

Klassifikasjon

ISTx1003 Statistisk læring og Data Science

Stefanie Muff, Institutt for matematiske fag

November 5 og 8, 2021

Anerkjennelse

Disse slides bygger på slides fra Mette Langaas, 2020.

Takk til Mette for at jeg fikk bruke noen av materialene.

Plan tema “Klassifikasjon”

- Læringsmål og ressurser
- Hva er klassifikasjon?
- Trening, validering og testing (3 datasett)
- k -nærmeste nabo (KNN): en intuitiv metode
- Forvirringsmatrise og feilrate for å evaluere metoden
- Logistisk regresjon

Læringsmål

- kunne forstå hva klassifikasjon går ut på og kjenne situasjoner der klassifikasjon vil være en aktuell metode å bruke
- kjenne begrepene treningssett, valideringssett og testsett og forstå hvorfor vi lager dem og hva de skal brukes til
- vite hva en forvirringsmatrise er, og kjenne til begrepene *feilrate* (error rate) og *nøyaktighet* (accuracy)
- forstå tankegangen bak k -nærmeste nabo-klassifikasjon, valg av k
- kjenne til modellen for logistisk regresjon, og kunne tolke de estimerte koeffisientene
- forstå hvordan vi utfører klassifikasjon i Python
- kunne besvare problem 2 av prosjektoppgaven

Læringsressurser

Tema Klassifikasjon:

- **Kompendium:** Klassifikasjon (pdf og html, by Mette Langaas)
- **Korte videoer:** (by Mette Langaas)
 - Introduksjon og k -nærmeste nabo klassifiasjon (10:58)
 - Logistisk regresjon (14:17)
- Denne forelesningen
- **Disse slides** med notater

Klassifikasjon – hva er det?

- Mål:
 - tilordne en ny observasjon til en av flere *kjente* klasser
 - lage en klassifikasjonsregel
 - estimere sannsynligheten for at en ny observasjon tilhører de ulike klassene

For hver av de uavhengige observasjonene $i = 1, \dots, n$ har vi

- Forklарingsvariabler $(x_{1i}, x_{2i}, \dots, x_{pi})$
- En kategorisk responsvariablel y_i .

Eksempler I

- Hvilke tall er handskrevet på brev? Og hva er sannsynligheten for de ulike tallene (=klasser)?

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

- Kommer en gitt kunde til å betale tilbake lånet sitt?
- Prognose om noen blir syk (hjertesykdom, kreft...) og sannsynligheten for det.

Eksempler II

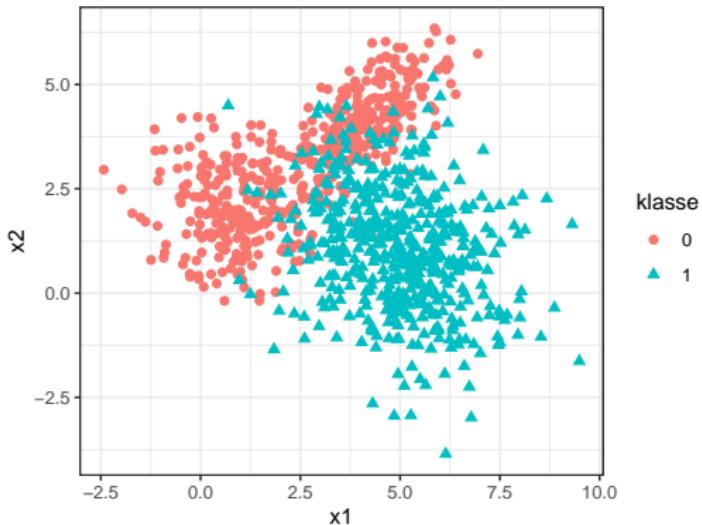
Problem 2 i prosjektet:

“Spam eller ham?” Spam filter: Klassifikasjon for å finne ut om en e-post er spam eller ikke (=“ham”).

Binær: Respons variable er “ja”/“nei” (1/0).

Syntetisk eksempel

- Et datasett med følgende struktur: (x_{1i}, x_{2i}, y_i) .

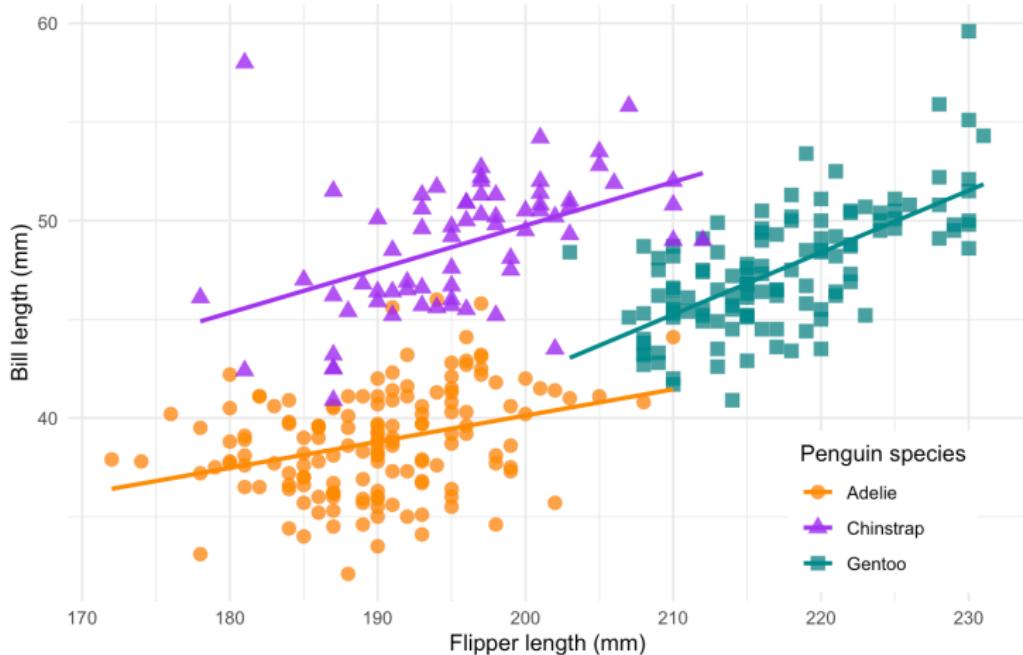


Spørsmål vi vil besvare: Hva er de beste klassifikasjonsgrensene?

Et reelt eksempel

Flipper and bill length

Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER



Data

Datasett: $(x_{1i}, x_{2i}, \dots, x_{pi}, y_i)$

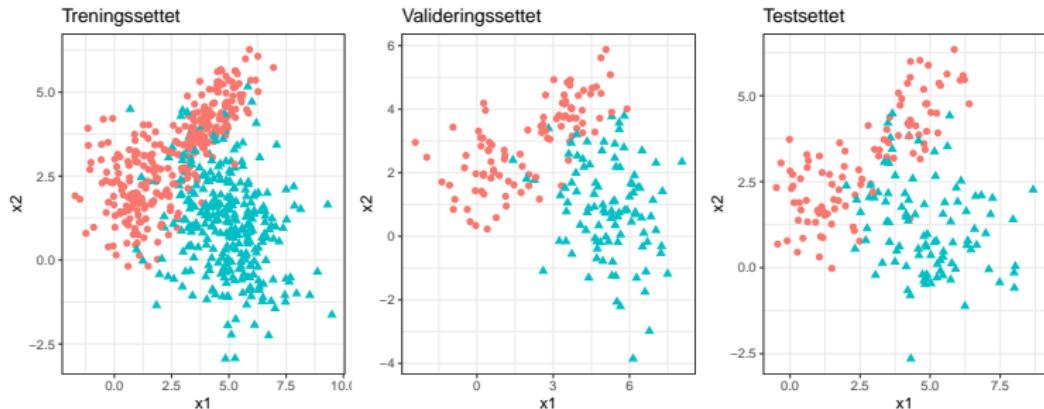
Vi må dele datasettet i tre deler:

- Treningsdata
- Valideringsdata
- Testdata



Hvorfor det?

Trening-, validering- og testsett for syntetiske data



k-nærmeste-nabo-klassifikasjon

For å *finne* klassifikasjonsreglen bruker vi bare treningsdataene.

Algoritmen:

- 1) Ny observasjon: $x_0 = (x_{1,0}, x_{2,0}, \dots, x_{p,0})$. Hvilken klasse bør denne klassifiseres til?
- 2) Finn de k nærmeste naboen til observasjonen i treningssettet.
- 3) Sannsynligheten for at den nye observasjonen tilhører klasse 1 anslår vi er andelen av de k nærmeste naboen som har tilhører klasse 1. Ditto for de andre klassene.
- 4) Klassen til den nye observasjonen er den som har størst sannsynlighet. Det blir det samme som å bruke flertallsavstemming.

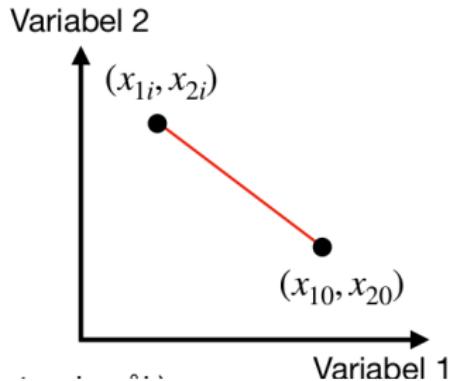
k-nærmeste-nabo-klassifikasjon

Tre spørsmål:

- Hva betyr “nærmest”? Vi trenger en definisjon av avstand.
- Hvilke verdier kan k ha?
- Hvordan bestemmer vi k ?

Hva betyr nærmest?

Nærmest er definert ved å bruke Euklidsk avstand.

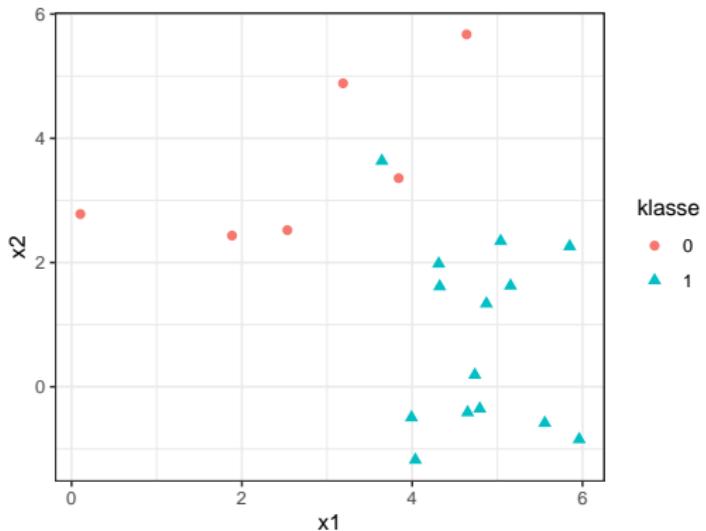


Euklidsk avstand:

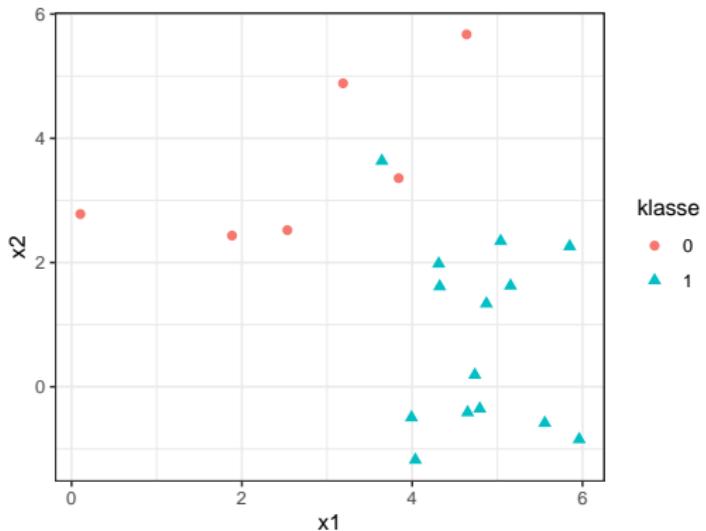
$$D_E(i, 0) = \sqrt{\sum_{j=1}^p (x_{ji} - x_{j0})^2}$$

Andre avstandsmål kan også brukes, men Euklidsk avstand er mest vanlig.

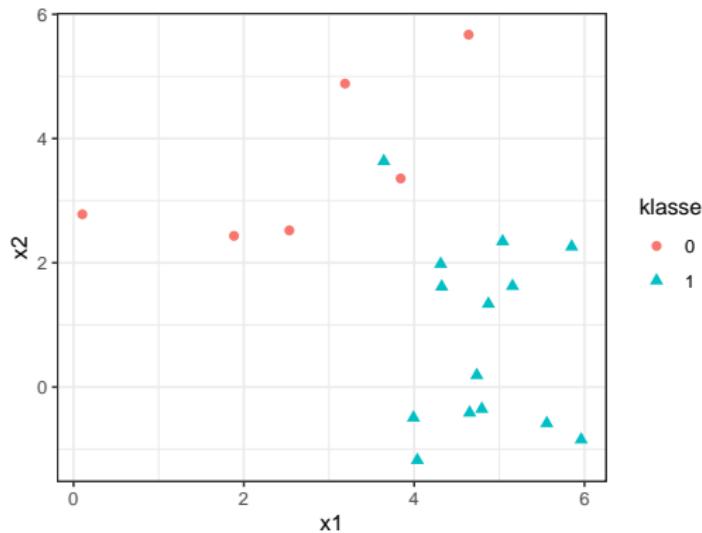
Nærmeste naboer



Nærmeste naboer

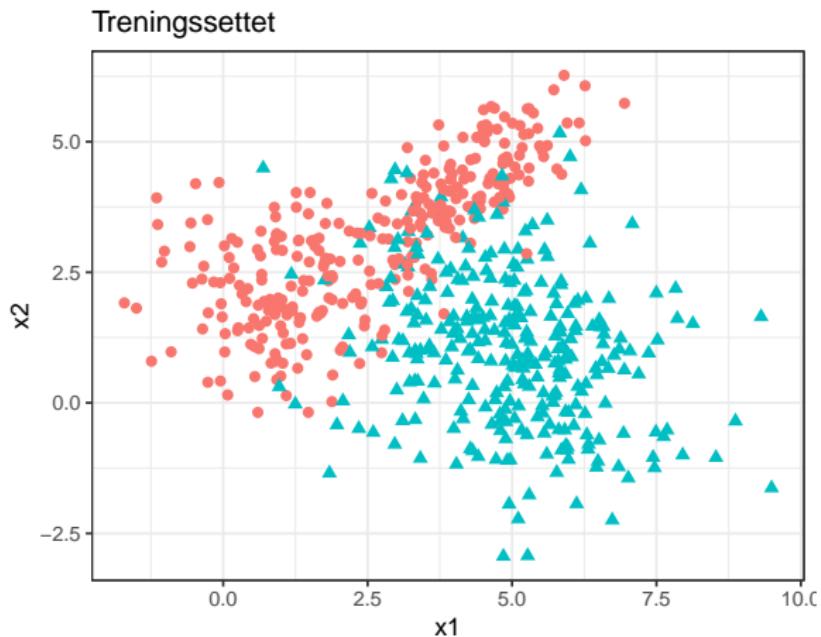


Tegn klassegrenser



Hvordan ser klassegrensene ut?

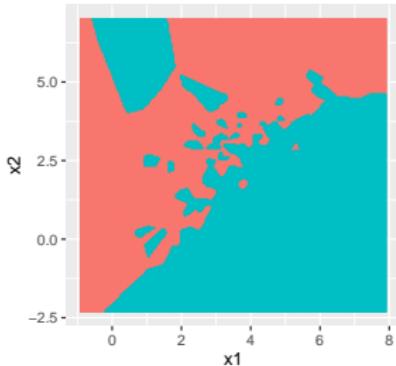
Husk: Vi bruker treningssettet for å finne klassifikasjonsreglen:



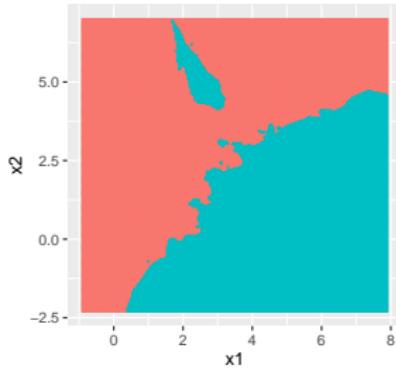
Hvordan ser klassegrensene ut?

Svar: It depends!

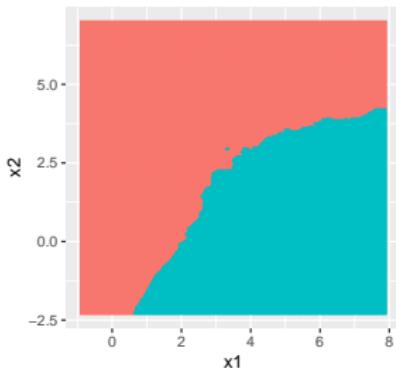
$k=1$



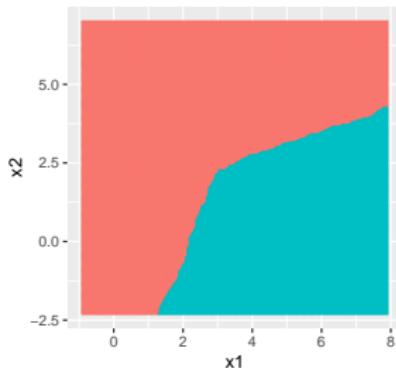
$k=5$



$k=15$



$k=99$



Men vent... hvordan velger vi k da?

Vi så jo at

- Hvis man velger k for lite, da blir grensen *for fleksibel*.

Men:

- Hvis man velger k for stor, kan grensen bli *for uflexibel*.

Det er noe som kalles en trade-off, og vi skal bruk valideringssettet for å få en idé om kvaliteten i klassifikasjonen.

Forvirringsmatrise

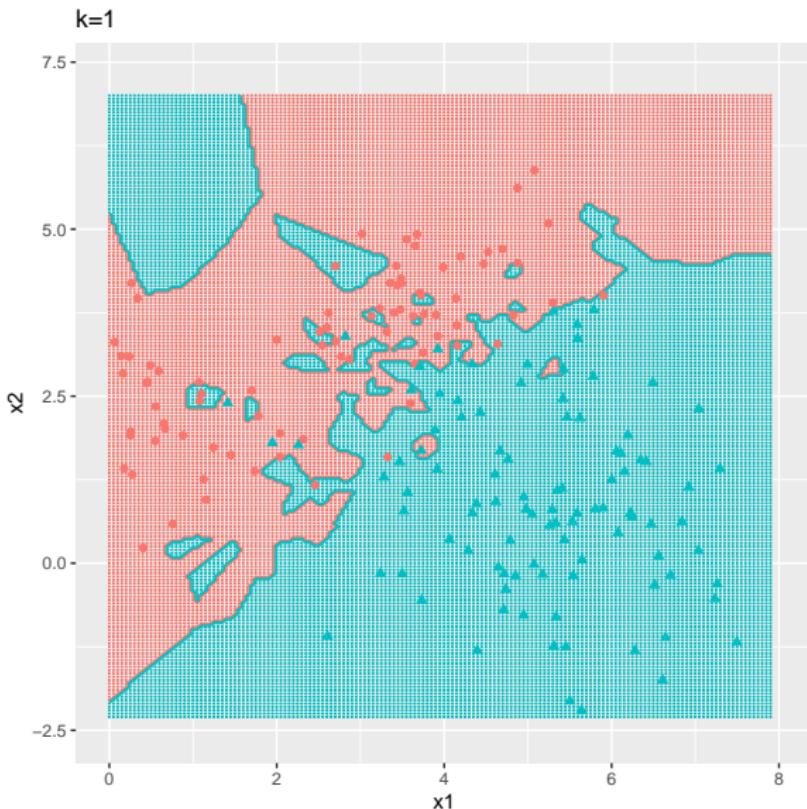
Forvirringsmatrise med to klasser:

		Sann klasse 0	Sann klasse 1
Predikert klasse 0	Sann negativ (TN)	Falsk negativ (FN)	
	Falsk positiv (FP)	Sann positiv (TP)	

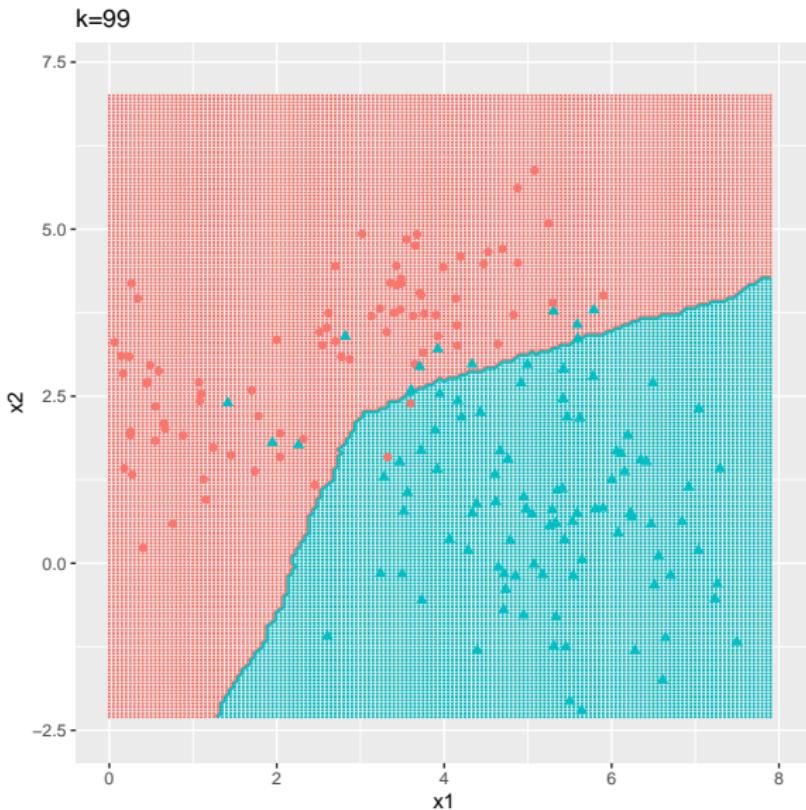
(Fra Mette Langaas)

- **Feilrate:** andel feilklassifiserte observasjoner.
- Derfor: Vi velger den k som minimerer feilraten på *valideringssettet* (ikke trainingssettet!).

Marker og tell antall gale klassifiseringer på valideringssettet ($k = 1$):

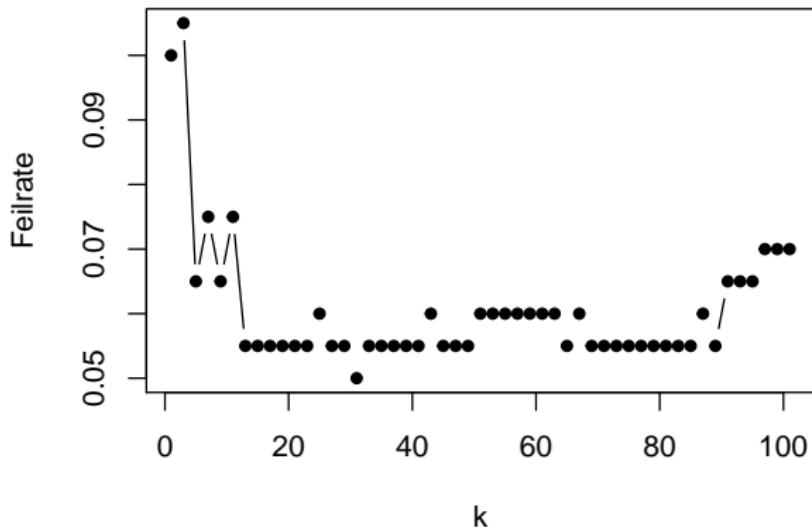


Igjen, men nå med $k = 99$:

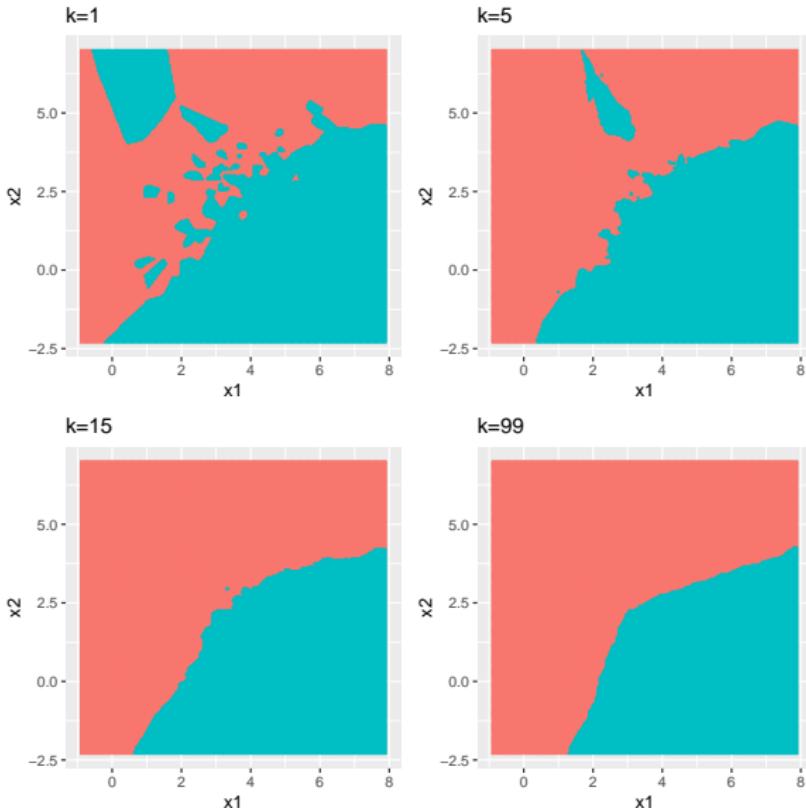


Feilrate i valideringssettet

Det kan vi nå systematisk gjøre med alle $k = 1, 3, \dots, k \leq n$:



Klassegrensene ser ganske likt ut for $k = 15$ og $k = 99$ ($k \geq 15$):



Data

Husk at vi hadde *tre* deler i datasettet:

- **Treningssett:** For å lage en klassifikasjonsregel
- **Valideringssett:** For å finne på optimale hyperparametere
- **Testsett:** For å evaluere regelen på fremtidige data

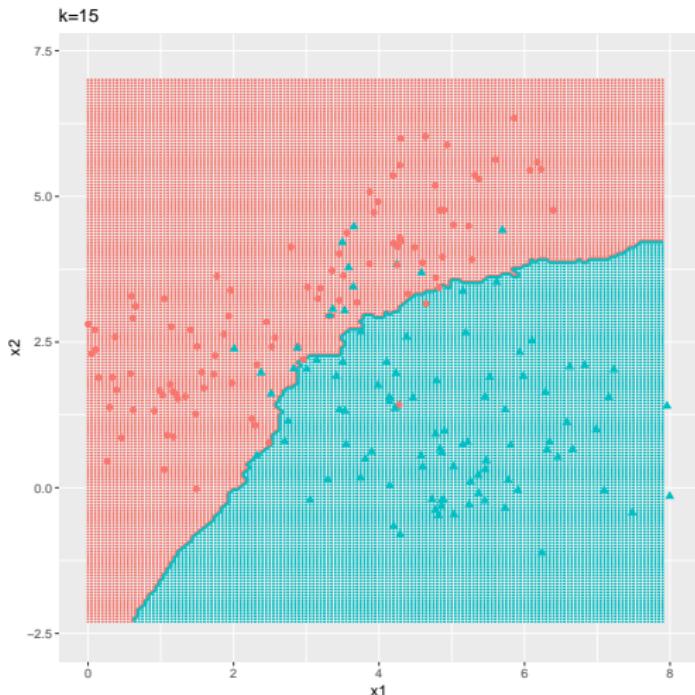
treningssett

valideringssett

testsett

Nå kan vi bruke testsettet for å *finne feilraten* for fremtidige data.

Feilraten på testsettet for KNN med $k = 15$ (egentlig er $k = 31$ best, men det gjør ikke en stor forskjell):



Feilraten er 0.09.

k-nærmeste nabo klassifikasjon i Python

```
knaboer = np.arange(1,199,step=2)
val_feilrate = np.empty(len(knaboer))

for i,k in enumerate(knaboer):
    knn = KNeighborsClassifier(n_neighbors=k,p=2)
    knn.fit(df_tren[['x1','x2']], df_tren['y'])
    val_feilrate[i] = 1-knn.score(df_val[['x1','x2']],
                                   df_val['y'])
```

Se prosjektoppgaven for mer Python kode.

Plan tema “Klassifikasjon”

- Læringsmål og ressurser
- Hva er klassifikasjon?
- Trening, validering og testing (3 datasett)
- k -nærmeste nabo (KNN): en intuitiv metode
- Forvirringsmatrise og feilrate for å evaluere metoden
- Logistisk regresjon

Logistisk regresjon - mål

- Forstå modellen
- Tolke estimerte koeffisienter
- Hypotesetest og p -verdi
- Velge mellom modeller
- Kunne utføre dette i Python

Logistisk regresjon

- Kan bare handtere *to kasser* $y_i \in \{0, 1\}$.
- Vi antar at Y_i har en **Bernoulli fordeling** med suksessannsynlighet p_i , derfor:

$$y_i = \begin{cases} 1 & \text{med sannsynlighet } p_i, \\ 0 & \text{med sannsynlighet } 1 - p_i. \end{cases}$$

- **Mål:** For forklaringsvariabler $(x_{1i}, x_{2i}, \dots, x_{pi})$ vi vil estimere $p_i = \Pr(y_i = 1 \mid x_1, \dots, x_p)$.

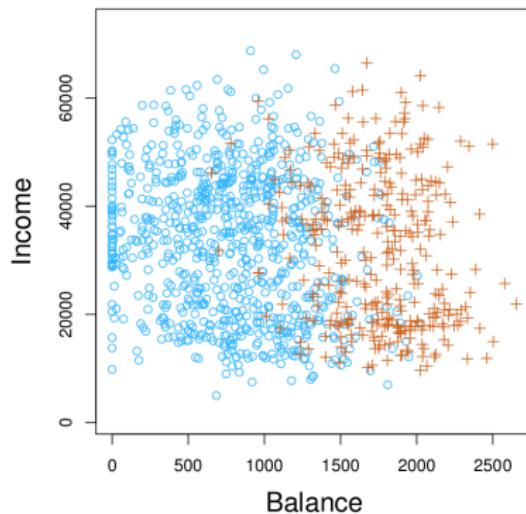
Eksempel: Kredittkort data

Datasettet **Default** er tatt fra her:

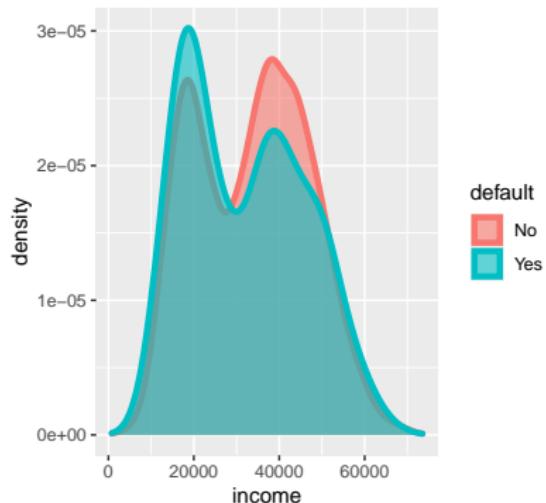
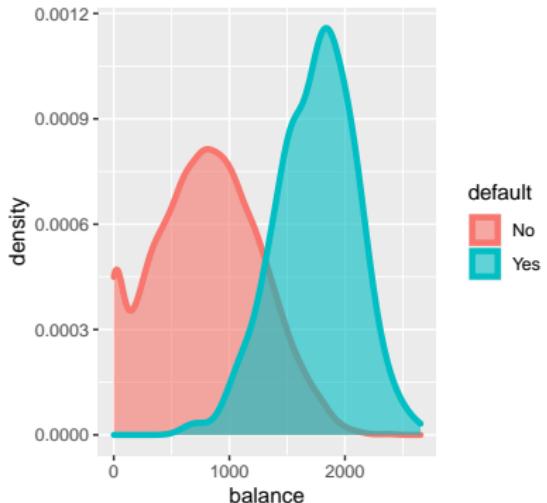
<https://rdrr.io/cran/ISLR/man/Credit.html>

Mål : forutsi om en person ikke betaler kredittkortregning (“person defaults”), avhengig av årsinntekten (income) og balansen på kredittkortet (balance).

Orange: default=yes, blue: default=no.



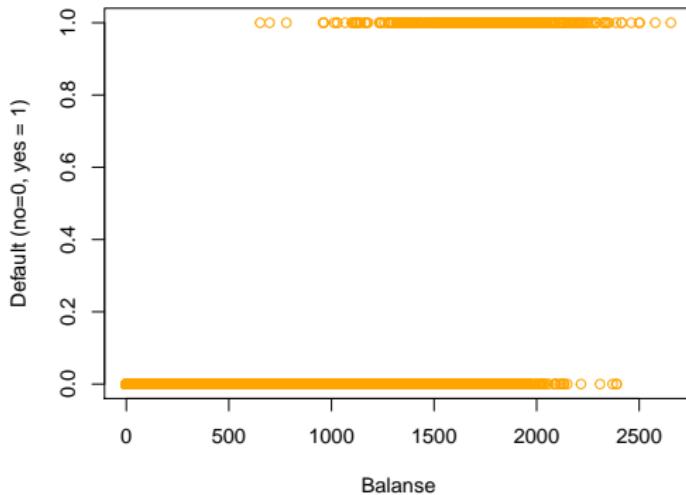
Det ser ut som at “Balance” er en ganske god forklarende variabel til Default (nei/ja).



Kan vi bare bruke linear regresjon for binær klassifikasjon?

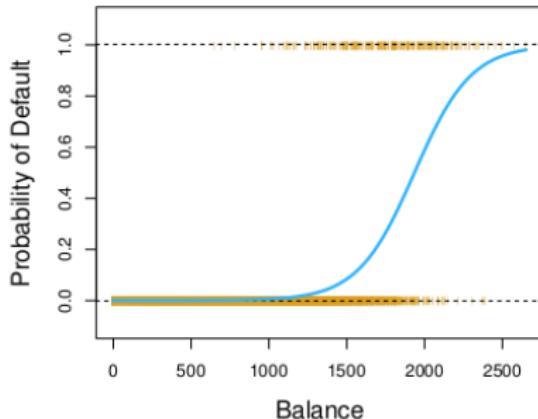
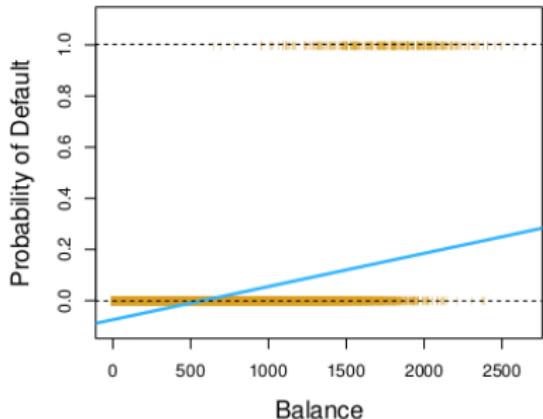
For en binær responsvariabel $Y = \text{yes}$ or no , og forklaringsvariabler X bruker vi vanligvis en *dummy encoding* :

$$Y = \begin{cases} 0 & \text{hvis no ,} \\ 1 & \text{hvis yes .} \end{cases}$$

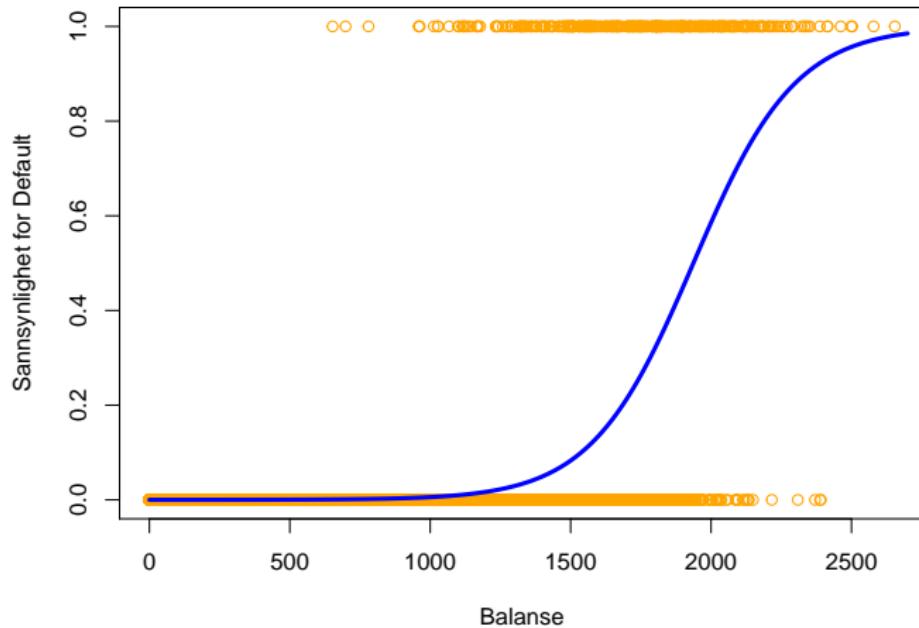


- Kunne vi da ikke bare formulere en linear regresjonsmodell for Y vs. X ?
- Vi kunne jo klassifisere responsen som “ja” (1) hvis $\hat{Y} > 0.5$.
- Problemet med linear regresjon: Vi kan forutsi $Y < 0$ eller $Y > 1$ med modellen, men en sannsynlighet er alltid mellom 0 og 1.

For kredittortdatasettet:



Vi trenger logistisk regresjon!



Enkel logistisk regresjon

- Vi antar at responsen Y_i er binomisk fordelt med suksessannsynlighet p_i
- **Ideen:** å koble p_i sammen med forklaringsvariablen med en logistisk funksjon:

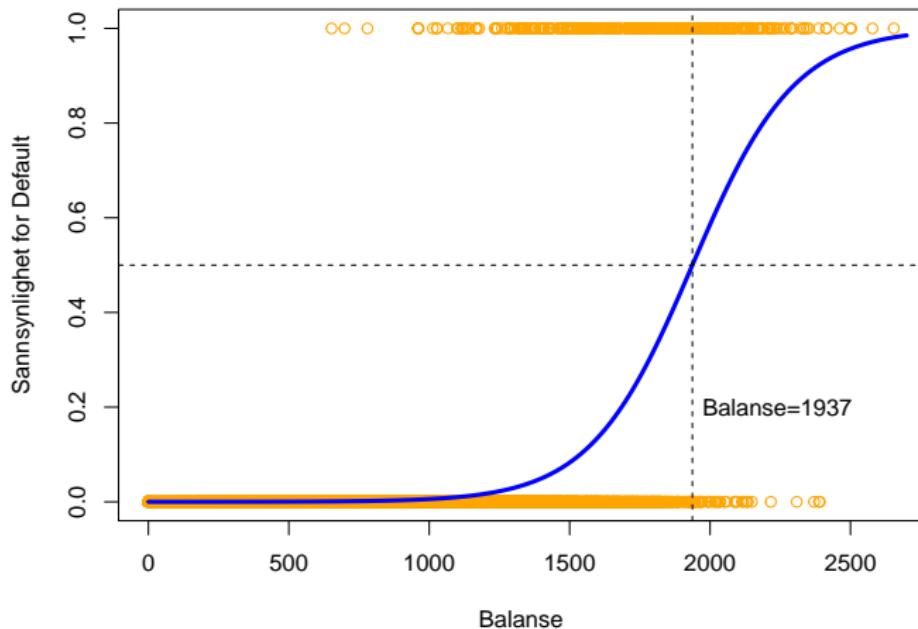
$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{1i})}{1 + \exp(\beta_0 + \beta_1 x_{1i})}.$$

- Denne verdien p_i er alltid mellom 0 og 1, som den skal være.

Klassifikasjonsregel

I kredittkorteksempelet er x_{1i} balansen til person i .

Klassifikasjonsregel: $p \geq 0.5$ vs $p < 0.5$.



Enkel logistisk regresjon med Python

Logit Regression Results						
Dep. Variable:	y	No. Observations:	6000			
Model:	Logit	Df Residuals:	5998			
Method:	MLE	Df Model:	1			
Date:	Fri, 03 Sep 2021	Pseudo R-squ.:	0.4726			
Time:	19:08:50	Log-Likelihood:	-460.65			
converged:	True	LL-Null:	-873.50			
Covariance Type:	nonrobust	LLR p-value:	1.403e-181			
	coef	std err	z	P> z	[0.025	0.975]
Intercept	-11.0385	0.488	-22.599	0.000	-11.996	-10.081
balance	0.0057	0.000	19.386	0.000	0.005	0.006

Tolkning av β_1 : odds

- Odds er definert som ratioen mellom sannsynlighet for suksess (p) og fiasko ($1 - p$):

$$odds = \frac{p}{1 - p}$$

- Noen eksempler (og zoom poll):
- I logistisk regresjon: Hvis vi øker verdien til kovariaten fra x_{i1} til $x_{i1} + 1$ så blir den nye oddsen lik den gamle multiplisert med e^{β_1} .

Enkel logistisk regresjon

	coef	std err	z	P> z	[0.025	0.975]
Intercept	-11.0385	0.488	-22.599	0.000	-11.996	-10.081
balance	0.0057	0.000	19.386	0.000	0.005	0.006

- $\beta_0 = -11.0385$, $\beta_1 = 0.0057$.
- $e^{0.0057} = 1.0057$: Hvis **balance** øker med 1, går oddsen opp en *faktor* 1.0057.
- $e^{0.0057 \cdot 100} = 1.768$: Hvis **balance** øker med 100, går oddsen opp en *faktor* 1.769.
- Husk at $H_0 : \beta_1 = 0$ versus $H_1 : \beta_1 \neq 0$. Her har vi $p < 0.001$ for testen, derfor forkaster vi H_0 .

Logistisk regresjon med flere forklaringsvariabler

	coef	std err	z	P> z	[0.025	0.975]
Intercept	-11.6549	0.574	-20.307	0.000	-12.780	-10.530
balance	0.0058	0.000	19.256	0.000	0.005	0.006
income	1.471e-05	6.51e-06	2.259	0.024	1.95e-06	2.75e-05

- Nytt:

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})}{1 + \exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i})}$$

- Tolkning av β_1 og β_2 ?
- p -verdi?

Hvilken modell er bedre?

- For klassifikasjonen, skal vi velge modellen med bare **balance** eller med **balance + income**?
- Ideen: Skjekk feilrate på valideringssettet!
- Husker du bruk av forvirringsmatrise?
- Det er resultatene fra modellene:

	pred:no	pred:yes
true:no	1923	10
true:yes	52	15

	pred:no	pred:yes
true:no	1924	9
true:yes	50	17

- Feilratene er derfor 0.031 (enkel modell) og 0.0295 (multippel modell)

Feilrate testsett

Til slutten kan vi sjekke feilraten på testsettet – dette settet har vi enda ikke brukt.

Her finner vi 0.0245

Python code

```
formel='y ~ balance + income'
modell = smf.logit(formel,data=df)
resultat = modell.fit()
resultat.summary()
print(np.exp(resultat.params))

cutoff = 0.5
test_pred = resultat.predict(exog = df_test)
y_testpred = np.where(test_pred > cutoff, 1, 0)
y_testobs = df_test['y']
print("Feilrate:", 1-accuracy_score(y_true=y_testobs,
y_pred=y_testpred,normalize=True))
```

Videre denne uken

- Se på videoene om Klyngeanalyse og K-gjennomsnitt (vent med videoen om hierarkisk klyngeanalyse).
- Les i kompendiet
- Begynn å jobbe med prosjektoppgavene
- Se her for mer informasjon:
<https://wiki.math.ntnu.no/istx1003/2021h/start>