Prosjekt

Titanic katastrofen- ECON 3170

Introduksjon

I denne oppgaven skal vi - ved hjelp av ulike maksinlæringsmodeller - forsøke å predikere om passasjerer på Titanic overlevde eller ikke. Datasettet er hentet fra kaggle, og oppgaven er inspirert av Kaggles maksinlæringskonkurranse (Kaggle). Vi innleder med å innspisere datasettet, og forsøke å få oversikt over de ulike variablene, og omfanget av manglende verdier. Vi vil så forsøke å behandle de manglende verdiene, og tilpasse de slik at de blir enklere å bruke i treningen av maskinlæringsmodellene. Videre vil vi inspisere datasettet ved hjelp av plots for å forsøke å se sammenhengen mellom ulike variabler, og andelen overlevende. Vi vil så tilpasse parametere og hyperparametere for å forsøke å få så presise modeller som mulig. Til slutt vil vi implementere modellene i et dashboard hvor en kan finne overlevelsessansynlighet for en selvvalgt karakter.

Pakker

I prosjetet kommer vi hovedsaklig til å benytte oss av to pakker: tidyverse og tidymodels. Begge pakkene er samlinger av mange ulike pakker. Vi kommer i hovedsak til å bruke Tidyverse til manipulering og visualisering av data, og Tidymodels til maskinlæring.

```
library(tidyverse)
library(tidymodels)
```

Data

```
titanic <- suppressMessages(read_csv("Titanic-Dataset.csv"))
glimpse(titanic[1,])
```

```
Rows: 1
Columns: 12
$ PassengerId <dbl> 1
$ Survived    <dbl> 0
$ Pclass      <dbl> 3
$ Name        <chr> "Braund, Mr. Owen Harris"
$ Sex         <chr> "male"
$ Age         <dbl> 22
$ SibSp       <dbl> 1
$ Parch       <dbl> 0
$ Ticket      <chr> "A/5 21171"
$ Fare        <dbl> 7.25
$ Cabin       <chr> NA
$ Embarked    <chr> "S"
```

#NOTE: "suppressMessages", for å unngå output melding.

Datasettet består av 891 observasjoner med 12 variabler, og er en oversikt over et utvalg av passasjerene ombord på Titanic. Første kolonne er PassengerId som gir hver observasjon en unik ID. Den andre kolonnen er Survival; en binær variabel som indikerer om passasjeren overlevde (1) eller ikke (0). Tredje kolonne er Pclass som forteller hvilken klasse passasjeren var på. Fjerde kolonne, Name, viser navnene til hver enkelt passasjer. Kolonne fem, Sex, er en kategorisk variabel som betegner kjønnet til passasjeren. Den sjette kolonnen, Age, er en kontinuerlig variabel som gir alderen til passasjeren. Kolonne sju, SibSp, er en variabel som viser antallet søsken eller ektefeller ombord på Titanic. Kolonne åtte, Parch viser antallet foreldre eller barn ombord.Nienede kolonne, Ticket, viser bilettnummeret til passasjeren Tiende kolonne, Fare, betegner prisen passasjeren brukte på biletten. Kolonne elleve, Cabin, er lugarnummeret til passasjeren. Embarked, den tolvte kolonnen er en kategorisk variabel som viser hvor passasjeren gikk på. hvor "C" er Cherbourg, "Q" er Queenstown og "S" er Southampton.

Kilde: (Kaggle)

```
head(titanic)
```

```
# A tibble: 6 x 12
  PassengerId Survived Pclass Name    Sex     Age SibSp Parch Ticket  Fare Cabin
        <dbl>    <dbl>  <dbl> <chr>   <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
1           1        0      3 Braund~ male     22     1     0 A/5 2~  7.25 <NA>
2           2        1      1 Cuming~ fema~    38     1     0 PC 17~ 71.3  C85
3           3        1      3 Heikki~ fema~    26     0     0 STON/~  7.92 <NA>
4           4        1      1 Futrel~ fema~    35     1     0 113803 53.1  C123
5           5        0      3 Allen,~ male     35     0     0 373450  8.05 <NA>
```

```
6           6        0      3 Moran,~ male     NA     0     0 330877  8.46 <NA>
# i 1 more variable: Embarked <chr>
```

Pclass og Fare virker veldig spennede, de fleste har nok sett filmen og der blir tydligfremstilt at hvis du er i første klasse har du en større sjanse for å overleve. Samme med Fare, hvis du har betalt mye for biletten indikerer dette en høyere klasse som kan hinte til en større sannsynlighet for å overleve.

Cabin og Ticket er nok også verdier som har en sterk tilknytting med hvilken klasse du er i og hvor mye du betalte. Likevel blir det vannskelig å skulle si om noen overlever p.ga hvilket bilettnummer man har. Ved første øyekast ser det ut som at lugar variablen har mange manglende verdier. Hva den faktiske lugaren var er nok kanskje ikke altfor interessant, men hvilket dekk den ligger på kunne vært av stor interesse. Ettersom at dekket har mye å si for om du klarte å komme deg ut i tide.

Alder og kjønn er igjen ganske interessante variabler. Vi kjenner jo alle til "Kvinner og barn først". Her vil det nok være interessant å manipulere dataen for å fremmheve om vedkommende er et barn eller ikke. En annen dimensjon kan også være om personen blir ansett som gammel eller ikke, ettersom at disse gruppen antaglivis får prioritet under evakuering av skipet. Samt også kombinere disse verdiene med hvilken klasse de var i.

Navn er litt "tricky". Man kan gjerne tenke seg at naven til noen ikke er av stor betydning når det kommer til overlevelse som forsåvidt kan stemme. Likevel inkluderer navene tittler so f.eks mrs og master, og dette kan igjen være interessant.

Embarked er igjen variabel det blir litt vannskelig å si noe om, ettersom at det er kun tre forskjellige steder, som egentlig ikke burde ha særlig stor betydning. På den andre siden kan det være en korrelasjon mellom hvor man gikk om bord og hvilken klassen man er i.

Parch og Sibsp er en ganske interessante variabler. Kan det være noe sammenheng mellom hvor stor familie du har ombord og overlevelse.

PassengerId vikrer ikke veldig relevant, den virker litt mer som en variabel for å kunne referere til passasjer og den har nok mest sannsynlig ikke noe å gjøre med overlevelsen.

(Trenger vi denne gjennomgangen?)

Inspirasjon (Allohvk)

Utforsking av dataen

Behandling av manglende verdier(NA's)

```
skimr::skim(titanic)
```

Table 1: Data summary

Name	titanic
Number of rows	891
Number of columns	12
Column type frequency:	
character	5
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Name	0	1.00	12	82	0	891	0
Sex	0	1.00	4	6	0	2	0
Ticket	0	1.00	3	18	0	681	0
Cabin	687	0.23	1	15	0	147	0
Embarked	2	1.00	1	1	0	3	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
PassengerId	0	1.0	446.00	257.35	1.00	223.50	446.00	668.5	891.00	
Survived	0	1.0	0.38	0.49	0.00	0.00	0.00	1.0	1.00	
Pclass	0	1.0	2.31	0.84	1.00	2.00	3.00	3.0	3.00	
Age	177	0.8	29.70	14.53	0.42	20.12	28.00	38.0	80.00	
SibSp	0	1.0	0.52	1.10	0.00	0.00	0.00	1.0	8.00	
Parch	0	1.0	0.38	0.81	0.00	0.00	0.00	0.0	6.00	
Fare	0	1.0	32.20	49.69	0.00	7.91	14.45	31.0	512.33	

Vi bruker funksjonen skim fra skimr pakken for å skaffe en oversikt over dataen. Det mest interresante i utskriften er kolonnen "n_missing" som viser hvor mange manglende verdier det er per variabel. Dersom vi ser på variabelen "Cabin" ser vi at den har 687 manglende verdier. Det er rimelig å anta at selve lugarnummeret ikke er så viktig for å predikere overlevelse, og at variabler som Pclass da vil være til større hjelp. Dersom man hadde gjort noen antagelser om hvordan dekkene var organisert kunne en brukt for eksempel Pclass og Fare til å forsøke å estimere lugarnummeret, men ettersom dette vil bli noe upresist og kunne resultert i uheldige

utslag på prediksjonene - som følge av et lite datasett - velger vi istedenfor å bare fjerne hele variabelen.

```
titanic <- titanic |>
  select(-Cabin)
```

Variabelen Age mangler 177 verdier, noe som kan være problematisk ettersom vi tenker at alder kan være viktig en viktig indikator for om noen overlevde eller ikke. Siden datasettet ikke er så stort velger vi å forsøke å fylle de manglende verdiene med gjennomsnittsalderen til passasjerer som deler tittel, og reiser med samme klasse (Pclass). Ved å dele inn etter både tittel og klasse forsøker vi å unngå at for mange observasjoner får den samme alderen, noe som kan bidra til å påvirke presisjonen til modellene.

Inspirasjonen er hentet fra artikkelen til *Allohvk*

```
#Lager lister til å fylle inn verdiene
master.mean <- list()
mr.mean <- list()
mrs.mean <- list()
miss.mean <- list()


#Setter opp en for-loop til å finne gjennomsnittet for alle med en gitt tittel, i en gitt kla
for (i in 1:3){
  for (j in c("Master", "Mr.", "Mrs.", "Miss")) {
    if (j == "Master"){
      master.mean[[i]] <- titanic |>
        filter(grepl("Master.", Name, fixed = TRUE), Pclass == i, Sex == "male") |>
        summarise(mean_age = mean(Age, na.rm = TRUE)) |>
        pull(mean_age)
    } else if(j == "Mr."){
      mr.mean[[i]] <- titanic |>
        filter(grepl("Mr.", Name, fixed = TRUE), Pclass == i, Sex == "male") |>
        summarise(mean_age = mean(Age, na.rm = TRUE)) |>
        pull(mean_age)
    } else if (j == "Mrs.") {
      mrs.mean[[i]] <- titanic |>
        filter(grepl("Mrs.", Name, fixed = TRUE), Pclass == i, Sex == "female") |>
        summarise(mean_age = mean(Age, na.rm = TRUE)) |>
        pull(mean_age)
    } else if(j == "Miss") {
      miss.mean[[i]] <- titanic |>
        filter(grepl("Miss.", Name, fixed = TRUE), Pclass == i, Sex == "female") |>
```

```
        summarise(mean_age = mean(Age, na.rm = TRUE)) |>
        pull(mean_age)
    }
  }
}


#Fyller inn de manglende verdiene
for (i in 1:3){
  titanic <- titanic |>
    mutate(Age = ifelse(grepl("Mr.", Name, fixed = T) & Pclass == i & is.na(Age), mr.mean[[i]
    mutate(Age = ifelse(grepl("Miss", Name, fixed = T) & Pclass == i & is.na(Age), miss.mean
    mutate(Age = ifelse(grepl("Mrs.", Name, fixed = T) & Pclass == i & is.na(Age), mrs.mean[
    mutate(Age = ifelse(grepl("Master", Name, fixed = T) & Pclass == i & is.na(Age), master.m
}
```

Vi fyller inn de manglende verdiene ved å ta utgangspunkt i titlene - Master, Mr. , Mrs. , Miss. - og finner gjennomsnittsalderen for klassen de tilhører, og fyller dette inn i datasettet. Ettersom det eksisterer flere titler som vi ikke dekket sjekker vi om vi finner flere manglende verdier, og forsøker samme metode for å sette inn verdiene.

```
#Sjekker om det er fler manglende verider
titanic |>
  filter(is.na(Age))
```

```
# A tibble: 1 x 11
  PassengerId Survived Pclass Name          Sex     Age SibSp Parch Ticket  Fare
        <dbl>    <dbl>  <dbl> <chr>         <chr> <dbl> <dbl> <dbl> <chr>  <dbl>
1         767        0      1 Brewe, Dr. A~ male     NA     0     0 112379  39.6
# i 1 more variable: Embarked <chr>
```

```
#Finner gjennomsnittet for tittel gitt klasse
mean.dr.p1 <- titanic |>
  filter(grepl("Dr.", Name, fixed = TRUE) & Pclass == 1) |>
  summarise(round(mean(Age, na.rm = TRUE))) |>
  pull()

#Setter inn for verdien
titanic <- titanic |>
  mutate(Age = ifelse(is.na(Age), mean.dr.p1, Age))
```

Fra utskriften til skimr funksjonen ser vi at det nå kun mangler 2 verdier for variabelen Embarked. Vi undersøker disse nærmere:

```
titanic |>
  filter(is.na(Embarked))
```

```
# A tibble: 2 x 11
  PassengerId Survived Pclass Name          Sex     Age SibSp Parch Ticket  Fare
        <dbl>    <dbl>  <dbl> <chr>         <chr> <dbl> <dbl> <dbl> <chr>  <dbl>
1          62        1      1 Icard, Miss.~ fema~    38     0     0 113572    80
2         830        1      1 Stone, Mrs. ~ fema~    62     0     0 113572    80
# i 1 more variable: Embarked <chr>
```

Vi ser at de manglende verdiene er for to kvinner, som begge overlevde, og som reiste med første klasse. For å opprettholde variasjonen i datasettet, og bevare flest mulig av observasjonene vi ønsker å predikere, vil vi forsøke å beholde observasjonene ved å fylle inn en verdi for Embarked.

```
titanic |>
  filter(Sex == "female" & Pclass == 1) |>
  group_by(Embarked, Pclass) |>
  summarise(mean_survived = mean(Survived), antall = n())
```

```
# A tibble: 4 x 4
# Groups:   Embarked [4]
  Embarked Pclass mean_survived antall
  <chr>     <dbl>         <dbl>  <int>
1 C             1         0.977     43
2 Q             1         1          1
3 S             1         0.958     48
4 <NA>          1         1          2
```

Det vi kan tolke fra utskriften er at de fleste kvinnene som reiste med førsteklasse overlevde, uavhengig av hvor de gikk ombord. Det vil derfor være rimelig å fylle de manglende verdiene med et tilfeldig trekk mellom "C" og "S", ettersom disse var de vanligste stedene å gå ombord.

```
tilfeldig_embarked <- sample(c("C", "S"), size = 1)

titanic <- titanic |>
  mutate(Embarked = ifelse(is.na(Embarked),
                           yes = tilfeldig_embarked,
```

```
                              no = Embarked))

paste("Det er",sum(is.na(titanic)), "NAs datasettet")
```

```
[1] "Det er 0 NAs datasettet"
```

Visualisering av data

Når vi nå har bearbeidet dataene ønsker vi å utforske de ytterligere, for å få et bedre inntrykk av sammenhengene i dataene.

```
titanic |>
  group_by(Survived) |>
  summarise(Antall = n())
```

```
# A tibble: 2 x 2
  Survived Antall
     <dbl>  <int>
1        0    549
2        1    342
```

Vi ser her at det kun er 342 passasjerer som overlevde, og at det var 549 passasjerer som omkom. Vi ønsker først å se på hvordan antall overlevende fordeler seg på klassen de reiste med. Antallet overlevende ser ut til å være ganske likt mellom klassene. Å se på andelen overlevende fordelt på klasse kan derfor gi et bedre inntrykk.

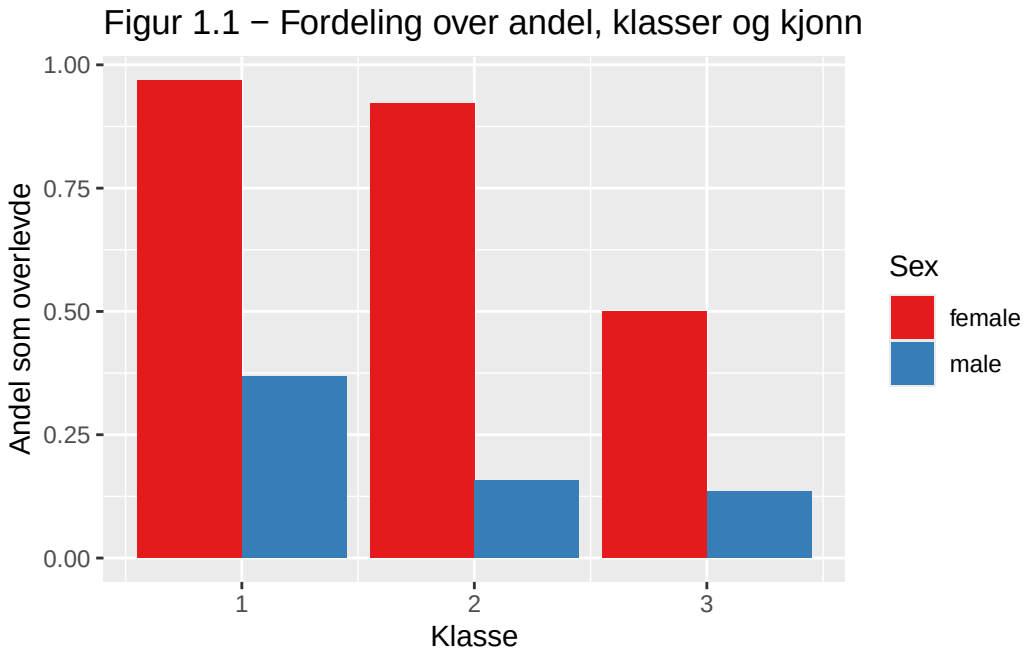
```
titanic |>
  group_by(Pclass) |>
  summarise(Overlevelsesandel = sum(Survived)/n())
```

```
# A tibble: 3 x 2
  Pclass Overlevelsesandel
   <dbl>             <dbl>
1      1             0.630
2      2             0.473
3      3             0.242
```

Kjønn og overlevlese

Vi ser av utskriften at andelen som overlevde er betydelig høyere for klasse 2 og 1 enn for klasse 2. Videre ønsker vi å undersøke betydningen til kjønn. Vi finner Overlevelsesandelen, fordelt på kjønn, fordelt på klasse.

```
library(tidyverse)

titanic |>
  group_by(Pclass,Sex) |>
  summarise(Overlevelsesandel = sum(Survived)/n()) |>
  ggplot(aes(x = Pclass, y = Overlevelsesandel, fill = Sex)) +
  geom_col(position = position_dodge()) +
  ylab("Andel som overlevde") +
  xlab("Klasse") +
  ggtitle("Figur 1.1 - Fordeling over andel, klasser og kjonn") +
  scale_fill_brewer(palette = "Set1")
```

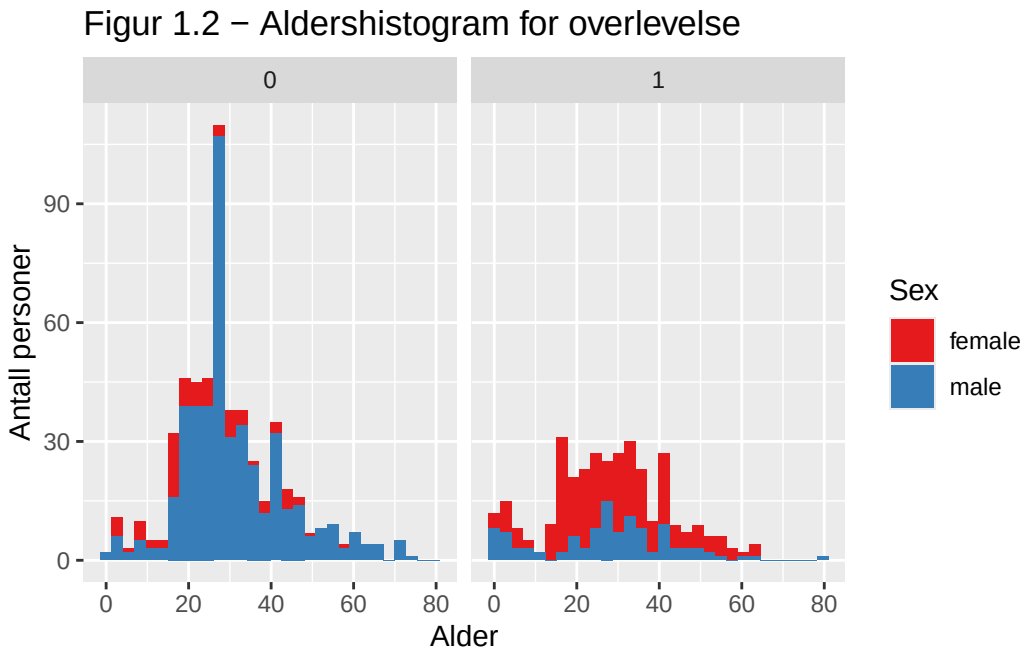


Av diagrammet kan en se at det er en markant forskjell på overlevelsesandelen for kvinner og menn, for alle klasser. En kan også se at observasjonen fra utskriften også holder her, men at effekten er større for menn enn for kvinner. Vi ser at andelen for kvinner som overlevde i tredje klasse er større enn andelen for menn som overlevde i første klasse. Dette gir mening med tanke på at kvinner og barn var prioritert - "Kvinner og barn først". Vi ønsker derfor å se på hvordan alder, og kjønn fordeler seg på overlevelse.

Vi plotter derfor et histogram som viser aldersfordelingen, og kjønnsfordelingen for døde og overlevende.

```
titanic |>
  ggplot(aes(x = Age, fill = Sex)) +
  geom_histogram() +
  facet_wrap(vars(Survived)) +
  scale_fill_brewer(palette = "Set1") +
  ggtitle("Figur 1.2 - Aldershistogram for overlevelse") +
  xlab("Alder") +
  ylab("Antall personer")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



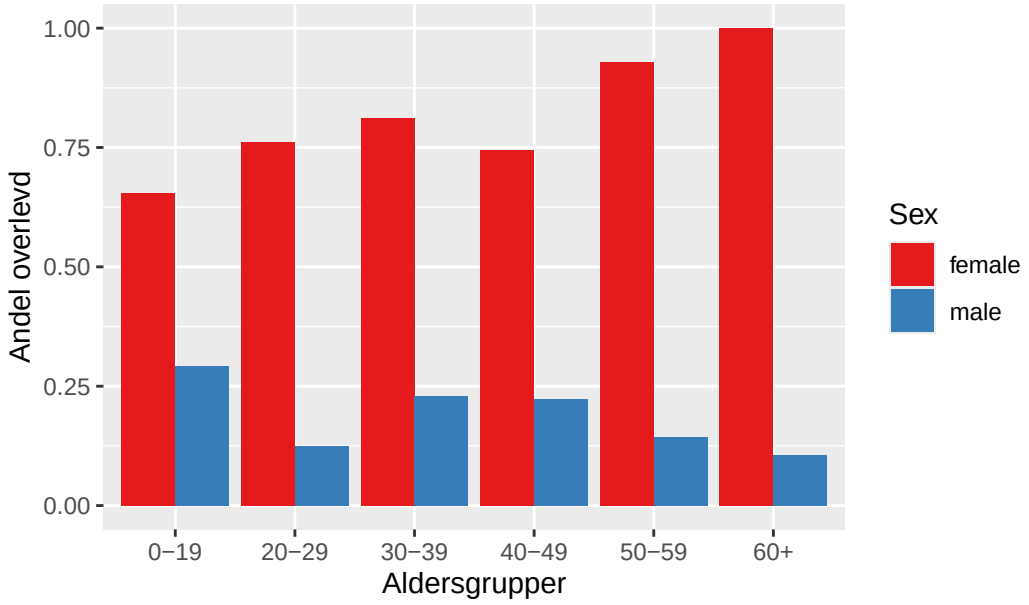
Histogrammet bekrefter de tidligere observasjonene. Vi ser at histogrammet for overlevende har en tyngre hale for Alder < 18, noe som indikerer at det var en større overlevelse blant barn og unge. Dette styrker hypotesen om at kjønn og alder er tett korrelert med sannsynligheten for å overleve.

For å vise denne effekten tydeligere deler vi inn i aldersgrupper

```
### Er dette et bedre plot? Viser det samme, men lettere å sammenlinge
titanic |>
```

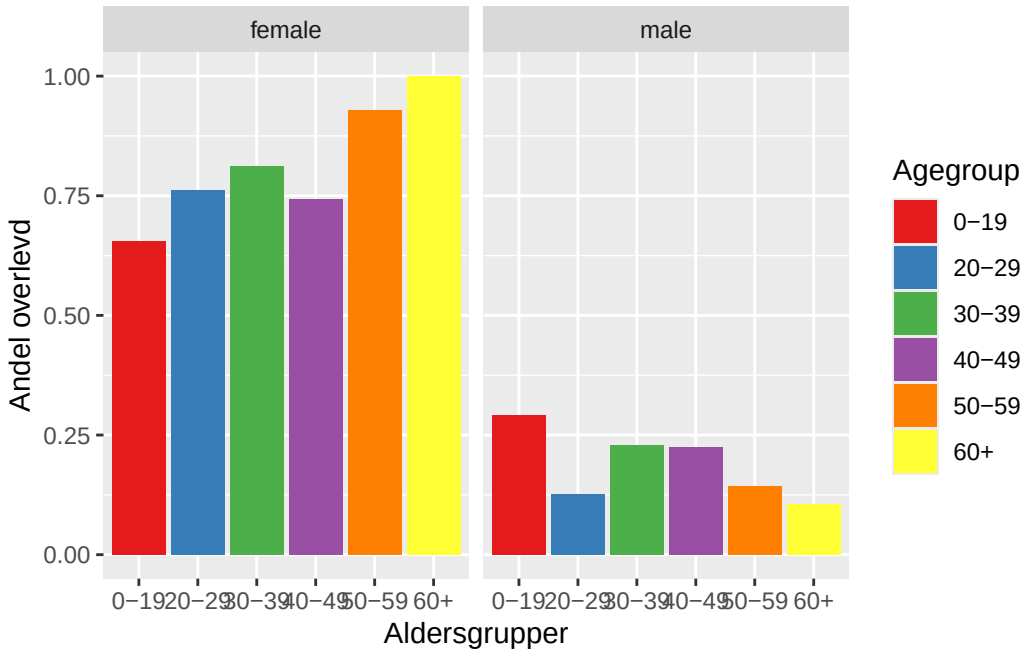
```
  mutate(
    Agegroup = cut(
     Age,
      breaks = c(-Inf,20, 30, 40, 50, 60, Inf),
     labels = c("0-19", "20-29", "30-39", "40-49", "50-59", "60+"))) |>
  group_by(Agegroup,Sex) |>
  summarise(Andel = sum(Survived)/n())|>
  ggplot(aes(x = Agegroup, y = Andel, fill = Sex)) +
  geom_col(position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  xlab("Aldersgrupper") +
  ylab("Andel overlevd") +
  ggtitle("Figur 1.3 Andelen overlevne i ulike aldersgrupper, fordelt på kjønn")
```

Figur 1.3 Andelen overlevne i ulike aldersgrupper, fordelt på kj




```
titanic |>
  mutate(
    Agegroup = cut(
     Age,
      breaks = c(-Inf, 20, 30, 40, 50, 60, Inf),
     labels = c("0-19", "20-29", "30-39", "40-49", "50-59", "60+"))) |>
  group_by(Agegroup,Sex) |>
  summarise(Andel = sum(Survived)/n())|>
  ggplot(aes(x = Agegroup, y = Andel, fill = Agegroup)) +
```

```
  geom_col(position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  xlab("Aldersgrupper") +
  ylab("Andel overlevd") +
  facet_wrap(vars(Sex))
```

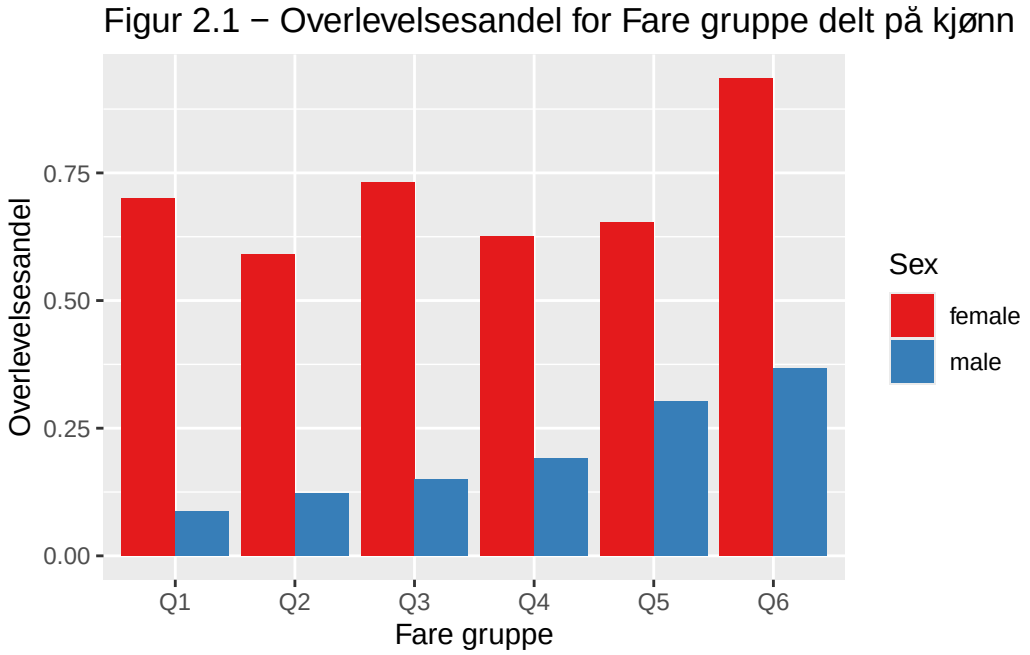


Vi ser her at tendensen fra tidligere - at kvinner i større grad overlever en menn - består. Vi ser at alder har en svak sammenheng med overlevelse, men at denne sammenhengen er ulik for kvinner og menn. For kvinner er det en stigende sammenheng, med unntak av aldersgrupper 41-50, som går litt ned. Grunnen for dette kan være at kvinner mellom 0 og 20, enten selv var barn - og dermed ble prioritert - eller at de var unge barnløse kvinner som hadde prioritet etter barn, og kvinner med barn slik at de trekker overlevelsesandelen ned. Kvinner i gruppen 41-50 kan , pågrunn av alder, antas å reise uten barn, og ble derfor nedprioritert til fordel for barn, kvinner med barn og eldre kvinner. For menn ser man at den høyeste overlevelsesraten er for menn mellom 0-20, noe som er plausibelt ettersom barn ble prioritet. Den laveste overlevelsesraten er for menn i alderen 20-31, videre kan en se at det ikke er særlig stor sammenheng mellom alder og overlevelse, men at det er en synkende tendens for aldre over 30.

Fare, Embarked og Overlevelse

Videre ønsker vi å undersøke sammenhengen mellom prisen på billett, og overlevelse. Deler derfor variabelen fare inn i 6 grupper, og finner overlevelsesandelen per gruppe.

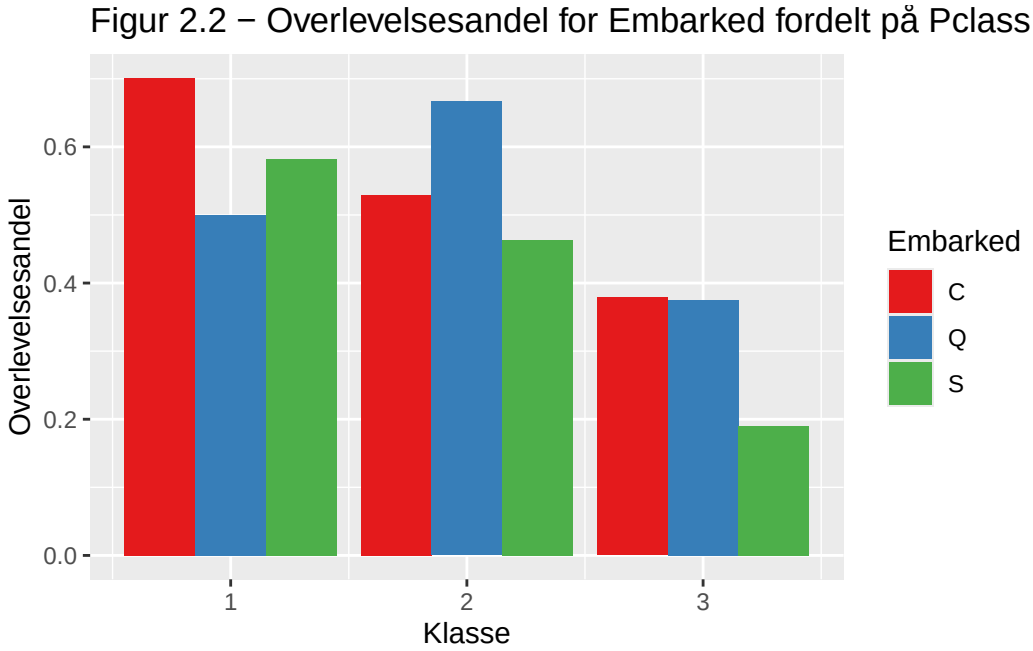
```r
titanic |>
  filter(Fare < 200) |>
  mutate(Fare_group = ntile(Fare, 6)) |>
  mutate(Fare_group = factor(Fare_group, labels = c("Q1", "Q2", "Q3", "Q4", "Q5", "Q6"))) |>
  group_by(Fare_group, Sex) |>
  summarise(andel = sum(Survived)/n()) |>
  ggplot(aes(x = Fare_group, y = andel, fill = Sex)) +
  geom_col(position = "dodge") +
  scale_fill_brewer(palette = "Set1") +
  xlab("Fare gruppe") +
  ylab("Overlevelsesandel") +
  ggtitle("Figur 2.1 - Overlevelsesandel for Fare gruppe delt på kjønn")
```



Utvalget har noen uteliggende observasjoner som vi har fjernet for å få en tydeligere visualisering, vi deler så utvalget inn i seks ulike grupper, og viser overlevelsesratene for personene i de ulike Fare gruppene, og deler inn i kjønn. Vi ser av denne visualiseringen at det ikke er noe tydelig sammenheng for kvinner hva gjelder hvilken Fare gruppe de var i. For menn ser vi at det er en jevn økning i overlevelsesandelen når faregruppen øker.

Til sist ønsker vi å se på hvordan overlevelsesandelen for passasjerene på de ulike ombordstigningene er.

```r
titanic |>
  group_by(Embarked, Pclass) |>
  summarise(andel_overlevde = sum(Survived)/ n()) |>
  ggplot(aes(x = Pclass, y = andel_overlevde, fill = Embarked )) +
  geom_col(position = position_dodge()) +
  scale_fill_brewer(palette = "Set1") +
  ggtitle("Figur 2.2 - Overlevelsesandel for Embarked fordelt på Pclass") +
  xlab("Klasse") +
  ylab("Overlevelsesandel")
```



Vi ser her at det ikke er noe tydelig mønster for overlevelsesandel, og hvor passasjerene gikk ombord. Dette er også rimelig med tanke på at hvor en gikk om bord trolig ikke påvirker ens sjanse til å overleve.

Konklusjon fra datavisualisering

Vi kan se av grafene at det i hovedsak er kjønn og hvilken klasse en tilhører som har sterkest sammenheng med overlevelsen. Vi ser at hvor mye biletten kostet har noe å si for overlevelse, men denne sammengen er sterkest for menn, og at hvor en gikk på ikke har så mye å si. Alder har mest å si for menn.

Nye variabler

Vi ser at den kan være hensiktsmessig å formalisere noen av variablene, og lage nye variabler. Dette kan forbedrer modellens ytelse, gjør resultatene lettere å forstå og kan gi ny innsikt.

Vi velger å lage en dummy variabel for om man er mindreåring eller ikke, en for hvor stor familie personen reiser med, og antall personer per billett. Inspirasjon hentet (Donges, 2018).

```
titanic <- titanic |>
  add_count(Ticket, name = "Person_per_ticket") |>
  mutate(Minor = ifelse(Age < 18, 1, 0),
         Family_size = SibSp+Parch,
         Alone = ifelse(Family_size == 0, 1, 0)) |>
  select(-c(Name, Ticket, SibSp, Parch, PassengerId)) |>
  mutate(Pclass = as.factor(Pclass),
         Alone = as.factor(Alone),
         Survived = as.factor(Survived),
         Minor = as.factor(Minor),
         Sex = as.factor(Sex),
         Embarked = as.factor(Embarked))
```

Maskinlæring

Før vi tilpasser modellene vi ønsker å bruke, deler vi datasettet inn i trening og test sett. Dataene deles inn i en 80/20 fordeling hvor variabelen Survived skal være jevnt fordelt.

```
set.seed(3170)
titanic_split<- initial_split(titanic, prop = .8, strata = Survived)
titanic_train <- training(titanic_split)
titanic_test <- testing(titanic_split)
titanic_split
```

```
<Training/Testing/Total>
<712/179/891>
```

Lager en recepie:

```
titanic_recipe <-
  recipe(Survived ~ ., data = titanic_train) |>
  step_dummy(Sex, Embarked, Alone, Pclass, Minor)
```

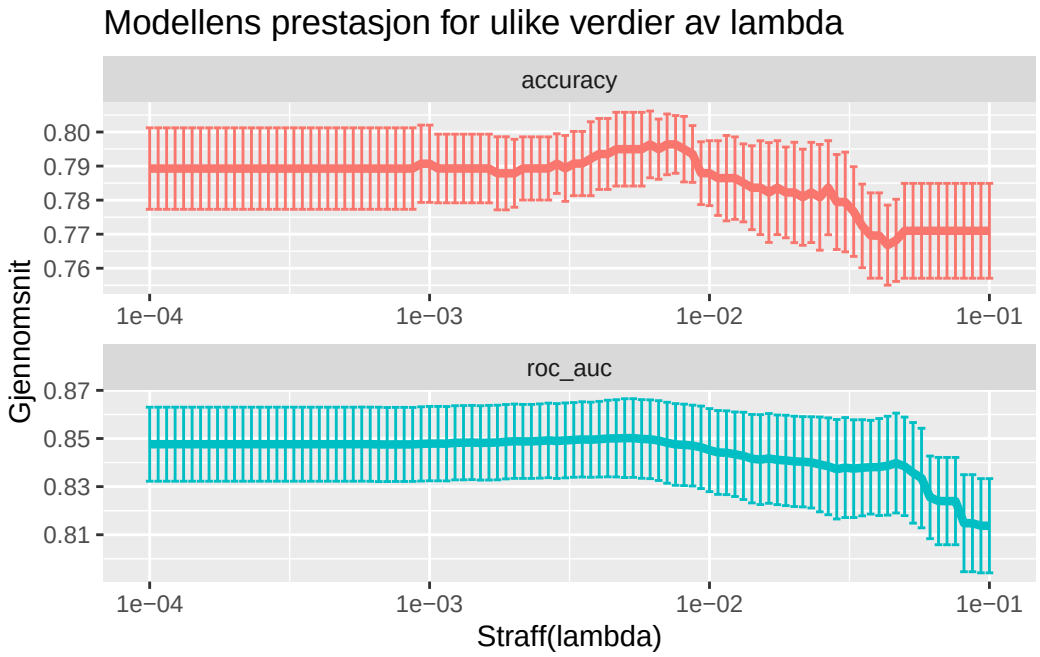
Lasso

Vi setter opp lasso modellen og tuner den. Plottet viser hvordan accuracy og roc_auc utvikler seg for ulike verdier av lambda.

```
lasso_model <-
  logistic_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet") |>
  set_mode("classification")


wflow_lasso <-
  workflow() |>
  add_recipe(titanic_recipe) |>
  add_model(lasso_model)

folds <- vfold_cv(titanic_train, 5, strata = Survived)

penalty_grid <- grid_regular(penalty(range = c(-4, -1)), levels = 100)

doParallel::registerDoParallel() # Parallelprogramering for raskere tune

lasso_tune <-
  wflow_lasso |>
  tune_grid(resamples = folds, grid = penalty_grid)

lasso_tune |>
  collect_metrics() |>
  filter(.metric != "brier_class") |>
  ggplot(aes(penalty, mean, color = .metric)) +
  geom_errorbar(aes(
    ymin = mean - std_err,
    ymax = mean + std_err
  )) +
  geom_line(size = 1.5) +
  facet_wrap(~.metric, scales = "free", nrow = 2) +
  scale_x_log10() +
  theme(legend.position = "none") +
  ggtitle("Modellens prestasjon for ulike verdier av lambda")+
  xlab("Straff(lambda)") +
  ylab("Gjennomsnit")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
```

```
i Please use `linewidth` instead.
```



Bruker den beste roc_auc verdien i modellen og tilpasser modellen med denne verdien.

```
beste_lamda <- select_best(lasso_tune, metric = "roc_auc")
beste_lamda
```

```
# A tibble: 1 x 2
  penalty .config
    <dbl> <chr>
1 0.00534 Preprocessor1_Model058
```

```
lasso_fit <- finalize_workflow(wflow_lasso, beste_lamda) |>
  fit(data = titanic_train)
```

Kjører prediksjoner med de nye parameterne

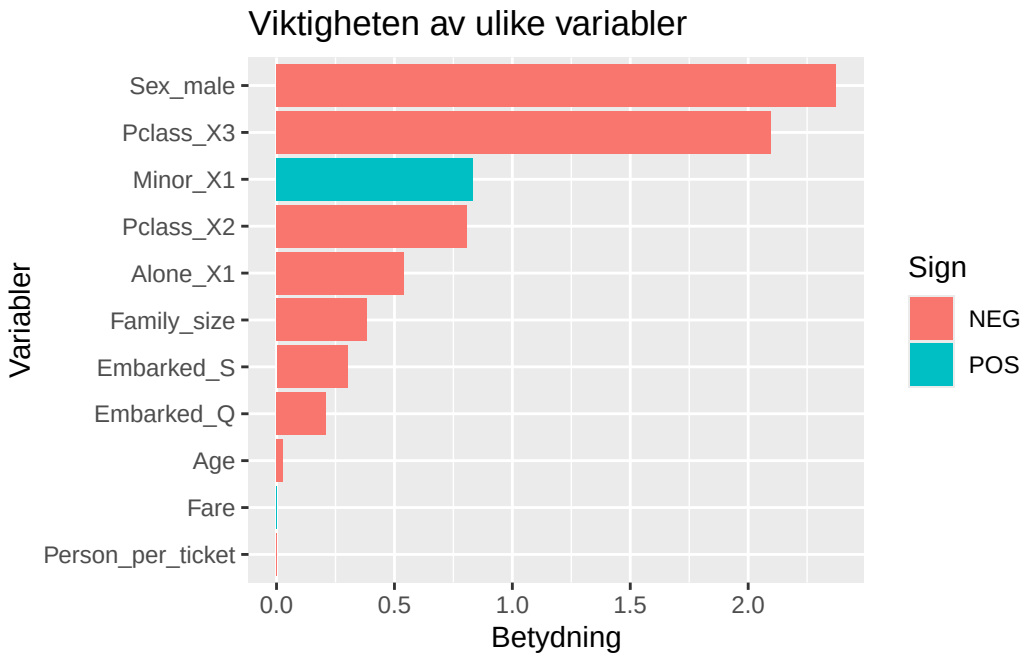
```
predict(lasso_fit, titanic_test) |>
  bind_cols(titanic_test) |>
  accuracy(truth = Survived, estimate = .pred_class)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.838
```

```
auc_lasso <-
  predict(lasso_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_auc(Survived, .pred_0) |>
  pull(.estimate)

lasso_roc <-
  predict(lasso_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_curve(Survived, .pred_0)
```

Finner de hvilke variabler som er viktigst for modellen.

```
lasso_fit |>
  extract_fit_parsnip() |>
  vip::vi() |>
  ggplot(aes(y = reorder(Variable, Importance),
             x = Importance, fill = Sign)) +
  geom_col() +
  theme() +
  ggtitle("Viktigheten av ulike variabler") +
  xlab("Betydning") +
  ylab("Variabler")
```



Konstruerer en forvirringsmatrise med prediksjonene fra modellen

```
library(cvms)

prediksjoner_lasso <- predict(lasso_fit, titanic_test) |>
  bind_cols(titanic_test)

forvirringsmatrise_lasso <- prediksjoner_lasso |>
  conf_mat(Survived, .pred_class)

riktigmatrise_lasso <- as.tibble(forvirringsmatrise_lasso$table) |>
  mutate(Prediction = ifelse(Prediction == 0, "Not survived", "Survived")) |>
  mutate(Truth = ifelse(Truth == 0, "Not survived", "Survived"))
```

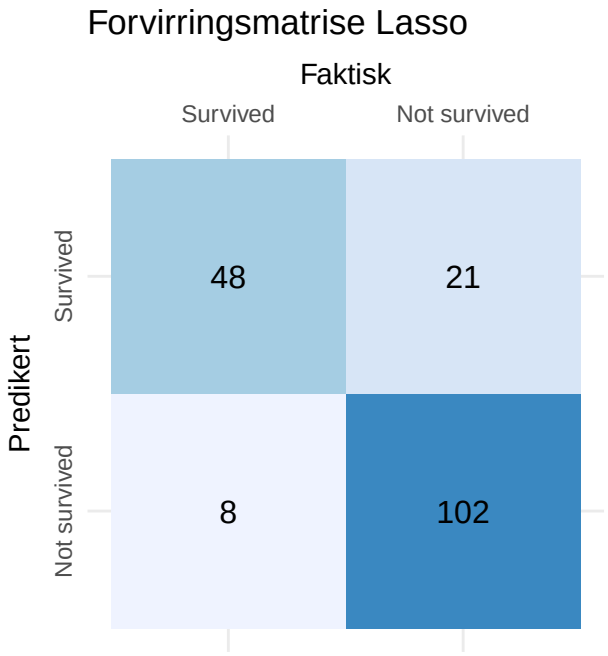
```
Warning: `as.tibble()` was deprecated in tibble 2.0.0.
i Please use `as_tibble()` instead.
i The signature and semantics have changed, see `?as_tibble`.
```

```
plot_confusion_matrix(riktigmatrise_lasso, "Prediction", "Truth", "n",
                      add_normalized = F,
                      add_row_percentages = F,
                      add_col_percentages = F) +
  xlab("Faktisk") +
```

```
  ylab("Predikert") +
  ggtitle("Forvirringsmatrise Lasso")
```

```
Warning in plot_confusion_matrix(riktigmatrise_lasso, "Prediction", "Truth", :
'ggimage' is missing. Will not plot arrows and zero-shading.
```

```
Warning in plot_confusion_matrix(riktigmatrise_lasso, "Prediction", "Truth", :
'rsvg' is missing. Will not plot arrows and zero-shading.
```

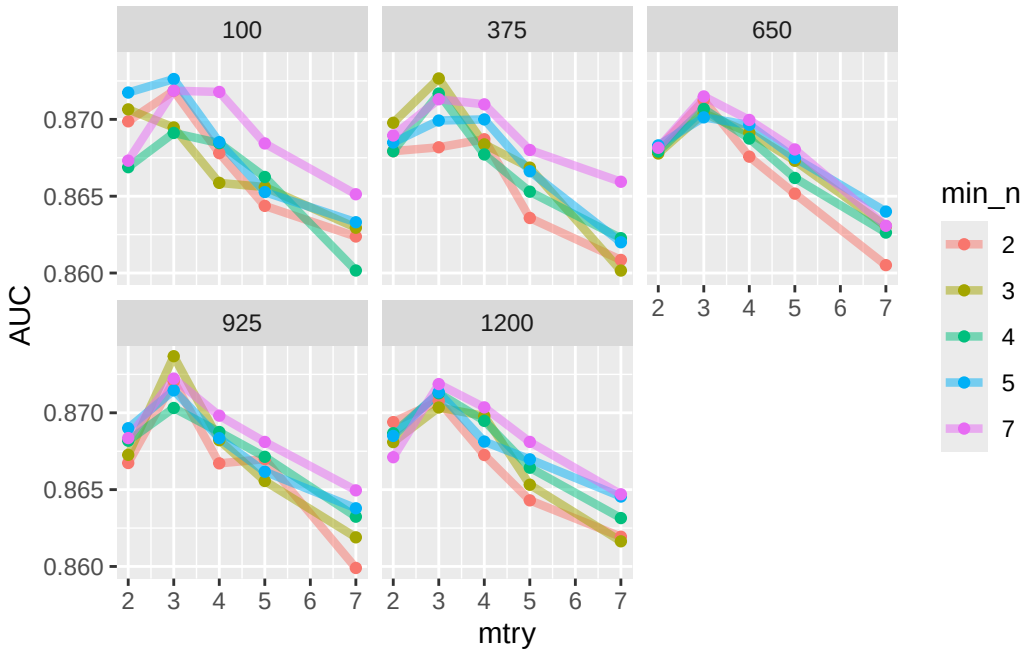


Random forest

Definerer modellen, lager en workflow med samme recipe, definerer en grid og tilpasser hyperparametere.

```
set.seed(3170)

rf_mod <- rand_forest(
  mtry = tune(),
  trees = tune(),
  min_n = tune()
) |>
```

```
  set_engine("ranger", importance = "impurity") |>
  # For å kunne bruke Vip pakken
  set_mode("classification")



rf_wflow <- workflow() |>
  add_recipe(titanic_recipe) |>
  add_model(rf_mod)


rf_grid <- grid_regular(
  mtry(range = c(2, 7)),
  min_n(range = c(2,7)),
  trees(range = c(100,1200)),
  levels = 5
)


doParallel::registerDoParallel()

tune_rf <- tune_grid(
  rf_wflow,
  resamples = folds,
  grid = rf_grid
)

tune_rf |>
  collect_metrics() |>
  filter(.metric == "roc_auc") |>
  mutate(min_n = factor(min_n)) |>
  ggplot(aes(mtry, mean, color = min_n)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC") +
  facet_wrap(vars(trees))
```



Vi ser her hvilke kombinasjoner som gir høyest treffsikkerhet.

Bruker resultatene fra tilpassingen til å trene modellen, og kjøre prediksjoner med den trente modellen.

```
set.seed(3170)
beste_mtry_min_tree <- select_best(tune_rf, metric = "roc_auc")

rf_fit <- finalize_workflow(rf_wflow, beste_mtry_min_tree) |>
  fit(titanic_train)


pred.rf <- predict(rf_fit, titanic_test) |>
  bind_cols(titanic_test) |>
  accuracy(Survived, .pred_class)
pred.rf
```

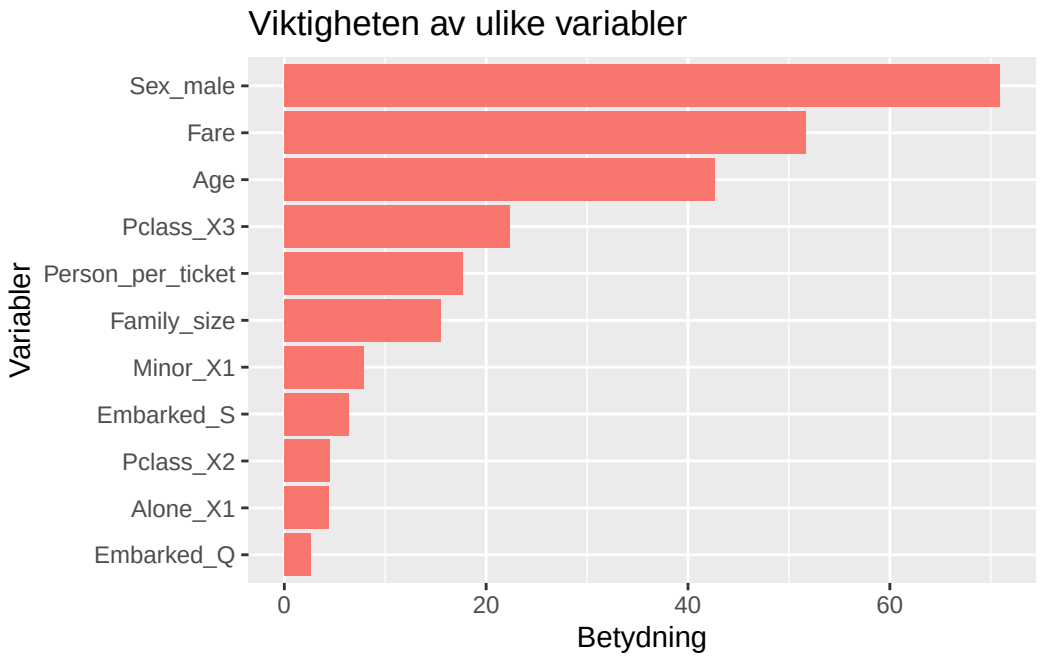
```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.844
```

Finner AUC og ROC for prediksjonen

```r
auc_rf <-
  predict(rf_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_auc(Survived, .pred_0) |>
  pull(.estimate)

rf_roc <- predict(rf_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_curve(Survived, .pred_0)
```

Finner hvilke variabler som er viktige for prediksjonen

```r
rf_fit |>
  extract_fit_parsnip() |>
  vip::vi() |>
  ggplot(aes(y = reorder(Variable, Importance), x = Importance, fill = "#F8766D")) +
  geom_col() +
  theme(legend.position = "none") +
  ggtitle("Viktigheten av ulike variabler") +
  xlab("Betydning") +
  ylab("Variabler")
```



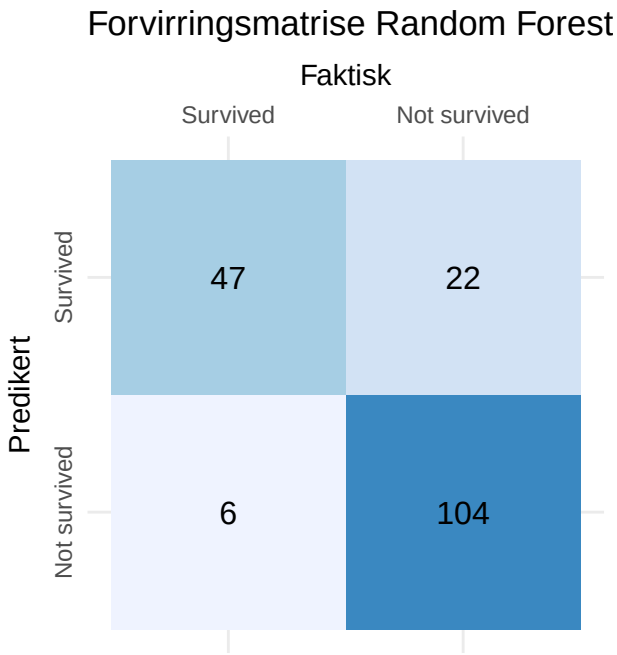
Denne modellen trekker frem - i likhet med LASSO - at kjønn er viktigst for prediksjonen. I kontrast til LASSO vekter random forest modellen variabler som Fare og Age høyt, og bryr seg mindre om klassen til passasjeren.

Konstruerer så en forvirringsmatrise for Random Forest modellen

```
prediksjoner_rf <- predict(rf_fit, titanic_test) |>
  bind_cols(titanic_test)

forvirringsmatrise_rf <- prediksjoner_rf |>
  conf_mat(Survived, .pred_class)

riktigmatrise_rf <- as.tibble(forvirringsmatrise_rf$table) |>
  mutate(Prediction = ifelse(Prediction == 0, "Not survived", "Survived")) |>
  mutate(Truth = ifelse(Truth == 0, "Not survived", "Survived"))

plot_confusion_matrix(riktigmatrise_rf, "Prediction", "Truth", "n",
                      add_normalized = F,
                      add_row_percentages = F,
                      add_col_percentages = F) +
  xlab("Faktisk") +
  ylab("Predikert") +
  ggtitle("Forvirringsmatrise Random Forest")
```

```
Warning in plot_confusion_matrix(riktigmatrise_rf, "Prediction", "Truth", :
'ggimage' is missing. Will not plot arrows and zero-shading.
```

```
Warning in plot_confusion_matrix(riktigmatrise_rf, "Prediction", "Truth", :
'rsvg' is missing. Will not plot arrows and zero-shading.
```



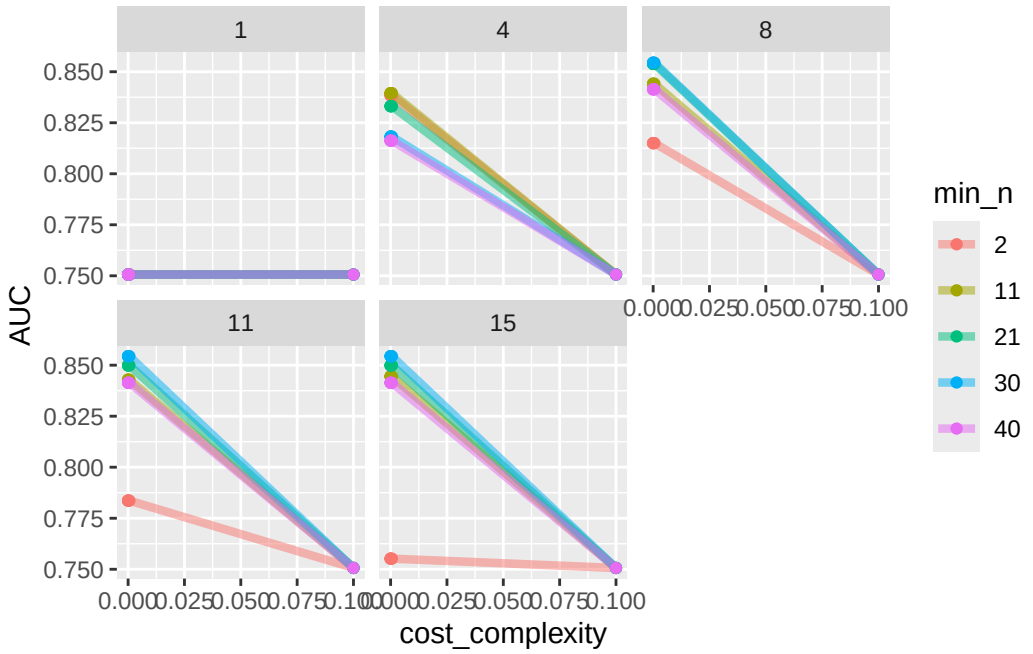
Decision tree

Følger de samme prosedyrene som ved tidligere modeller

```
decision_model <- decision_tree(
  cost_complexity = tune(),
  tree_depth = tune(),
  min_n = tune()) |>
  set_engine("rpart") |>
  set_mode("classification")



decision_wflow <- workflow() |>
  add_recipe(titanic_recipe) |>
  add_model(decision_model)

decision_tree_params <- parameters(decision_model) |>
  update(
    cost_complexity = cost_complexity(),
    tree_depth = tree_depth(),
    min_n = min_n()
  )
```

```
Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
i Please use `hardhat::extract_parameter_set_dials()` instead.
```

```r
decision_grid <- grid_regular(
  decision_tree_params,
  levels = 5
)

doParallel::registerDoParallel()

tune_decision <- tune_grid(
  decision_wflow,
  resamples = folds,
  grid = decision_grid,
)

tune_decision |>
  collect_metrics() |>
  filter(.metric == "roc_auc") |>
  mutate(min_n = factor(min_n)) |>
  ggplot(aes(cost_complexity, mean, color = min_n)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC") +
  facet_wrap(vars(tree_depth))
```



Lar tidymodels ta seg av ranges på ulike parameterer. Noe som vi ikke gjorde for random forest eller lasso.

```
best_cost_depth_min <- select_best(tune_decision, metric = "roc_auc")
best_cost_depth_min
```

```
# A tibble: 1 x 4
  cost_complexity tree_depth min_n .config
            <dbl>      <int> <int> <chr>
1    0.0000000001          8    30 Preprocessor1_Model086
```

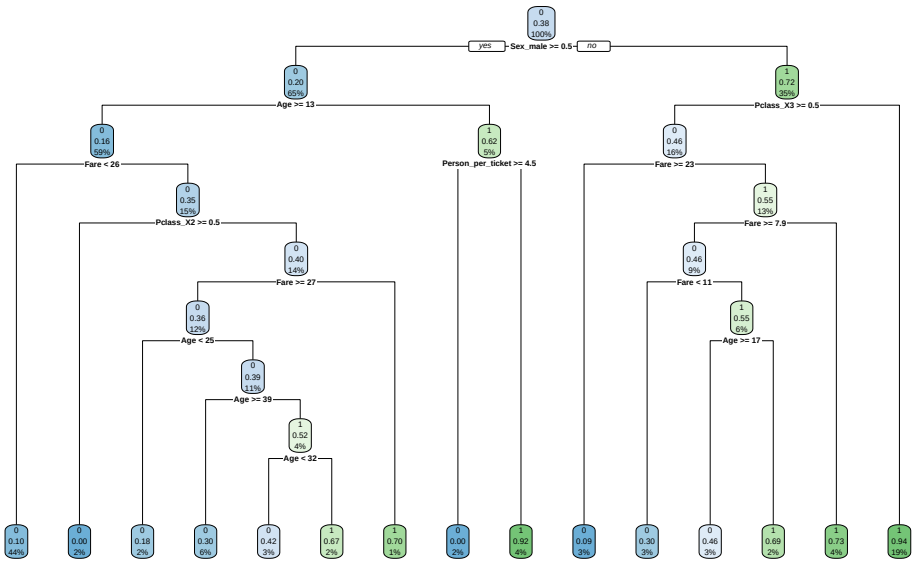
```
decision_fit <- finalize_workflow(decision_wflow, best_cost_depth_min) |>
  fit(titanic_train)


predict(decision_fit, titanic_test) |>
  bind_cols(titanic_test) |>
  accuracy(Survived, .pred_class)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.827
```

Dårligste Accuarcy så langt

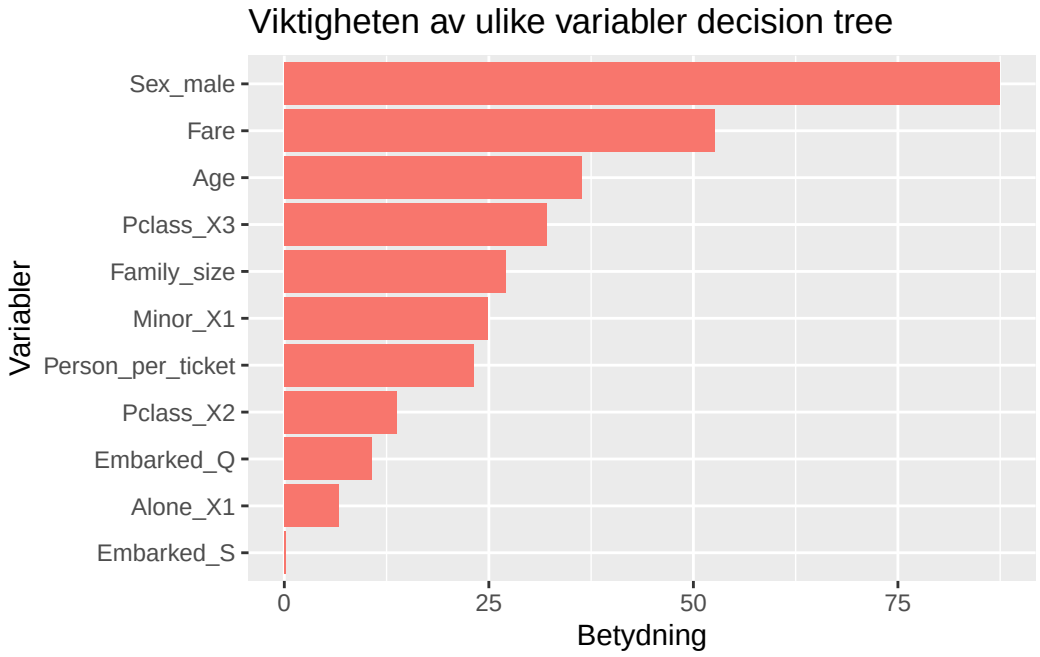
```
auc_decision <-
  predict(decision_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_auc(Survived, .pred_0) |>
  pull(.estimate)

decision_roc <- predict(decision_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_curve(Survived, .pred_0)
```

```
library(rpart)
library(rpart.plot)

decision_final_fit <- finalize_workflow(decision_wflow, best_cost_depth_min) |>
  last_fit(titanic_split)

final_tree <- extract_workflow(decision_final_fit)

final_tree |>
  extract_fit_engine() |>
  rpart.plot(roundint = FALSE)
```



For å bruke funkjsonen til rpart, må vi fitte på en annen måte, vi kunne ha brukt denne metodene med last_fit gjennom hele, men vi syntes det var greiere å kun jobbe med fit, isteden for å kjøre last_fit

```r
decision_fit |>
  extract_fit_parsnip() |>
  vip::vi() |>
  ggplot(aes(y = reorder(Variable, Importance), x = Importance, fill = "#F8766D")) +
  geom_col() +
  theme(legend.position = "none") +
  ggtitle("Viktigheten av ulike variabler decision tree") +
  xlab("Betydning") +
  ylab("Variabler")
```

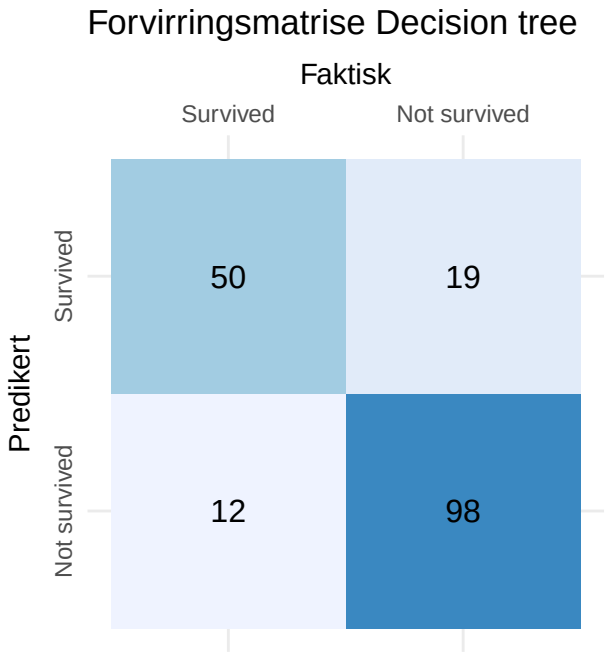



```r
prediksjoner_decision <- predict(decision_fit, titanic_test) |>
  bind_cols(titanic_test)

forvirringsmatrise_decision <- prediksjoner_decision |>
  conf_mat(Survived, .pred_class)

riktigmatrise_decision <- as.tibble(forvirringsmatrise_decision$table) |>
  mutate(Prediction = ifelse(Prediction == 0, "Not survived", "Survived")) |>
  mutate(Truth = ifelse(Truth == 0, "Not survived", "Survived"))

plot_confusion_matrix(riktigmatrise_decision, "Prediction", "Truth", "n",
                      add_normalized = F,
                      add_row_percentages = F,
```

```
                      add_col_percentages = F) +
  xlab("Faktisk") +
  ylab("Predikert") +
  ggtitle("Forvirringsmatrise Decision tree")
```

```
Warning in plot_confusion_matrix(riktigmatrise_decision, "Prediction", "Truth",
: 'ggimage' is missing. Will not plot arrows and zero-shading.
```

```
Warning in plot_confusion_matrix(riktigmatrise_decision, "Prediction", "Truth",
: 'rsvg' is missing. Will not plot arrows and zero-shading.
```



Xgboost

```
xg_model <- boost_tree(
  trees = 1000,
  tree_depth = tune(),
  learn_rate = tune(),
  mtry = tune(),
  min_n = tune(),
  loss_reduction = tune(),
```

```
  sample_size = tune()
  )|>
  set_engine("xgboost") |>
  set_mode("classification")


xg_wflow <- workflow() |>
  add_recipe(titanic_recipe) |>
  add_model(xg_model)

xg_grid <- grid_latin_hypercube(
  tree_depth(),
  learn_rate(),
  finalize(mtry(), titanic_train),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  size = 50
)
```

```
Warning: `grid_latin_hypercube()` was deprecated in dials 1.3.0.
i Please use `grid_space_filling()` instead.
```

```
doParallel::registerDoParallel()

tune_xg <- tune_grid(
  xg_wflow,
  resamples = folds,
  grid = xg_grid,
)

best_param_xg <- select_best(tune_xg, metric = "roc_auc")
best_param_xg
```

```
# A tibble: 1 x 7
   mtry min_n tree_depth learn_rate loss_reduction sample_size .config
  <int> <int>      <int>      <dbl>          <dbl>       <dbl> <chr>
1     6     9         11     0.0991     0.00000154       0.483 Preprocessor1_Mo~
```

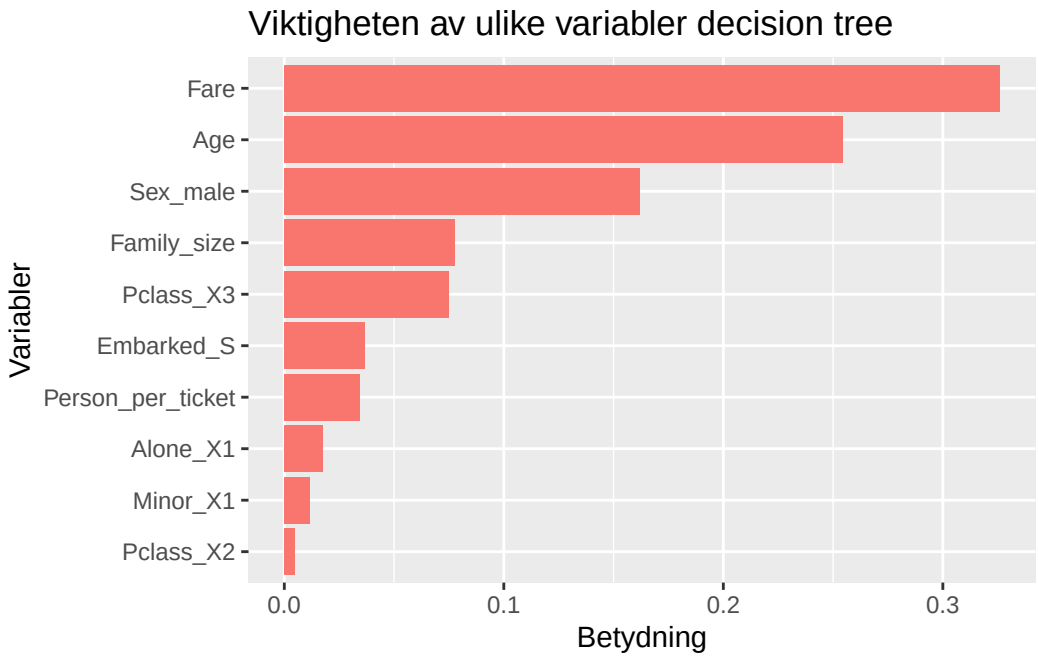
grid: https://juliasilge.com/blog/xgboost-tune-volleyball/, blir litt vanskelig å plotte

```r
xg_fit <- finalize_workflow(xg_wflow, best_param_xg) |>
  fit(titanic_train)

predict(xg_fit, titanic_test) |>
  bind_cols(titanic_test) |>
  accuracy(Survived, .pred_class)
```

```
# A tibble: 1 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.855
```

```r
auc_xg <-
  predict(xg_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_auc(Survived, .pred_0) |>
  pull(.estimate)

xg_roc <- predict(xg_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_curve(Survived, .pred_0)
```

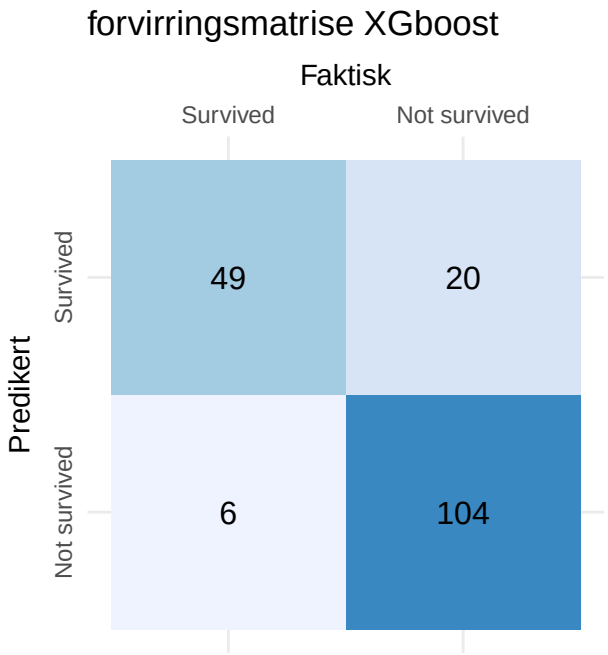
```r
xg_fit |>
  extract_fit_parsnip() |>
  vip::vi() |>
  ggplot(aes(y = reorder(Variable, Importance), x = Importance, fill = "#F8766D")) +
  geom_col() +
  theme(legend.position = "none") +
  ggtitle("Viktigheten av ulike variabler decision tree") +
  xlab("Betydning") +
  ylab("Variabler")
```




```
prediksjoner_xg <- predict(xg_fit, titanic_test) |>
  bind_cols(titanic_test)

forvirringsmatrise_xg <- prediksjoner_xg |>
  conf_mat(Survived, .pred_class)

riktigmatrise_xg <- as.tibble(forvirringsmatrise_xg$table) |>
  mutate(Prediction = ifelse(Prediction == 0, "Not survived", "Survived")) |>
  mutate(Truth = ifelse(Truth == 0, "Not survived", "Survived"))

plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", "n",
                      add_normalized = F,
                      add_row_percentages = F,
                      add_col_percentages = F) +
  xlab("Faktisk") +
  ylab("Predikert") +
  ggtitle("forvirringsmatrise XGboost")
```

```
Warning in plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", :
'ggimage' is missing. Will not plot arrows and zero-shading.
```

```
Warning in plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", :
'rsvg' is missing. Will not plot arrows and zero-shading.
```



MLP

Starter med å definere modell, og lage en workflow.

```
mlp.model <- mlp(hidden_units = tune(), penalty = tune(), epochs = tune()) |>
  set_engine("nnet") |>
  set_mode("classification")

mlp_wflow <- workflow() |>
  add_model(mlp.model) |>
  add_recipe(titanic_recipe)
```

Henter så ut hyperparameterne og implempenterer de i en grid ved hjelp av grid_latin_hypercube med 50 nivåer

```
mlp_params <- extract_parameter_set_dials(mlp_wflow)
meterics <- metric_set(roc_auc, accuracy, brier_class)

grid <- grid_latin_hypercube(
  mlp_params,
  size = 50
)
```

```
doParallel::registerDoParallel()

mlp_tune <- tune_grid(
  mlp_wflow,
  resamples = folds,
  grid = grid,
  metrics = meterics,
  control = control_grid(save_pred = TRUE)
)
```

Henter deretter ut de beste parameterne, og tilpasser modellen

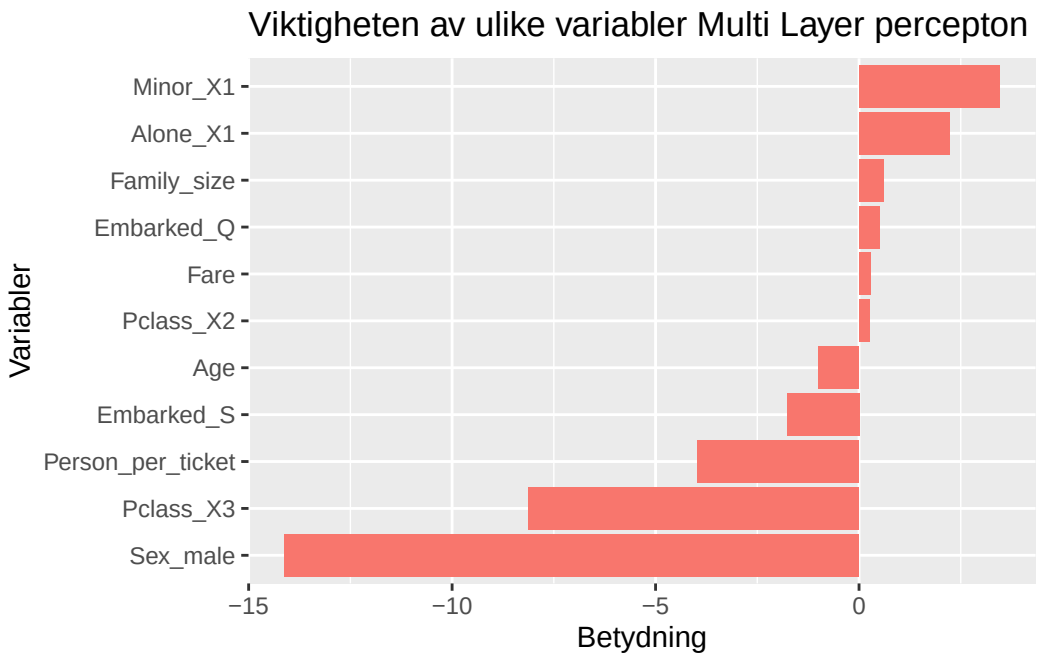
```
logistic_param.reg <- select_best(mlp_tune, metric = "roc_auc") |>
  select(-.config)

fn.mlp_wflow <- mlp_wflow |>
  finalize_workflow(logistic_param.reg)

fn.mlp_fit <- fn.mlp_wflow |>
  fit(titanic_train)
```

Kjører prediksjoner, og plotter ROC-AUC.

```
auc_mlp <-
  predict(fn.mlp_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_auc(Survived, .pred_0) |>
  pull(.estimate)

mlp_roc <- predict(fn.mlp_fit, titanic_test, type = "prob") |>
  bind_cols(titanic_test) |>
  roc_curve(Survived, .pred_0)
```

```
library(NeuralNetTools)

fn.mlp_fit |>
  extract_fit_parsnip() |>
  vip::vi() |>
  ggplot(aes(y = reorder(Variable, Importance), x = Importance, fill = "#F8766D"))+
  geom_col() +
  theme(legend.position = "none") +
```

```
  ggtitle("Viktigheten av ulike variabler Multi Layer percepton") +
  xlab("Betydning") +
  ylab("Variabler")
```

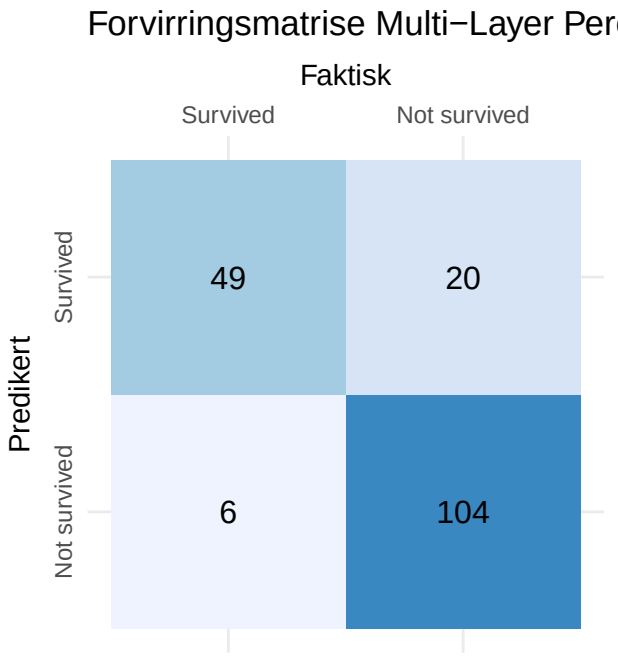



```
prediksjoner_mlp <- predict(fn.mlp_fit, titanic_test) |>
  bind_cols(titanic_test)

forvirringsmatrise_xg <- prediksjoner_xg |>
  conf_mat(Survived, .pred_class)

riktigmatrise_xg <- as.tibble(forvirringsmatrise_xg$table) |>
  mutate(Prediction = ifelse(Prediction == 0, "Not survived", "Survived")) |>
  mutate(Truth = ifelse(Truth == 0, "Not survived", "Survived"))

plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", "n",
                      add_normalized = F,
                      add_row_percentages = F,
                      add_col_percentages = F) +
  xlab("Faktisk") +
  ylab("Predikert") +
  ggtitle("Forvirringsmatrise Multi-Layer Percepton")
```

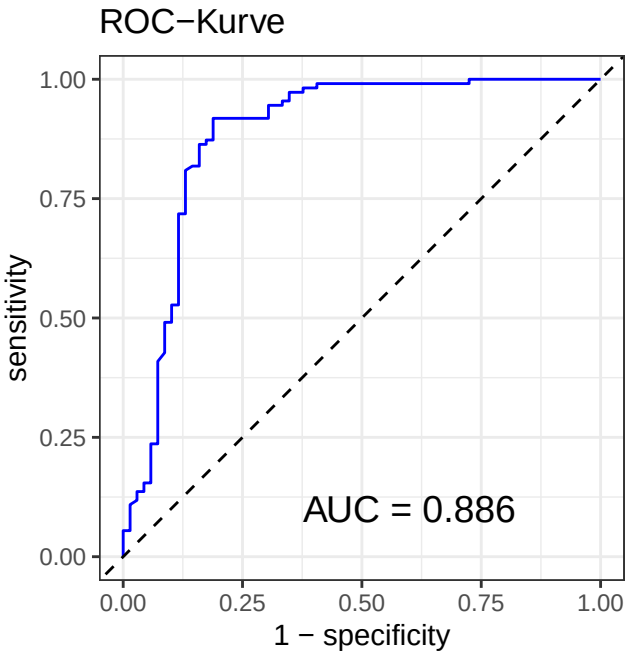
```
Warning in plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", :
```

```
'ggimage' is missing. Will not plot arrows and zero-shading.
```

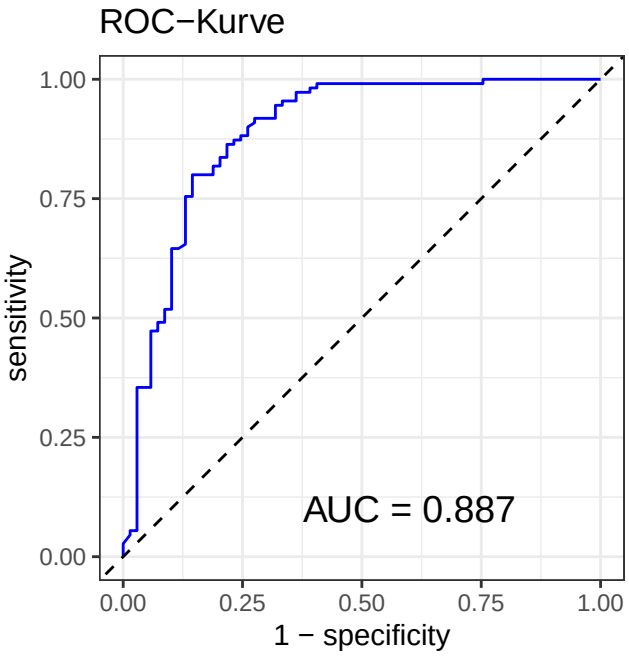
```
Warning in plot_confusion_matrix(riktigmatrise_xg, "Prediction", "Truth", :
'rsvg' is missing. Will not plot arrows and zero-shading.
```



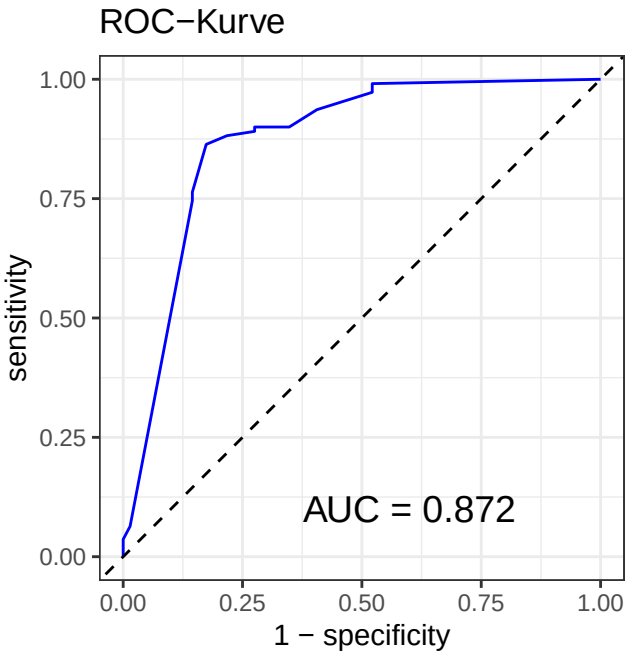

```
par(mfrow = c(4,2))

lasso_roc |>
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path(col = "blue") +
  geom_abline(slope = 1, linetype = "dashed") +
  coord_equal()+
  annotate("text", x = 0.6, y = 0.1,
           label = paste("AUC =", round(auc_lasso,3) ), size = 5) +
  theme_bw() +
  ggtitle("ROC-Kurve")
```



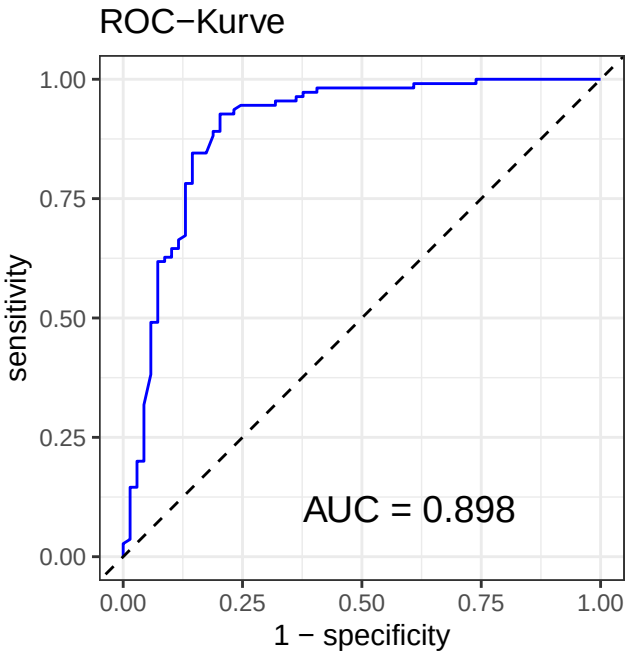

```
rf_roc|>
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path(col = "blue") +
  geom_abline(slope = 1, linetype = "dashed") +
  coord_equal()+
  annotate("text", x = 0.6, y = 0.1, label = paste("AUC =", round(auc_rf,3) ), size = 5) +
  theme_bw() +
  ggtitle("ROC-Kurve")
```



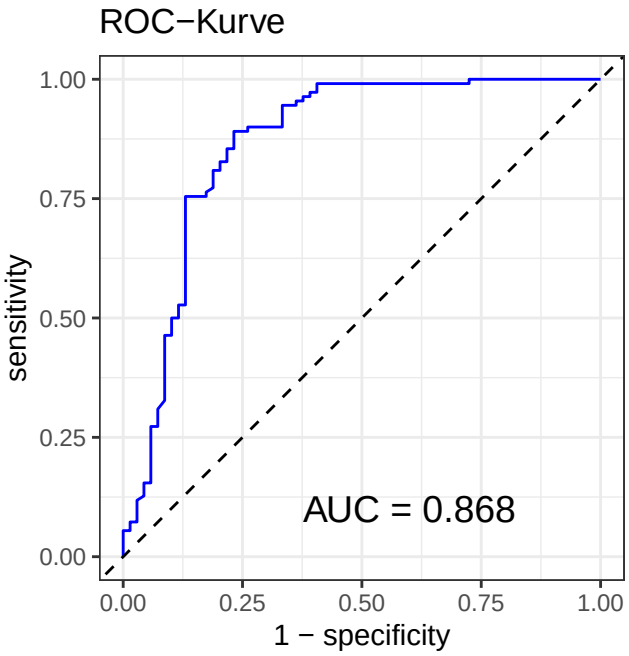

```
decision_roc|>
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path(col = "blue") +
  geom_abline(slope = 1, linetype = "dashed") +
  coord_equal()+
  annotate("text", x = 0.6, y = 0.1,
           label = paste("AUC =", round(auc_decision,3) ), size = 5) +
  theme_bw() +
  ggtitle("ROC-Kurve")
```



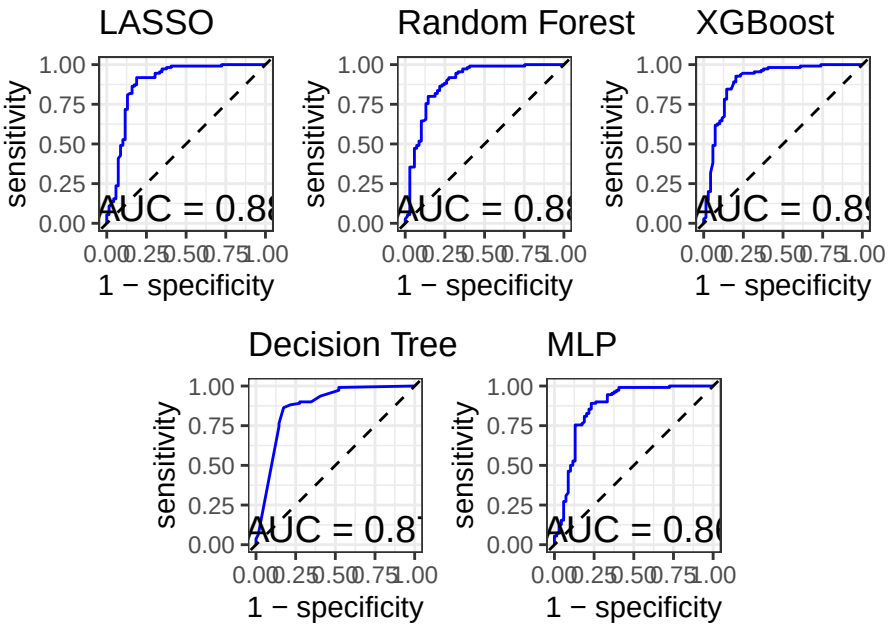

```
xg_roc|>
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path(col = "blue") +
  geom_abline(slope = 1, linetype = "dashed") +
  coord_equal()+
  annotate("text", x = 0.6, y = 0.1,
           label = paste("AUC =", round(auc_xg,3) ), size = 5) +
  theme_bw() +
  ggtitle("ROC-Kurve")
```




```
mlp_roc |>
   ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path(col = "blue") +
  geom_abline(slope = 1, linetype = "dashed") +
  coord_equal()+
  annotate("text", x = 0.6, y = 0.1,
           label = paste("AUC =", round(auc_mlp, 3) ), size = 5) +
  theme_bw() +
  ggtitle("ROC-Kurve")
```




```
library(patchwork)
# Define a function to plot ROC curves
plot_roc_curve <- function(roc_data, auc_value, title) {
  roc_data |>
    ggplot(aes(x = 1 - specificity, y = sensitivity)) +
    geom_path(col = "blue") +
    geom_abline(slope = 1, linetype = "dashed") +
    coord_equal() +
    annotate("text", x = 0.6, y = 0.1,
             label = paste("AUC =", round(auc_value, 3)), size = 5) +
    theme_bw() +
    ggtitle(title)
}

# Create individual plots
lasso_plot <- plot_roc_curve(lasso_roc, auc_lasso, "LASSO")
rf_plot <- plot_roc_curve(rf_roc, auc_rf, "Random Forest")
decision_plot <- plot_roc_curve(decision_roc, auc_decision, "Decision Tree")
xg_plot <- plot_roc_curve(xg_roc, auc_xg, "XGBoost")
mlp_plot <- plot_roc_curve(mlp_roc, auc_mlp, "MLP")

# Combine them into one figure in a single row
(lasso_plot | rf_plot | xg_plot) / (decision_plot | mlp_plot)
```




```
# Display the combined plot
```

Vi ser her at modellene som gjør det best er LASSO, Random-Forest og XGboost, dette gir mening ettersom at

Kilder

- Kaggle. *Titanic - Machine Learning from Disaster*. Kaggle. Hentet 1. november 2024 fra https://www.kaggle.com/competitions/titanic
- Allohvk. *Titanic - Advanced EDA*. Kaggle. Hentet 1. november 2024 fra https://www.kaggle.com/code/allohvk/titanic-advanced-eda?scriptVersionId=77739368
- Donges, N. (2018). *Predicting the survival of Titanic passengers*. Towards Data Science. Hentet 4. november 2024 fra https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8