# Political Data Science

## Lektion 10: Superviseret læring I

Undervist af Jesper Svejgaard, foråret 2018
Institut for Statskundskab, Københavns Universitet
github.com/jespersvejgaard/PDS

# I dag

1. Overblik og opsamling fra sidst
2. Dagens pensum
3. Workshop
4. Opsamling og næste gang

# Overblik

1. Intro til kurset og R
2. R Workshop I: Explore
3. R Workshop II: Import, tidy, transform
4. R Workshop III: Programmering & Git
5. Web scraping & API
6. Tekst som data
7. Visualisering
8. GIS & spatiale data
9. Estimation & prædiktion
10. Superviseret læring I
11. Superviseret læring II
12. Usuperviseret læring
13. Refleksioner om data science
14. Opsamling og eksamen

# Opsamling fra sidst I

Find løsning på opgaverne fra sidste uge på GitHub:

- `PDS/scripts/09_script.pdf`

(ligger i scripts-mappen, selvom det er en .pdf)

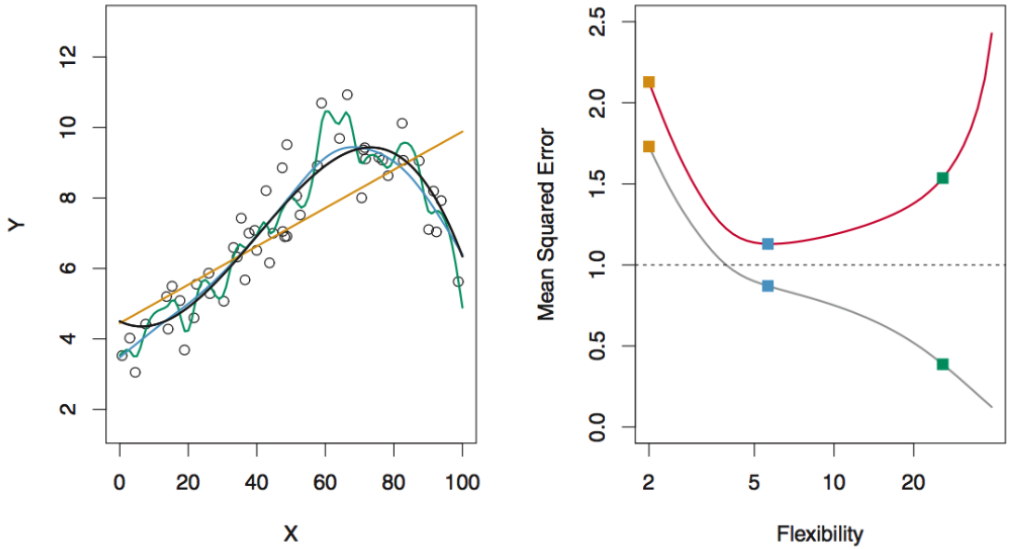
# Opsamling fra sidst II

Hvad er de mest centrale forskelle på en estimations- og prædiktionstilgang?

Diskutér med din sidemakker.

# Opsamling fra sidst III

Hvilke indsigter illustreres i figuren her? Snak med din sidemakker.



# Overblik

Fokus sidste gang:

- **Konceptuelt:** Estimation/prædiktion, out-of-sample performance, performancemål, den datagenererende proces, bias-variance tradeoff

Fokus i dag:

- **Konceptuelt:** Logistisk regression, regularisering, tuning, resampling-metoder
- **Praktisk:** Prædiktion med OLS og logit + evaluere performance

Fokus næste gang:

- **Konceptuelt:** Algoritmer baseret på klassifikations/regressionstræer
- **Praktisk:** Kulmination: prædiktion med træ-algoritmer, krydsvalidering, tuning

# Superviseret læring

# Superviseret læring

## Algoritmer

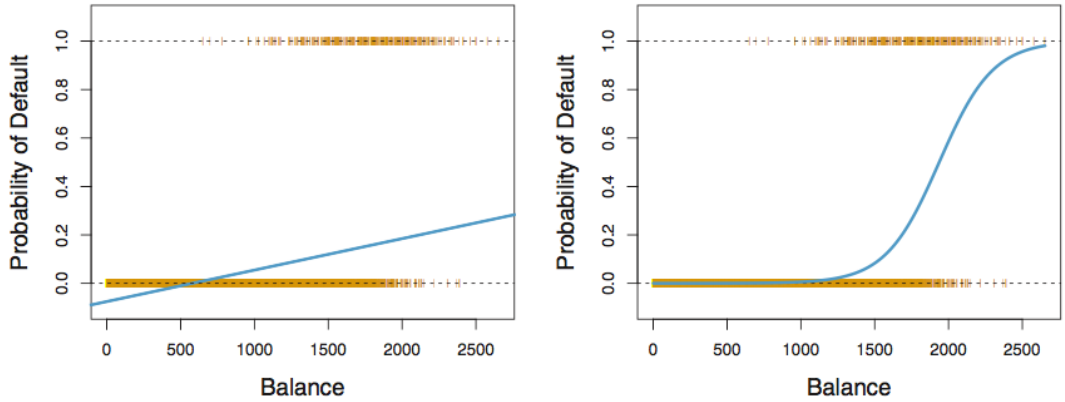
Lærebøgerne starter typisk med:

- Lineær regression (OLS)
- Logistisk regression (logit)

… så det gør vi også!

# Superviseret læring

## Logistisk regression: Illustration




- Lineær regression kan forudsige værdier for $\hat{Y}$ uden for intervallet [0, 1]
- Logistisk regression begrænser $\hat{Y}$ til intervallet [0,1]
- Vi bruger ofte logistisk regression når outcome er kategorisk

# Superviseret læring

## Logistisk regression

En logistisk regressionsmodel kan se ud som her:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1}}{1 + e^{\beta_0+\beta_1 X_1}}$$

Hvilket vi kan transformere til en logit model (log odds), der er lineær i X:

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1$$

Fortolkning:

- Når $X_1$ ændrer sig med 1, ændrer log odds sig med $\beta_1$.
- Fortegnet for log-odds det samme på tværs af værdierne for X er f.

# Superviseret læring

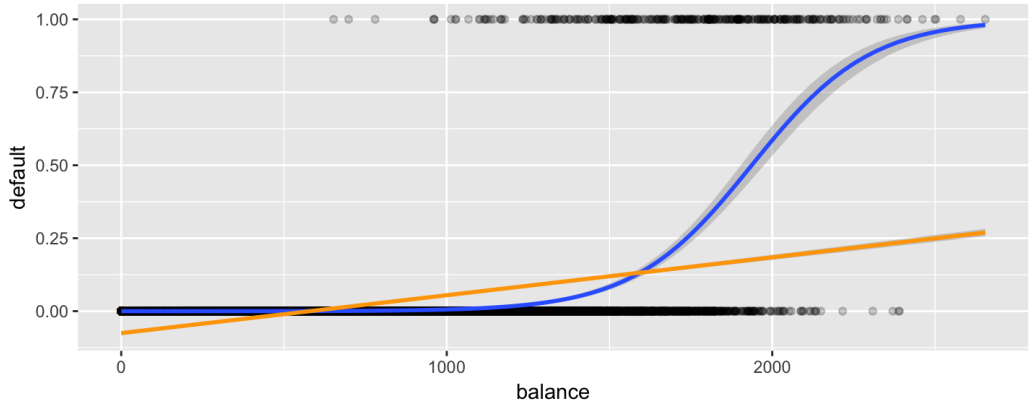
## Logistisk regression: Eksempel i R

```
# Loader pakker og datasættet 'Default'
library(ISLR); library(ggplot2); library(dplyr)

# Transformerer y til numerisk
Default <- Default %>% mutate(default = ifelse(default == "Yes", 1, 0))

# Fitter modeller
m_ols <- lm(default ~ balance, data = Default)
m_logit <- glm(default ~ balance, data = Default, family = "binomial")

# Prædikterer y
Default$pred_ols <- predict(m_ols, Default)
Default$pred_logit <- predict(m_logit, Default, type = "response")
```

# Superviseret læring

## Logistisk regression: Eksempel i R

```
ggplot(Default, aes(x = balance, y = default)) + geom_point(alpha = 0.2) +
  geom_smooth(method = "glm", method.args = list(family="binomial")) +
  geom_smooth(method = "lm", color = "orange")
```



# Regularisering & tuning

# Regularisering & tuning

## Regularisering

Hvad er regularisering?

- Modellering af en models fleksibilitet mhp. begrænsning af koefficienter
- Vi modellerer tradeoff ml. bias og varians

Hvorfor regularisering?

- Vi så sidste gang, at højere varians for en model $\Rightarrow$ dårligere out-of-sample performance
- Ved at begrænse koefficienterne, fx 'shrinke' dem mod 0, mindsker vi variansen

Eksempel:

- Antag $y = frafald$, $x_1 = alder$ og $x_2 = køn$,
- Resulterende i $\hat{\beta}_1 = 1 \pm 0.02$ og $\hat{\beta}_2 = 15 \pm 40$
- Shrinkage af $\hat{\beta}_2$ => muligvis højere performance out-of-sample

# Regularisering & tuning

## Shrinkage

Eksempel: Ridge regression (L2)

$$\hat{f}_{L2} = \arg\min f \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Logik:

- Større $\beta$-koefficienter => højere varians
- L2-termen biaser modellen mod mindre $\beta$-koefficienter => højere bias + mindre varians
- $\lambda$ sætter prisen på kompleksitet … men hvordan fastsætter vi så $\lambda$?

# Regularisering & tuning

## Tuning

Hvad er tuning?

- Fastsættelse af frie parametre empirisk (≠ teoridrevet)
- Vi kan måle prædiktionsperformance => vi kan teste kombinationer af parameterværdier

Hvordan måler vi bedst prædiktionsperformance:

- Sidste gang: vi stødte på MSE, accuracy og AUC
- Gennemgående præmis: prædiktion er korrelationsbaseret
- Fællestræk: fokus på out-of-sample performance
- I dag: teknikker til at estimere out-of-sample performance i form af **resampling metoder**

# Resampling metoder

# Resampling-metoder

- **Sidste uge**: in-sample performance ≠> out-of-sample performance
- **Resampling-metoder**: Teknikker til estimation af fx en models varians eller performance
- **Tricket**: Trække samples fra et datasæt og fitter samme model flere gange
- **Formål**: Fx estimere out-of-sample performance og tune parameter-værdier

# Resampling-metoder

## Metode 1: Validation set

Koncept:

1. Split datasættet i to dele, et træningssæt og valideringssæt.
2. Træn modellen på træningssættet og prædiktér outcome i valideringssættet
3. Beregn fejlen i valideringssættet, fx MSE, som er et estimat på out-of-sample performance

Ulempe:

- Den estimerede performance er betinget af splittet og varierer derfor
- Bruger ikke hele datasættet til at træne modellen (=> fejlen overestimeres)

# Resampling-metoder

## Metode 2: Leave-one-out cross-validation (LOOCV)

Koncept:

1. Split datasættet i $n$ dele
2. Udelad én observation $s$ og træn modellen på $n - 1$ observationer
3. Prædiktér outcome for den udeladte observation og beregn fejlen, fx MSE
4. Gentag for alle observationer $s$ indtil $s = n$, og beregn gns. af fejlene

Fordele:

- Den estimerede performance er den samme hver gang og varierer derfor ikke
- Vi overestimerer ikke fejlen som i validation set

Ulempe:

- Computationelt tungt

# Resampling-metoder

## Metode 3: k-fold cross validation

Koncept:

1. Inddel datasættet $k$ folder (subsets), typisk 5 eller 10, og kald dem fx: $s = 1 \ldots k$
2. Udelad fold $s$ og fit modellen på de resterende $k - 1$ folder
3. Prædiktér $y$ i det udeladte fold $k$
4. Gentag fra trin 2 og udelad fold $s + 1$ indtil $s = k$
5. Beregn den gennemsnitlige fejl på tværs af valideringssættene, fx MSE

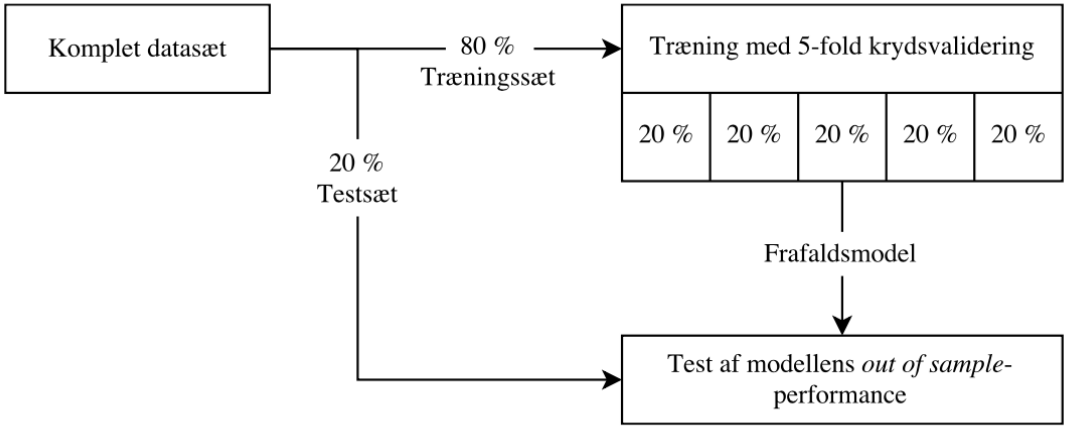
Fordel:

- Computationelt lettere end LOOCV
- Bedre performance end både validation set og LOOCV

# Resampling-metoder

## k-fold cross validation

Krydsvalidering illustreret:



# Resampling-metoder

## k-fold cross validation

Hvordan gør man?

1. Indbygget mulighed for CV i funktion til træning, fx `cv.glm()` og `cv.glmnet()`
2. Pakken `caret` kan wrappes om de modeller, man træner, og foretage CV (næste uge)
3. Distinkte funktioner til opdeling i folds, fx `kWayCrossValidation()`

# Resampling-metoder

## Eksempel i R: k-fold cross-validation

```r
# Loader pakker
library(vtreat)

# Splitter i 5 folds
k <- 5
folds <- kWayCrossValidation(nRows = nrow(Default), nSplit = k)

# Laver kolonne til prædikterede værdier
Default$pred_cv <- 0

# Looper igennem alle folds: Fitter på træningssættet og tester på udeladt fold
for(i in 1:k) {
  fold <- folds[[i]]
  model <- glm(default ~ balance, data = Default[fold$train, ], family = "binomial")
  Default$pred_cv[fold$app] <- predict(model, newdata = Default[fold$app, ], type = "response")
}
```

# Resampling-metoder

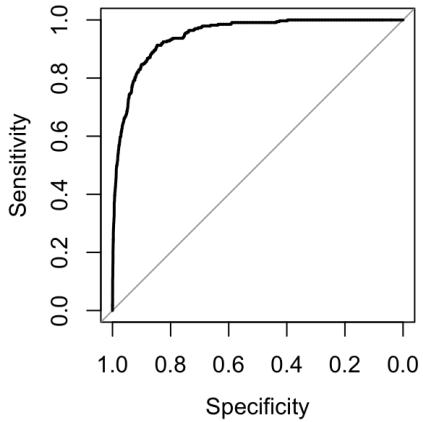
## Eksempel i R: k-fold cross-validation

```
# Loader pakke til illustration af performance
library(pROC)

# Beregner punkter til ROC-kurver
ROC <- roc(Default$default, Default$pred_cv)
```

# Resampling-metoder

## Eksempel i R: k-fold cross-validation

```
# Plotter ROC-kurve
plot(ROC)
```



# Resampling-metoder

## Eksempel i R: k-fold cross-validation

```
# Beregner AUC
auc(ROC)
```

```
## Area under the curve: 0.9471
```

# Workshop

# Workshop

Fokus i opgaver:

- Indlæse data og opdele i test- og træning
- Træne modeller og prædiktere outcome
- Evaluere performance (validation set-tilgangen)

Find opgaverne på Github under PDS/opgaver/:

- `10_opgaver.R`

# Næste gang

# Næste gang

- Indhold:
  - Superviseret læring II
- Pensum:
  - ISL: kap 8 afs 8.1-8.2
  - Montgomery & Olivella (2015)
- DataCamp:
  - Machine Learning Toolbox

# Tak for i dag!