

Praktiske Ferdigheter

ELMED219: Momentliste F01–F10

ELMED219

Vår 2026

- 1 Jupyter Notebooks og Google Colab ([Jupyter](#), [Colab](#))
- 2 Python-grunnleggende ([NumPy](#), [Pandas](#), [Matplotlib](#))
- 3 Maskinlæring med scikit-learn ([scikit-learn](#))
- 4 Nettverksanalyse med NetworkX ([NetworkX](#))
- 5 Dyplæring med PyTorch ([PyTorch](#))
- 6 AI-verktøy og LaTeX ([ChatGPT](#), [Gemini](#), [Claude](#), [Overleaf](#))

F01: Kjøre Jupyter Notebooks i Google Colab

Hva er Jupyter Notebooks?

- Interaktive dokumenter som kombinerer kode, tekst og visualiseringer
- Kjør Python-kode celle for celle
- Standard i datavitenskapelig arbeid

Google Colab:

- Gratis sky-basert Jupyter-miljø fra Google
- Ingen installasjon – kjører i nettleseren
- Gratis GPU-tilgang (viktig for dyplæring!)
- Integrert med Google Drive

Kom i gang:

- 1 Gå til colab.research.google.com
- 2 Logg inn med Google-konto
- 3 "New notebook" eller åpne fra GitHub

Tips

Aktiver GPU: Runtime → Change runtime type → GPU

F02: Bruke Python-variabler, lister og enkle funksjoner

Variabler:

```
alder = 45           # int
navn = "Pasient_A"   # str
risiko = 0.73        # float
er_syk = True        # bool
```

Lister:

```
symptomer = ["hodepine", "kvalme", "tretthet"]
verdier = [1.2, 3.4, 5.6, 7.8]
symptomer.append("feber") # Legg til element
```

Funksjoner:

```
def beregn_bmi(vekt, hoyde):
    return vekt / (hoyde ** 2)

bmi = beregn_bmi(70, 1.75) # -> 22.9
```

F03: Importere og bruke biblioteker (numpy, pandas, matplotlib)

Import av biblioteker:

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

NumPy:

- Numeriske beregninger
- Arrays og matriser
- Matematiske funksjoner

```
arr = np.array([1, 2, 3])
mean = np.mean(arr)
```

Pandas:

- Datamanipulering
- DataFrames (tabeller)
- CSV, Excel I/O

```
df = pd.read_csv("data.csv")
df.head()
```

F04: Lese og inspisere datasett med pandas

Lese data:

```
df = pd.read_csv("pasienter.csv")
```

Inspisere data:

```
df.head()           # F r s t e 5 r a d e r
df.info()            # K o l o n n e t y p e r , n u l l v e r d i e r
df.describe()        # S t a t i s t i k k f o r n u m e r i s k e k o l o n n e r
df.shape             # ( a n t a l l r a d e r , a n t a l l k o l o n n e r )
df.columns           # K o l o n n e n a v n
```

Filtrering og utvalg:

```
# V e l g k o l o n n e r
df[["alder", "diagnose"]]

# F i l t r e r r a d e r
eldre = df[df["alder"] > 65]
```

F05: Trene en enkel modell med scikit-learn

Typisk ML-workflow:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# 1. Forbered data
X = df[["feature1", "feature2"]]
y = df["label"]

# 2. Del i trening og test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# 3. Lag og tren modell
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# 4. Evaluer
y_pred = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

F06: Visualisere resultater med matplotlib

Grunnleggende plotting:

```
import matplotlib.pyplot as plt

# Linjediagram
plt.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.xlabel("Tid"); plt.ylabel("Verdi")
plt.title("Min figur")
plt.show()
```

Histogram:

```
plt.hist(df["alder"], bins=20)
plt.xlabel("Alder")
plt.show()
```

Scatter plot:

```
plt.scatter(df["x"], df["y"])
plt.xlabel("X"); plt.ylabel("Y")
plt.show()
```

Seaborn

import seaborn as sns – Penere visualiseringer med enkel kode!

F07: Bruke NetworkX for enkel nettverksanalyse

Opprett og manipuler grafer:

```
import networkx as nx

# Opprett graf
G = nx.Graph()
G.add_nodes_from(["P1", "P2", "P3", "P4"])
G.add_edges_from([("P1", "P2"), ("P2", "P3"), ("P1", "P3")])

# Grunnleggende analyse
print(f"Antall_noder: {G.number_of_nodes()}")
print(f"Antall_kanter: {G.number_of_edges()}")

# Sentralitet
deg_cent = nx.degree_centrality(G)
print(f"Degree_centrality: {deg_cent}")

# Visualisering
nx.draw(G, with_labels=True, node_color="lightblue")
plt.show()
```

F08: Bygge og trene en modell med PyTorch

Enkel MLP i PyTorch:

```
import torch
import torch.nn as nn

class SimpleMLP(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out





model = SimpleMLP(784, 128, 10)  # MNIST-eksempel
```

Lab 2

Full trening med loss, optimizer, og treningsløkke gjennomgås i Lab 2.

F09: Bruke AI-verktøy (ChatGPT, Gemini, Claude) som kodehjelp

Nyttige bruksområder:

-  **Forklare kode:** “Hva gjør denne funksjonen?”
-  **Feilsøking:** “Hvorfor får jeg feilmeldingen?”
-  **Forslag:** “Gjør dette mer effektivt?”
-  **Generere kode:** “Skriv en funksjon...”

Tips for effektiv bruk:

- 1 Vær **spesifikk** i spørsmålene
- 2 Inkluder **kontekst**
- 3 **Verifiser** alltid AI-kode
- 4 Bruk som **læringsverktøy**

AI-drevne IDE-er (Integrated Development Environment)

IDE = Editor + kompilator + debugger + verktøy i ett. **AI-drevet IDE** = IDE med LLM-agenter som kan planlegge, skrive, teste og validere kode autonomt.

Betydning for medisinsk/biomedisinsk forskning: Senker terskelen for å utvikle analyseverktøy, akselererer prototyping av ML-modeller, og muliggjør raskere oversettelse fra forskning til klinikk.

Cursor, Windsurf	Editor med agenter, plan mode
Google Antigravity	Agent-first, multi-modell (Gemini 3)
Replit Agent	Sky-basert IDE med AI-agent
Lovable	Naturlig språk → full-stack app
GitHub Copilot, JetBrains AI	AI-assistenter i eksisterende IDE-er

F10: Skrive dokumenter med LaTeX/Overleaf

BibTeX (.bib-fil):

```
@article{smith2024,  
  author = {Smith, J.},  
  title = {AI in Medicine},  
  journal = {Nature Med.},  
  year = {2024},  
  volume = {30},  
  pages = {123--130},  
  doi = {10.1038/s41591-  
    024-01234-5}  
}  
  
@book{bishop2006,  
  author = {Bishop, C.},  
  title = {Pattern Recog.},  
  publisher = {Springer},  
  year = {2006},  
  isbn = {978-0387310732}  
}
```

I tekst: `\cite{smith2024}`

IMRAD-struktur:

```
\documentclass{article}  
\usepackage{graphicx,booktabs}  
\begin{document}  
\title{Tittel}  
\author{Forfatter}  
\maketitle  
\begin{abstract} ... \end{abstract}  
% Innledning  
\section{Introduction}  
Bakgrunn og mål \cite{smith2024}.  
% Materiale og metoder  
\section{Methods}  
Datainnsamling og analyse.  
% Resultater  
\section{Results}  
\begin{table}[h]  
\begin{tabular}{lcc} \toprule  
Modell & AUC & F1 \\ \midrule  
CNN & 0.92 & 0.88 \\ \bottomrule  
\end{tabular}  
\caption{Resultater}\label{tab:1}  
\end{table}  
% Diskusjon  
\section{Discussion}  
Implikasjoner og begrensninger.  
\bibliography{refs}  
\end{document}
```

Tidsskrift m/LaTeX:

- PLOS Medicine
- Nature / Nat. Med.
- Bioinformatics
- The Lancet
- BMJ
- Elsevier (mange)
- Springer Nature
- MDPI (open access)
- IEEE JBHI
- JMLR
- Frontiers

Overleaf Gallery

Oppsummering: Praktiske ferdigheter

Jupyter og Python:

- F01–F04: Colab, variabler, lister, funksjoner, pandas

Maskinlæring:

- F05–F06: scikit-learn workflow, matplotlib visualisering

Spesialisert:

- F07: NetworkX for nettverksanalyse
- F08: PyTorch for dyplæring

Verktøy:

- F09: AI-assistenter som kodehjelp
- F10: LaTeX/Overleaf for akademisk skriving

Lynkurs og Labs

Alle disse ferdighetene praktiseres gjennom Lynkurset og Lab 0–3. Øving gir mestring!