STK-IN1050 - Statistikk for informatikere

22. Klassifikasjon, del III

Matematisk institutt

2026-05-06

Plan for forelesningen

- Evaluering: hvorfor ACC ikke er nok
- Forvirringsmatrise, sensitivitet, spesifisitet, PPV, NPV
- ROC-kurve og AUC
- Brier score
- Sammenlikning av modeller
- Eksempel med ubalanserte data

Plan for forelesningen

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import statsmodels.formula.api as smf
from sklearn.metrics import (
    confusion_matrix,
    roc_curve,
    roc_auc_score,
    brier_score_loss,
    accuracy_score
)
from sklearn.naive_bayes import GaussianNB
```

Data på hjertesykdom (fra siste forelesning)

```
HDdata = pd.read_csv("../data/Heart-disease-Cleveland.txt", sep=",")
HDdata.columns = ["age","sex","cp","trestbps","chol","fbs","restecg",
                  "thalach","exang","oldpeak","slope","ca","thal","num"]
HDdata["y"] = (HDdata["num"] > 0).astype(int)
HDdata = HDdata.drop(columns=["num"])
factor_cols = ["sex", "cp","restecg","exang","slope","thal"]
HDdata[factor_cols] = HDdata[factor_cols].astype("category")
```

```
HDdata.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	y
0	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	1
1	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
2	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
3	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
4	56.0	1.0	2.0	120.0	236.0	0.0	0.0	178.0	0.0	0.8	1.0	0.0	3.0	0

Trenings- og testsett

Vi bruker samme tilfeldige splitting som tidligere.

```
np.random.seed(42)
mask = np.random.rand(len(HDdata)) < 0.5   # ca. 50 % train, 50 % test

df_train = HDdata[mask].copy()
df_test  = HDdata[~mask].copy()

len(df_train), len(df_test)
```

```
(145, 157)
```

Logistisk regresjonmodell

Vi bruker modellen vi fant i forrige forelesning:

$$y \sim \texttt{sex} + \texttt{exang} + \texttt{thalach} + \texttt{oldpeak}$$

```
logit_model = smf.logit("y ~ sex + exang + thalach + oldpeak", data=df_train)
logit_result = logit_model.fit(disp=False)
logit_result.summary2().tables[1].round(3).iloc[:, :4]
```

	Coef.	Std.Err.	z	P>|z|
Intercept	3.826	1.835	2.085	0.037
sex[T.1.0]	1.039	0.493	2.106	0.035
exang[T.1.0]	1.418	0.470	3.017	0.003
thalach	-0.039	0.012	-3.404	0.001
oldpeak	0.777	0.234	3.320	0.001

Prediksjoner på testdata

Vi beregner predikerte sannsynligheter og klasser (med terskel 0.5):

```
p_test_logit = logit_result.predict(df_test)
y_test_true = df_test["y"].values
y_test_hat_05 = (p_test_logit >= 0.5).astype(int)

acc_test_logit = accuracy_score(y_test_true, y_test_hat_05)
round(acc_test_logit, 3)
```

```
0.771
```

Hvorfor ikke bare ACC?

Andel korrekt klassifisert er målt ved

$$\text{ACC} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i = y_i).$$

Dette målet har problemer ved ubalanserte klasser:

- Anta 90 % er friske ($Y = 0$) og 10 % er syke ($Y = 1$)
- Modell som alltid predikerer 0:
  - ACC = 90 %
  - Men sensitivitet = 0 (fanger ingen syke)

Vi trenger mer detaljerte mål.

Forvirringsmatrise (confusion matrix)

Vi definerer:

- Sann $Y = 1$ (syk), sann $Y = 0$ (frisk)
- Predikert $\hat{Y} = 1$ eller $\hat{Y} = 0$

	Sann $Y = 1$	Sann $Y = 0$
$\hat{Y} = 1$	TP	FP
$\hat{Y} = 0$	FN	TN

- TP: Sanne positive (syk som syk)
- TN: Sanne negative (frisk som frisk)
- FP: Falske positive (frisk som syk)
- FN: Falske negative (syk som frisk)

Indekser: Sensitivitet, spesifisitet, PPV, NPV

	Sann $Y = 1$	Sann $Y = 0$
$\hat{Y} = 1$	TP	FP
$\hat{Y} = 0$	FN	TN

Sensitivitet

$$= \frac{\text{TP}}{\text{TP} + \text{FN}} = \Pr(\hat{Y} = 1 \mid Y = 1)$$

Positiv prediktiv verdi (PPV)

$$= \frac{\text{TP}}{\text{TP} + \text{FP}} = \Pr(Y = 1 \mid \hat{Y} = 1)$$

Spesifisitet

$$= \frac{\text{TN}}{\text{TN} + \text{FP}} = \Pr(\hat{Y} = 0 \mid Y = 0)$$

Negativ prediktiv verdi (NPV)

$$= \frac{\text{TN}}{\text{TN} + \text{FN}} = \Pr(Y = 0 \mid \hat{Y} = 0)$$

Forvirringsmatrise for logistisk modell

```
cm_logit = confusion_matrix(y_test_true, y_test_hat_05)
cm_logit
```

```
array([[73, 14],
       [22, 48]])
```

Merk at `sklearn` gir matrisen i rekkefølgen:

$$\begin{pmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{pmatrix}$$

Beregning av indekser fra forvirringsmatrisen:

```
TN, FP, FN, TP = cm_logit.ravel()
sens = TP / (TP + FN)
spec = TN / (TN + FP)
ppv  = TP / (TP + FP)
npv  = TN / (TN + FN)
print(f"sens={sens:.3f}, spec={spec:.3f}, ppv={ppv:.3f}, npv={npv:.3f}")
```

```
sens=0.686, spec=0.839, ppv=0.774, npv=0.768
```

- **Sensitivitet**: andel syke som klassifiseres som syke
- **Spesifisitet**: andel friske som klassifiseres som friske
- **PPV**: andel klassifisert syke som er syke
- **NPV**: andel klassifisert friske som er friske

Terskelverdi

Vi har inntil nå brukt terskel 0.5:

$$\hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{hvis } \hat{p}(\mathbf{x}) \geq 0.5 \\ 0, & \text{ellers} \end{cases}$$

Men vi kan velge andre terskler:

- Lav terskel (f.eks. 0.3) $\Longrightarrow$ flere klassifiseres som syke:
  - høyere sensitivitet, lavere spesifisitet
- Høy terskel (f.eks. 0.7) $\Longrightarrow$ færre klassifiseres som syke:
  - lavere sensitivitet, høyere spesifisitet

Eksempel: flere terskler

```
def sens_spec_at_threshold(p, y_true, t):
    y_hat = (p >= t).astype(int)
    TN, FP, FN, TP = confusion_matrix(y_true, y_hat).ravel()
    sens = TP / (TP + FN)
    spec = TN / (TN + FP)
    return sens.round(3), spec.round(3)

thresholds = [0.3, 0.5, 0.7]
rows = []
for t in thresholds:
    sens_t, spec_t = sens_spec_at_threshold(p_test_logit, y_test_true, t)
    rows.append({"threshold": t, "sensitivitet": sens_t, "spesifisitet": spec_t})
pd.DataFrame(rows)
```

	threshold	sensitivitet	spesifisitet
0	0.3	0.771	0.621
1	0.5	0.686	0.839
2	0.7	0.586	0.931

Receiver operating characteristic (ROC)-kurve

For hver terskel t mellom 0 og 1:

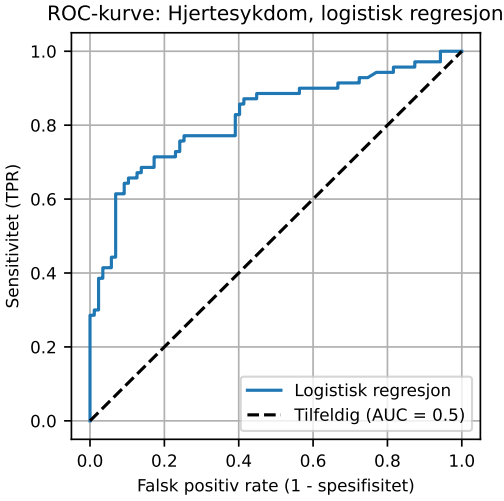
- **True Positive Rate (TPR)** = sensitivitet:

$$\text{TPR}(t) = \frac{\text{TP}(t)}{\text{TP}(t) + \text{FN}(t)}$$

- **False Positive Rate (FPR)** = 1 − spesifisitet:

$$\text{FPR}(t) = \frac{\text{FP}(t)}{\text{FP}(t) + \text{TN}(t)}$$

ROC-kurven er plottet av $(\text{FPR}(t), \text{TPR}(t))$ når t varierer.



AUC = areal under ROC-kurven

- Tolkning: Sannsynlighet for at en tilfeldig syk får høyere score enn en tilfeldig frisk
- Verdier:
  - 0.5 → tilfeldig
  - 1.0 → perfekt

```
auc_logit = roc_auc_score(y_test_true,
                          p_test_logit)
round(auc_logit, 3)
```

```
0.821
```

Brier score: kalibrering av sannsynligheter

Vi ønsker også å vurdere hvor gode sannsynlighetene er. Det kan vi gjøre med **Brier score** for binær respons:

$$\text{BS} = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - y_i)^2$$

- Brukes ofte til å rangere ulike modeller
- Lavere er bedre
- Straffer:
  - Feilklasser som er "sikre" (f.eks. $\hat{p}_i \approx 0.99$ for $y_i = 0$)
  - Dårlig kalibrerte sannsynligheter

```
round(brier_score_loss(y_test_true, p_test_logit), 3)
```

```
0.167
```

Gaussisk Naiv Bayes-modell

Vi bygger nå en Gaussisk Naive Bayes-modell med de samme prediktorene:

```
X_train = df_train[["sex", "exang", "thalach", "oldpeak"]].astype(float).values
y_train = df_train["y"].values

X_test = df_test[["sex", "exang", "thalach", "oldpeak"]].astype(float).values

gnb = GaussianNB()
gnb.fit(X_train, y_train)

p_test_gnb = gnb.predict_proba(X_test)[:, 1]  # sannsynlighet for klasse 1
y_test_hat_gnb = (p_test_gnb >= 0.5).astype(int)

acc_test_gnb = accuracy_score(y_test_true, y_test_hat_gnb)
round(acc_test_gnb, 3)
```

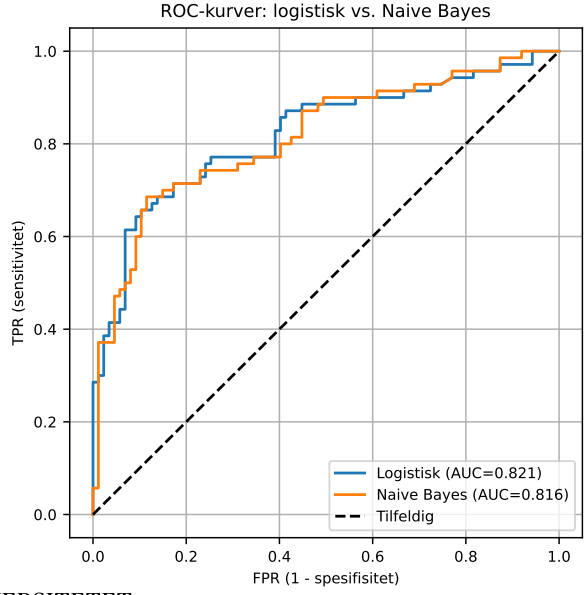
```
0.777
```

Sammenlikning: Gaussisk Naiv Bayes vs logistisk regresjon

	Modell	ACC	Sensitivitet	Spesifisitet	AUC	Brier
0	Logistisk regresjon	0.771	0.686	0.839	0.821	0.167
1	Naive Bayes	0.777	0.700	0.839	0.816	0.182

Høyere bedre: ACC, sensitivitet, spesifisitet, AUC

Lavere bedre: Brier score


ROC-kurver: logistisk vs. Naive Bayes
TPR (sensitivitet)
FPR (1 - spesifisitet)
0.0
0.2
0.4
0.6
0.8
1.0
Logistisk (AUC=0.821)
Naive Bayes (AUC=0.816)
Tilfeldig

Ubalanserte data – hvorfor ACC kan lure oss

Vi lager nå en kunstig ubalansert versjon av datasettet:

- Vi beholder alle friske ($y = 0$)
- Vi beholder bare 30% av de syke ($y = 1$)

```
df_pos = HDdata[HDdata["y"] == 1].copy()
df_neg = HDdata[HDdata["y"] == 0].copy()

np.random.seed(123)
df_pos_small = df_pos.sample(frac=0.3, random_state=123)
df_imbalanced = pd.concat([df_neg, df_pos_small], ignore_index=True)
df_imbalanced["y"].value_counts(normalize=True)

mask_imb = np.random.rand(len(df_imbalanced)) < 0.5
df_train_imb = df_imbalanced[mask_imb].copy()
df_test_imb  = df_imbalanced[~mask_imb].copy()
```

Logistisk modell på ubalansert data

	Modell	ACC	Sensitivitet	Spesifisitet	AUC	Brier
0	Logistisk regresjon	0.872	0.478	0.977	0.829	0.119
1	Naive Bayes	0.853	0.522	0.942	0.821	0.130

Høyere bedre: ACC, sensitivitet, spesifisitet, AUC

Lavere bedre: Brier score

Oppsummering

- For binære klassifikasjonsmodeller bør vi vurdere mer enn bare ACC:
  - Forvirringsmatrise $\rightarrow$ TP, TN, FP, FN
  - Sensitivitet, spesifisitet, PPV, NPV
- ROC-kurve og AUC:
  - Evaluering over alle terskler
  - Måler rangeringsevne
- Brier score:
  - Måler hvor gode sannsynlighetene er (kalibrering + presisjon)

Ubalanserte data: Krever særlig oppmerksomhet på sensitivitet og diverse mål