_daily)) %>%
  mutate(weight_delta = ifelse(is.na(weight_delta), 0, weight_delta)) %>%
  slice(2:n())

# Categorize weight delta directions
hive_data_2020_jun.train <- hive_data_2020_jun.train %>%
  mutate(weight_delta_direction = ifelse(weight_delta<0, "DOWN", "UP"))
hive_data_2020_jun.train$weight_delta_direction <-
  factor(hive_data_2020_jun.train$weight_delta_direction)

# Merge with weather data
weather_data_furesoe_2020_jun <- read.table(file="data/furesø-kommune-juni-2020.csv",
                                             sep="," ,header = TRUE)
weather_data_furesoe_2020_jun$dt <- as.Date(weather_data_furesoe_2020_jun$dt)
hive_data_2020_jun.train <- merge(hive_data_2020_jun.train,
                                  weather_data_furesoe_2020_jun, by="dt")

# Remove not used columns
hive_data_2020_jun.train <- select(hive_data_2020_jun.train,
                                   !c(hive_weight_kgs_daily,weight_delta, dt))

# Prepare validate data
hive_data_2019_jun.validate <- import_hive_data_daily(from = "'2019-05-31 00:00:00'", to="'2019-07-01 00:00:00'")

# Extract weight deltas
hive_data_2019_jun.validate <- hive_data_2019_jun.validate %>% mutate(weight_delta = hive_weight_kgs_daily -
  mutate(weight_delta = ifelse(is.na(weight_delta), 0, weight_delta)) %>%
  slice(2:n())

# # Categorize weight delta directions
hive_data_2019_jun.validate <- hive_data_2019_jun.validate %>%
  mutate(weight_delta_direction = ifelse(weight_delta<0, "DOWN", "UP"))
hive_data_2019_jun.validate$weight_delta_direction <-factor(hive_data_2019_jun.validate$weight_delta_direction)

# Merge with weather data
weather_data_furesoe_2019_jun <- read.table(file="data/furesø-kommune-juni-2019.csv", sep="," ,header = TRUE)
weather_data_furesoe_2019_jun$dt <- as.Date(weather_data_furesoe_2019_jun$dt)
hive_data_2019_jun.validate <- merge(hive_data_2019_jun.validate, weather_data_furesoe_2019_jun, by="dt")

# Remove not used columns
hive_data_2019_jun.validate <- select(hive_data_2019_jun.validate, !c(hive_weight_kgs_daily,weight_delta, dt))
```