# Intro

* Dagens program / formål
* I holder selv pauser efter behov, vi holder opsamling kl. 11 (ca.)

# Opgave 1 Lidt om arrays, tensorer, slicing og sådan noget…

**Arbejd sammen i par (eller 3 personer, hvis det ikke går op…)**

Forklar begreberne

* + Skalar
    - Et tal du ganger på fx en matrice eller vektor
  + Vektor
    - Det **er** et ét-dimensionelt array. I og med at det rent matematisk set er en pil i et n-dimensionelt rum.
  + Matrice
    - N-dimensionel vektor (basically)
  + NumPy array
    - I stedet for et normalt array i python, har det en masse helper funktioner (slicing, etc, blabla)

Løs følgende opgaver med papir og blyant (brug ***ikke*** computer!)

**Antag** at følgende tensor er defineret: numbers : [10, 45, 20, 67, 29, 12, 15, 99, 40, 23]

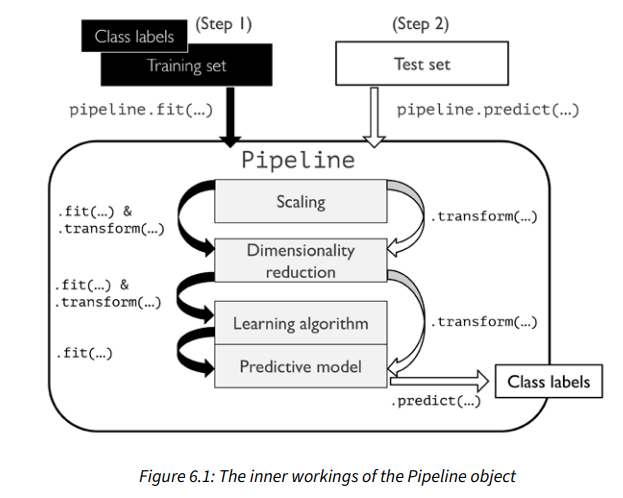
Løs følgende opgaver ved at finde et Python udtryk der giver korrekt svar (se løsning til 1. opg. for eksempel):

1. first\_number = numbers[0]
2. last\_number = numbers[-1]
3. before\_last\_number = numbers[-2]
4. seventh\_number = numbers[6]
5. first\_five\_numbers = [:5]
6. last\_three\_numbers = [-3:]
7. third\_to\_sixth\_numbers = [2:7]
8. reversed\_numbers = [::-1] Du tar hele arrayet, og din stepsize er -1, således at du vender den om.

# Opgave 2 Refleksionsspørgsmål

1. Hvad kan/skal man putte i en *pipeline* og hvad er formålet med en pipeline?

Transformers og estimators. Du slipper for dobbeltarbejde, i og med at du alligevel ville skulle standardize, regularize & fit på træning-, validering- og test datasættene.



1. Hvilke metoder bliver kaldt på en pipelines komponenter under *training*?

Fit og Transform.

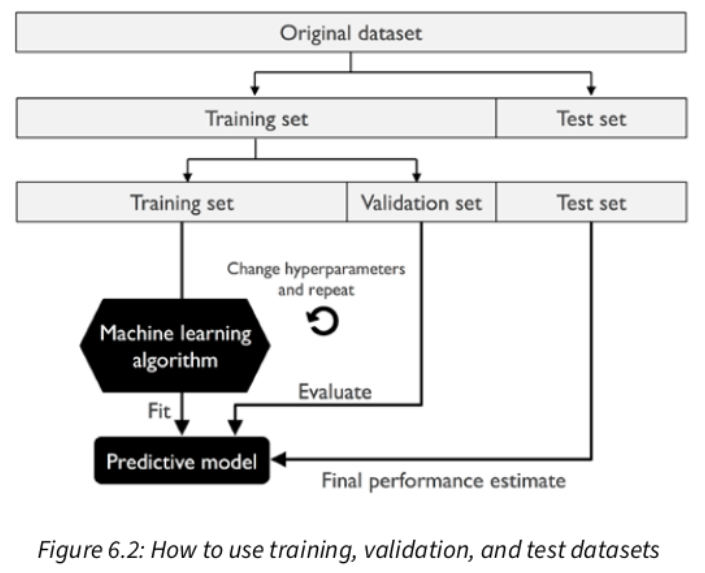
Når du kalder .fit() metoden på pipelinen, kalder du både fit() og transform() på de forskellige transformere, den data der kommer ud af hver transformer bliver givet videre til den næste, indtil den kommer til estimatoren.

1. Hvilke metoder bliver kaldt på en pipelines komponenter under *prediction*?

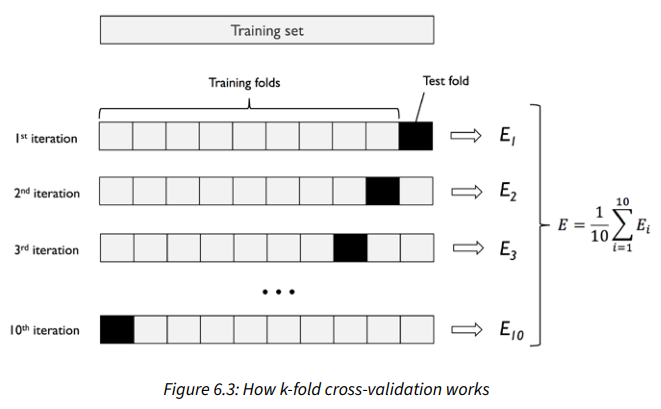
Transform og Predict.

1. Hvad er fordelen ved både at have et *test* og *validation* dataset?

I kontekst af metoder til at finde ens model performance (finde hvilke hyperparemeters der er væsentlige at skrue på for **dette** datasæt) bruger man typisk et “validation” sæt til at anskue hvilke parametre det skal være. Derfor er de i princippet kommet med under “fittingen”, og er ikke gode for at finde modellen **egentlige** nøjagtighed.



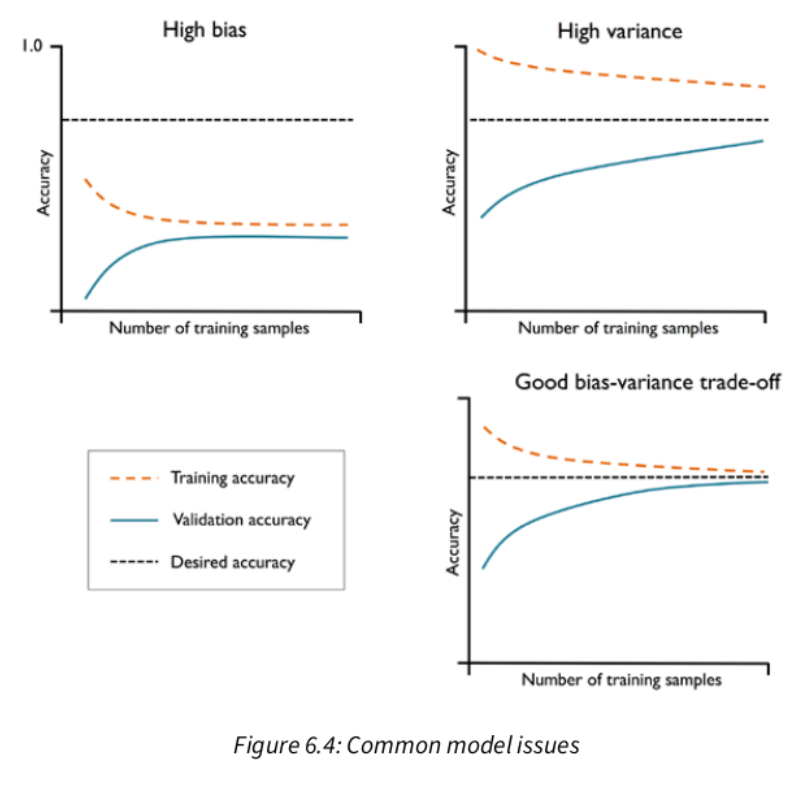
1. Beskriv *k-fold cross-validation* kort. Benytter den et validation dataset?



Nej. Fremgangsmåden er helt anderledes i forhold til holdout. Pointen her er at du får faktisk trænet med **alle** data, i kontrast til når du splitter dataset op i træning-, validering og test.

1. Hvad er *learning + validation curves* og hvad benyttes de til?

Learning Curve (Venstre er underfitted, højre er overfitted):



Få et visuelt indblik i om din model muligvis lider af underfitting eller overfitting. ELLER BARE ER PERFEKT?!

1. Hvad er *bias* og *variance* og hvilke problemer hører sammen med dem?

Bias refererer til fejlen der bliver introduceret i at approximere et reelt problem med en for simpel model. Variance refererer til fejlen der bliver introduceret ved at modellere støj i træningsdataen, i stedet for underliggende signal. (Svar fra CHATGPT)

Hvis du har high bias, lider din model af underfitting.

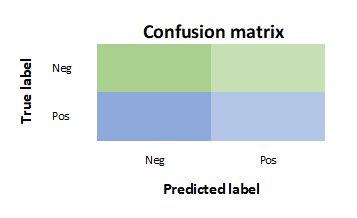
Hvis du har high variance så lider din model af overfitting.

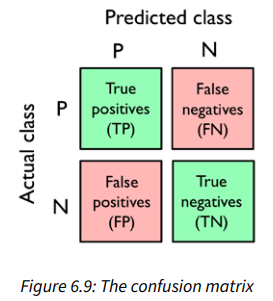
1. Hvad er *grid-search*?

En fyldestgørende / omfattende søgning af ens givne hyperparameters og deres værdi-rammer. Med det formål for at finde ud af hvilken kombination af hyperparameters der giver bedst mening at skrue på til hvilke værdier.

1. Hvad er *performance evaluation metrics*?

Det er forskellige målinger på din model. Alle performance metrics bearbejder på en eller anden måde de fire værdier der er i spil i en confusion matrix. Få at navne et par metrics: Precision, Recall, F1 og Matthews Correlation Coefficient (MCC).

1. Udfyld følgende *confusion matrix* med betegnelserne TP, FP, FN, TN (*true positive, false positive, false negative* og *true negative*) i følgende figur:



1. Hvis ovenstående tabel angår at afgøre retssager hvor man hellere vil lade en skyldig slippe fejlagtigt (false negative) end at dømme en uskyldig (false positive), for hvilken metric vil du så optimere?

Vi vil gerne optimere efter at have FPR så tæt på 0 som mulig.

Hvis det derimod angår hvilke patienter du vil indkalde til nærmere undersøgelse og du hellere vil indkalde en rask end at misse en mulig syg patient, for hvilken metric vil du så optimere for?

Raske personer (positive) Syge personer (negative)

Rask person der bliver indkaldt (false negative)

Syg person der **ikke** bliver indkaldt (false positive)

Ift ovenstående vil vi optimere efter at have FPR så tæt på 0 som muligt.

I sidste ende er det hvordan vi klassificere de forskellige klasser, og vi som brugere vægter højere ift confusion matricet.

1. Beregn ***ERR***, ***ACC***, ***FPR*,** ***TPR (REC)*, *PRE*** og ***F1*** for følgende confusion matrix:Et billede, der indeholder tekst, himmel, orange, farverig

   Automatisk genereret beskrivelse

Vi tager udgangspunkt i klasserne A og B:

| **TP** | **FP** |
| --- | --- |
| **FN** | **TN** |

ERR = (FP + FN) / (FP + FN + TP + TN) = 0.047619048

ACC = (TP + TN) / (FP + FN + TP + TN) = 1 - ERR = 0.952380952

FPR = FP / (FP + TN) = 0.060606061

TPR (Recall (REC)) = TP / (FN + TP) = 0.966666667

PRE = TP / (TP + FP) = 0.935483871

F1 = 0.950819672

# Opgave 3 Hands on pipelines og validering

Benyt dit dataset fra uge 7 (vedr. færgedata) og byg en pipeline som kører (!). Eksperimenter løs – fx med forskellige *transformers* og/eller *estimators*.

Benyt *cross validation* og udarbejd *training* samt *validation* kurver.

Prøv med et nyt dataset fra *Kaggle*.

# Opgave 4 … lidt mere om tensorer

**Antag** nu at følgende NumPy array er defineret: numbers = np.array ([

[0, 1, 2],

[10, 11, 12],

[20, 21, 22],

[30, 31, 32],

[40, 41, 42] ])

* # Find count of rows and colums:  
   rows\_count, columns\_count = ?, ?
* # Get the first element of each row and save it into array with shape (5,):  
   first\_elems\_A = ?
* # Get the first element of each row and save it as 2-dimensional array (with shape (5, 1)):  
   first\_elems\_B = ?
* # Get the second and third elements from the fourth and fifth rows:  
   some\_elems = ?
* # Divide each number by 2:  
   numbers\_divided\_by\_two = ?
* # Add 5 to each number:  
   numbers\_plus\_five = ?
* # Add this array: [[5, 6, 7], [15, 16, 17], [25, 26, 27], [35, 36, 37], [45, 46, 47]] to the "numbers" array:  
   new\_numbers = ?

# Opsamling / Outro (5 minutter)

* Var der noget vi ikke nåede eller ikke fik behandlet tilstrækkeligt?
* Var det svært / nemt?
* Lærte du tilstrækkeligt i dag?   
  Hvis nej: Hvad var årsagen?
* Andet og næste gang…