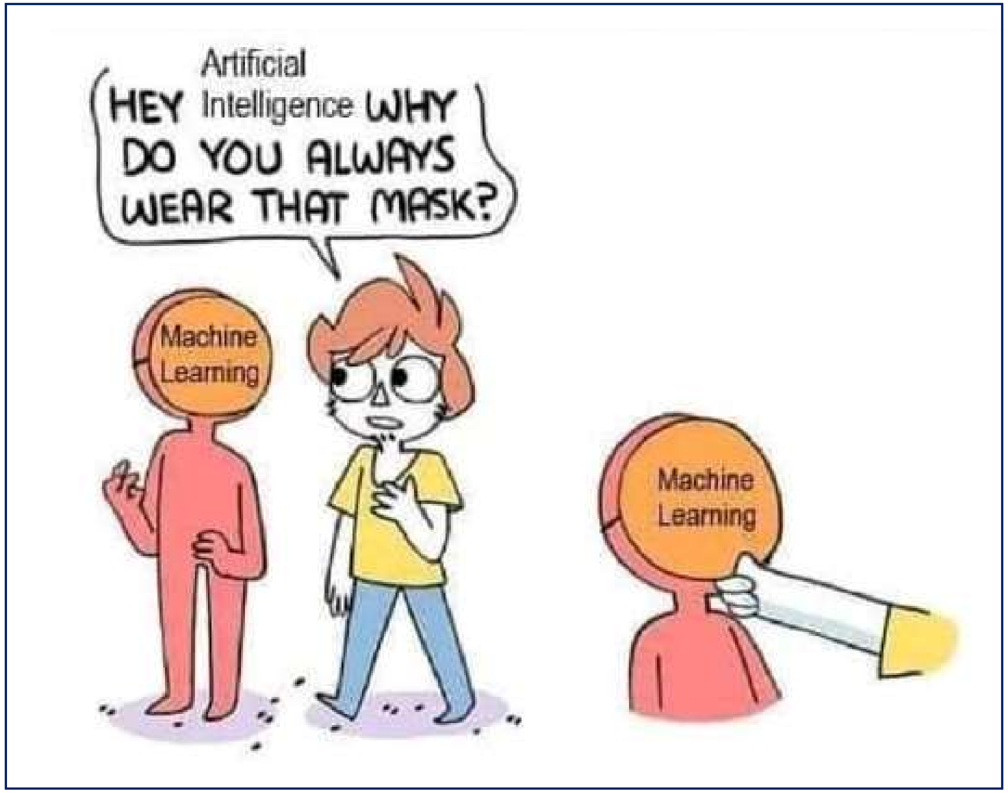
TALL SOM FORTELLER

# An introduction to (supervised) machine learning

6. SEPT 2024

BORISKA TOTH

Statistisk sentralbyrå
Statistics Norway

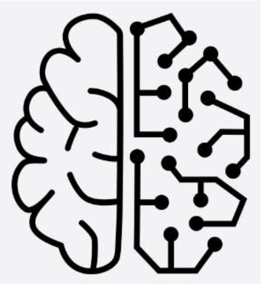




Statistisk sentralbyrå
Statistics Norway

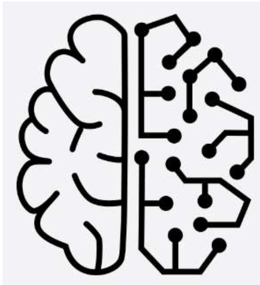
Source: Elon Musk på Twitter/X.

# This course- machine learning



- Machine learning: use algorithms to automatically learn from data (learn insights, recognize patterns)
- AI: (broader) developing computers and robots that mimic and exceed human cognitive abilities
- Machine learning is the backbone of contemporary AI

# This course- maching learning



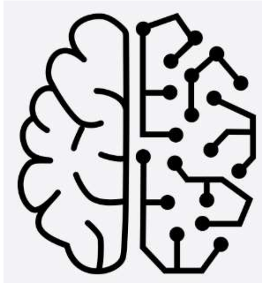
- A motivating example
- Give definitions: supervised learning, model, fitting, performance
- Development of a machine learning system
- Training data, and making features

---short break---

- Algorithms
- Testing, predicting
- Diagnosing problems and improving the setup

Statistisk sentralbyrå
Statistics Norway

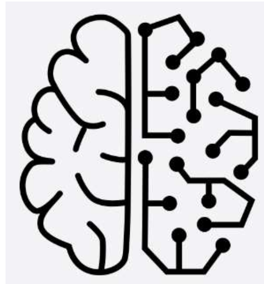
# Supervised learning



- Basic concept of supervised learning: learn a function f(X) that makes good predictions for an outcome Y, given a set of examples.
- VERSUS other types of learning:
  - Unsupervised learning
  - Semisupervised learning
  - Etc.
- Also: generative vs discriminative

| Item | COICOP |
|---|---|
| TINE milk 1% | 01141 |
| Macbook Pro 13” | 08131 |
| Travel insurance DNB | 12141 |

# Machine learning at SSB



- Big push to incorporate machine learning
- Classification (categorical): Replacing human coding
  - CPI, household budget survey COICOP
  - Time use survey ACL
  - Occupations for labor force survey
- Regression (continuous variables): editing, prediction
  - Editing outliers
  - Economics and price statistics

Statistisk sentralbyrå
Statistics Norway

# An example

- Sentiment analysis- predict the sentiment of news stories, customer reviews, social media posts, ...
- Sentiment analysis at SSB (ficitious): predict how happy people are on a scale of -10 to 10, based on many variables:

background (job title, age, section), time of year, participation in social and work events, tracking whereabouts in the building, text of recent email and Teams messages, ...

- Many variables: *10_happy_hour, 10_price_statistics, lunch_break_duration, clock_out_time, uses_of_word_challenge, years_at_SSB, number_messages, ...*

- All 830 employees give data: each person gets 12 random times of the year to submit a data point

| X | | | | | Y |
|---|---|---|---|---|---|
| 10_word_ publishing | age | 10_wellness | num_coffees | … | |
| 0 | 64 | 0 | 4 | … | 9 |
| 0 | 24 | 1 | 11 | … | 1 |
| … | … | … | … | … | … |

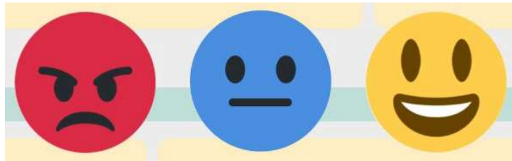
10 x 830 points: **training set**, come up with **$f(X)=Y_{pred}$**

| X | | | | | Y | | $Y_{pred}$ |
|---|---|---|---|---|---|---|---|
| 10_word_ publishing | age | 10_wellness | num_coffees | … | | | |
| 1 | 38 | 0 | 6 | … | -5 | =? | -4.3 |
| 0 | 55 | 1 | 0 | … | 6 | | 7.9 |
| … | … | … | … | … | … | | … |

2 x 830 points: **test set**, check **($Y_{pred}$ –Y)**

**Goal: minimize the mean absolute error over the test set, $1/n \sum |Y_{pred}-Y|$**

- Example- **sentiment analysis** at SSB



## Attempt 1: rule-based

$$Y_{pred} = \begin{cases} 9 \text{ if } & (\textit{num\_emails\_word\_exciting}>10 \ \& \ \textit{lunch\_cafeteria}>3 \ \& \ \textit{10\_running\_club} \ \& \ldots) \\ 1 \text{ if } & \ldots \\ \quad \ldots & \end{cases}$$

- 1000’s of variables

## Attempt 2: k-nearest neighbors (k=4)

- Pick the 4 employees in the training set that are «closest» (vector distance) to input X, and output their average sentiment score
- But: cannot compute a function f(X), all we have is a form of imputation
- New employee with higher *num_social_events* and *num_professional* than any previously seen

- Example- **sentiment analysis** at SSB

## Attempt 1: rule-based

$$Y_{pred} = \begin{cases} 9 \text{ if } (\textit{num\_emails\_word\_exciting}>10 \ \& \ \textit{lunch\_cafeteria}>3 \ \& \ \textit{10\_running\_club} \ \&\ldots) \\ 1 \text{ if } \ldots \\ \ldots \end{cases}$$

**MAE: 4.8**

- 1000's of variables

**A machine learning algorithm**

## Attempt 2: k-nearest neighbors (k=4)

**MAE: 3.1**

- Pick the 4 employees in the training set that are «closest» (vector distance) to input X, and output their average sentiment score
- But: cannot compute a function f(X), all we have is a form of imputation
- New employee with higher *num_social_events* and *num_professional* than any previously seen

- Example- ***sentiment analysis*** at SSB

## Attempt 3: linear regression

$$Y_{pred} = .2 * num\_seminars + .03 * years\_at\_SSB + \cdots$$

- Minimize least-squares error, **min $\Sigma(\beta^T X - Y)^2$**
- Very standard, easy to interpret, loads of statistical theory
- But: linear model (interactions, non-linear..)

## Attempt 4: polynomial regression

$$Y_{pred} = -1.2 * 01\_word\_challenge * 01\_word\_printer + pay\_raise^2 + \cdots$$

- Quadratic function still limited, for example *01_holiday_party*01_team_at_party*01_good_budget_wine*01_karaoke*
- More flexible model: blow up in number of coefficients, most of which aren't useful

- Example- ***sentiment analysis*** at SSB

## Attempt 3: linear regression

$$Y_{pred} = .2 * num\_seminars + .03 * years\_at\_SSB + \cdots$$

**MAE: 3.5**

- Minimize least-squares error, **min $\Sigma(\beta^T X - Y)^2$**
- Very standard, easy to interpret, loads of statistical theory
- But: linear model (interactions, non-linear..)

**Coefficients β are parameters of a machine learning model.**

## Attempt 4: polynomial regression

$$Y_{pred} = -1.2 * 01\_word\_challenge * 01\_word\_printer + pay\_raise^2 + \cdots$$

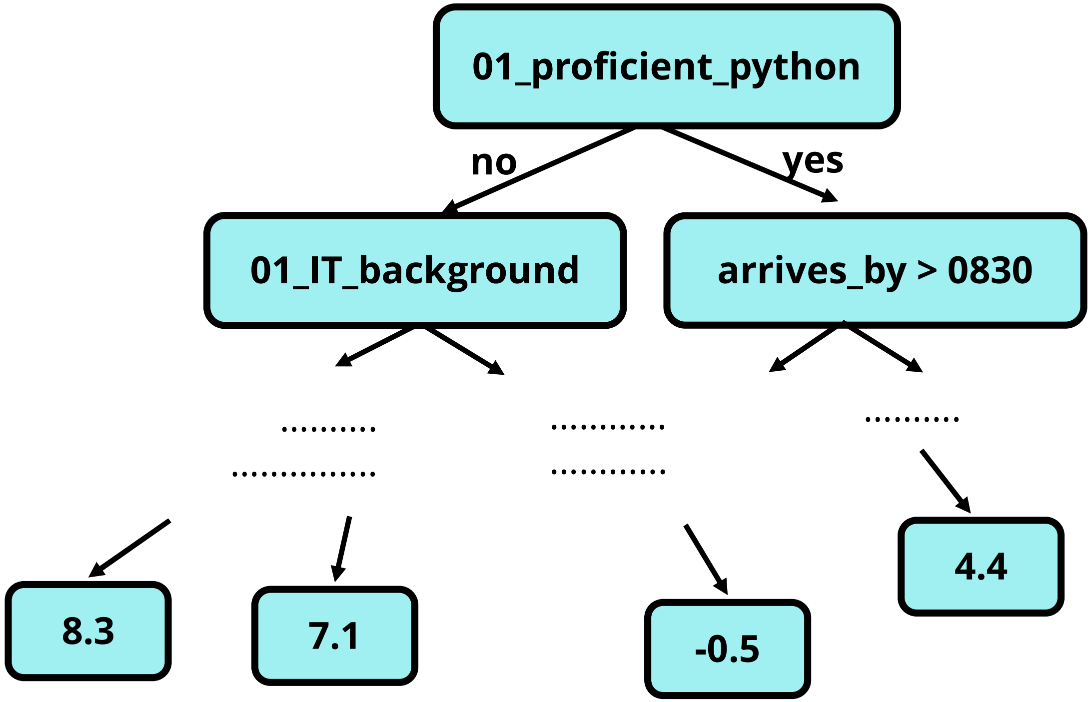
**MAE: 2.9**

- Quadratic function still limited, for example *01_holiday_party*01_team_at_party*01_good_budget_wine*01_karaoke*
- More flexible model: blow up in number of coefficients, most of which aren't useful

- Example- *sentiment analysis* at SSB

## Attempt 5: small decision tree (levels=20)

- Non-parametric **f(X)**, interaction between variables
- Higher splits correspond to more important predictive factors
- Automatically learns the most important predictors and ignores the rest

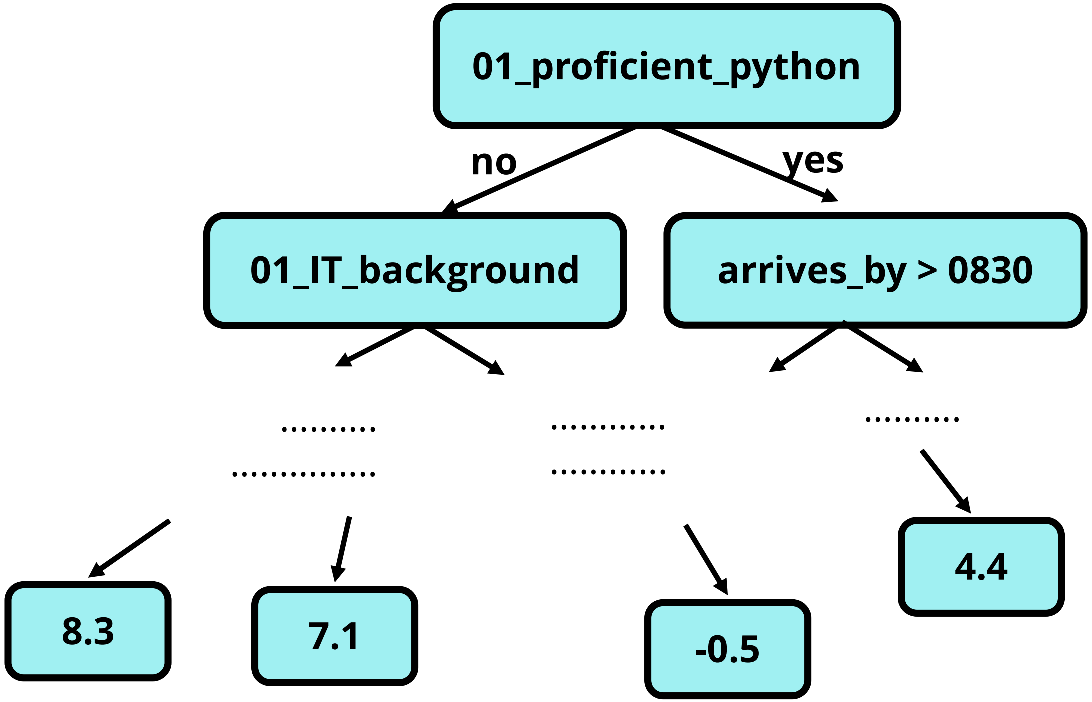


- Example- ***sentiment analysis*** at SSB

## Attempt 5: small decision tree (levels=20)

- Non-parametric **f(X)**, interaction between variables
- Higher splits correspond to more important predictive factors
- Automatically learns the most important predictors and ignores the rest

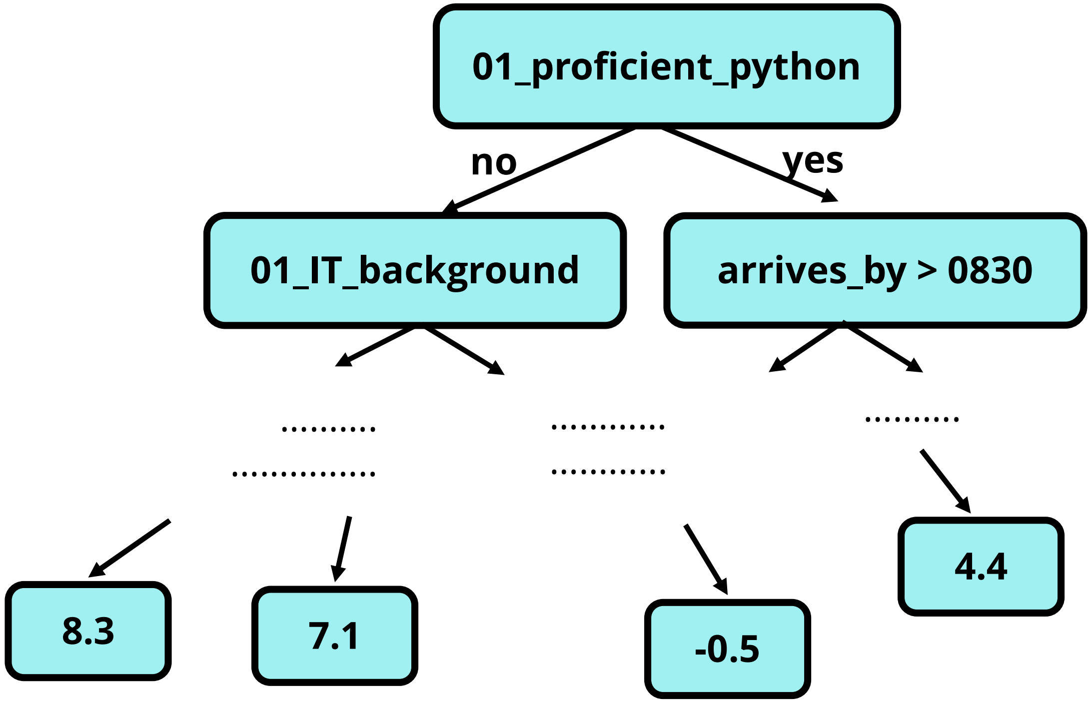
**MAE: 2.4**



- Example- ***sentiment analysis*** at SSB

## Attempt 6: large decision tree

- Unlimited depth- can fit any function



- Example- *sentiment analysis* at SSB

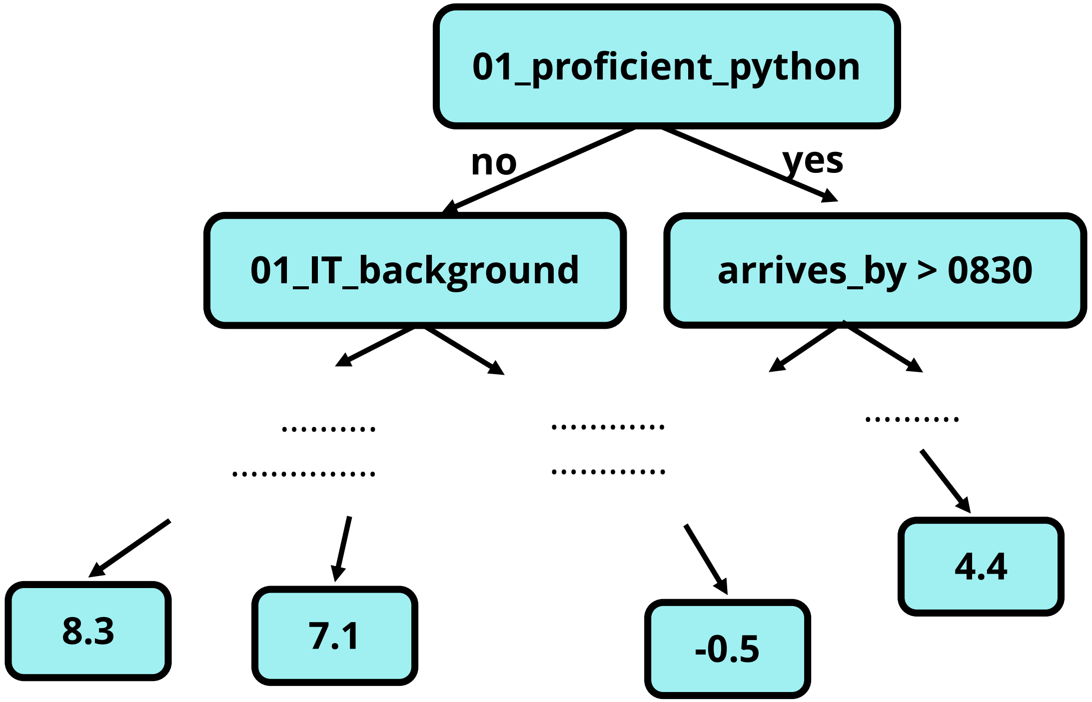
## Attempt 6: large decision tree

- Unlimited depth- can fit any function

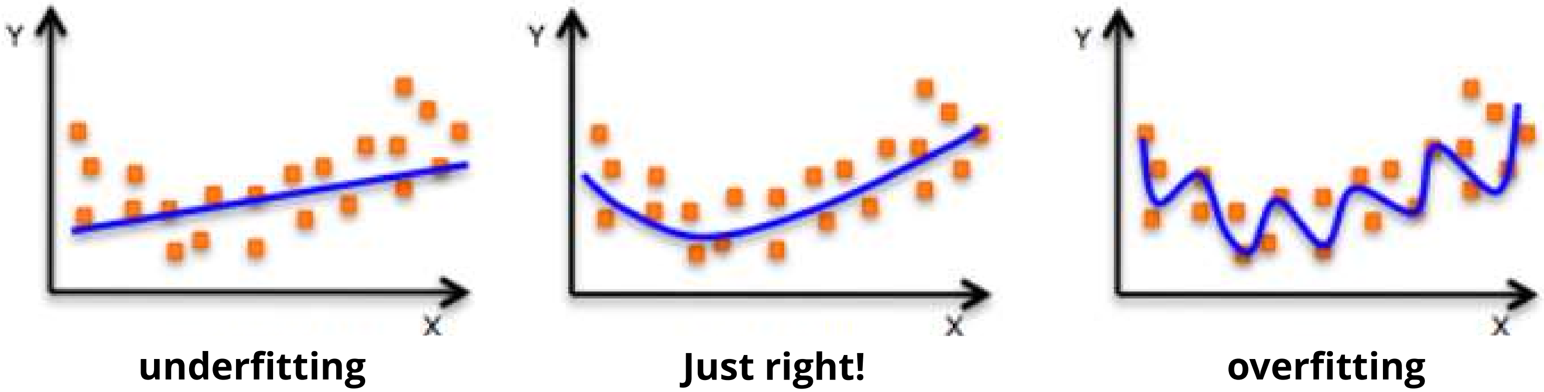
**MAE: 2.9**

- What went wrong??

**Overfitting?**

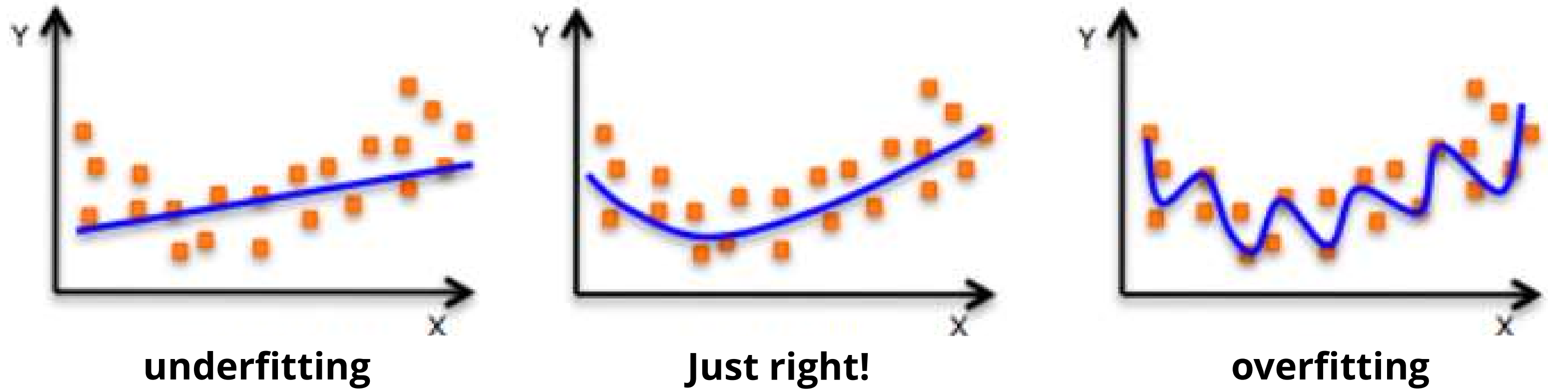


# Overfitting and underfitting




- Underfit: the algorithm lacks sufficient complexity to capture patterns in the data well
- Overfit: the model reflects the particular patterns seen in the training data too closely, and will not perform well on a new sample

# Overfitting and underfitting




- The central problem in machine learning: distinguish between stable patterns in the data that are present in different samples, versus the random variation seen in a particular sample

# Overfitting and underfitting

Source: datapizza

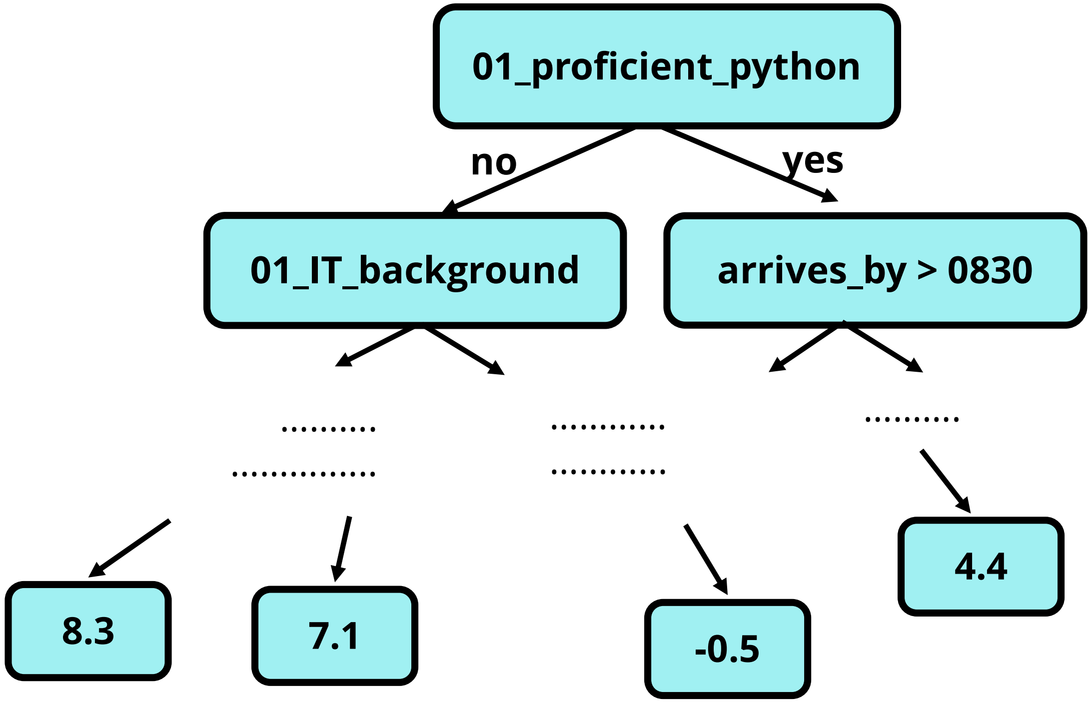
Statistisk sentralbyrå
Statistics Norway

- Example- *sentiment analysis* at SSB

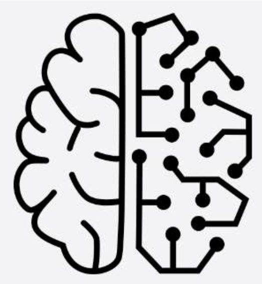
## Attempt 7: random forest

- Algorithm that uses a «forest» on many decision trees
- Each tree uses a smaller set of randomly chosen features from among all variables X
- Good control for overfitting

**MAE: 1.3**



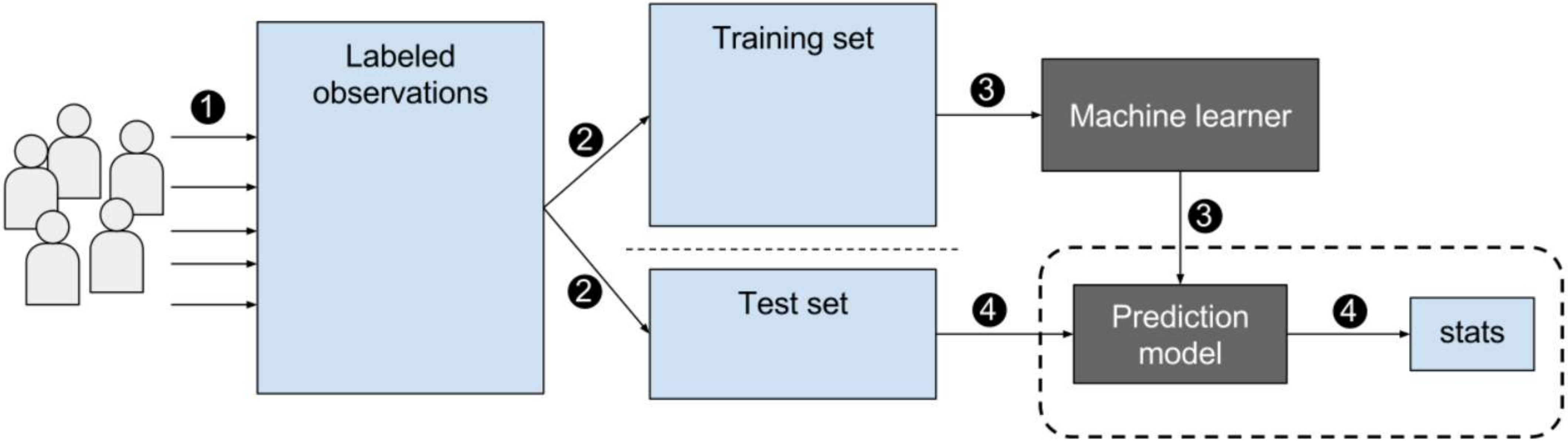
# Definitions



- Supervised learning: given a training set **(X,Y)**, come up with a function $\mathbf{f(X)=Y_{pred}}$
- The goal is to optimize some measure of performance on unseen data (same source as the training data ideally), ie mean squared error
    - Validate the performance on a test set
    - Predict on new data
- Algorithm: a mathematical procedure for building **f(X)**

  vs model: prediction function **f(X)** itself, algorithm fitted to data

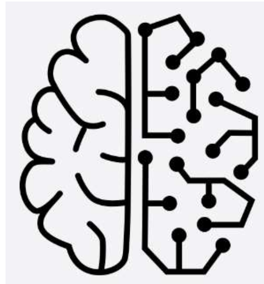
# Training / testing / prediction diagram




Source: wikimedia commons

Statistisk sentralbyrå
Statistics Norway

# Definitions



- Parameters: values that show how the algorithm is fitted to data (high-dimensional vector)
  - Coefficients **β** in a linear regression model $\boldsymbol{\beta}^T \mathbf{X} = \mathbf{Y}_{pred}$
  - Decisions in the branches of a decision tree (level_3_4=var_arrival_time, thresh_3_4=0840,…)
- Hyperparameters: values that determine how the algorithm works (low dimensional vector)
  - Number of terms in a linear regression model
  - Max depth of a decision tree

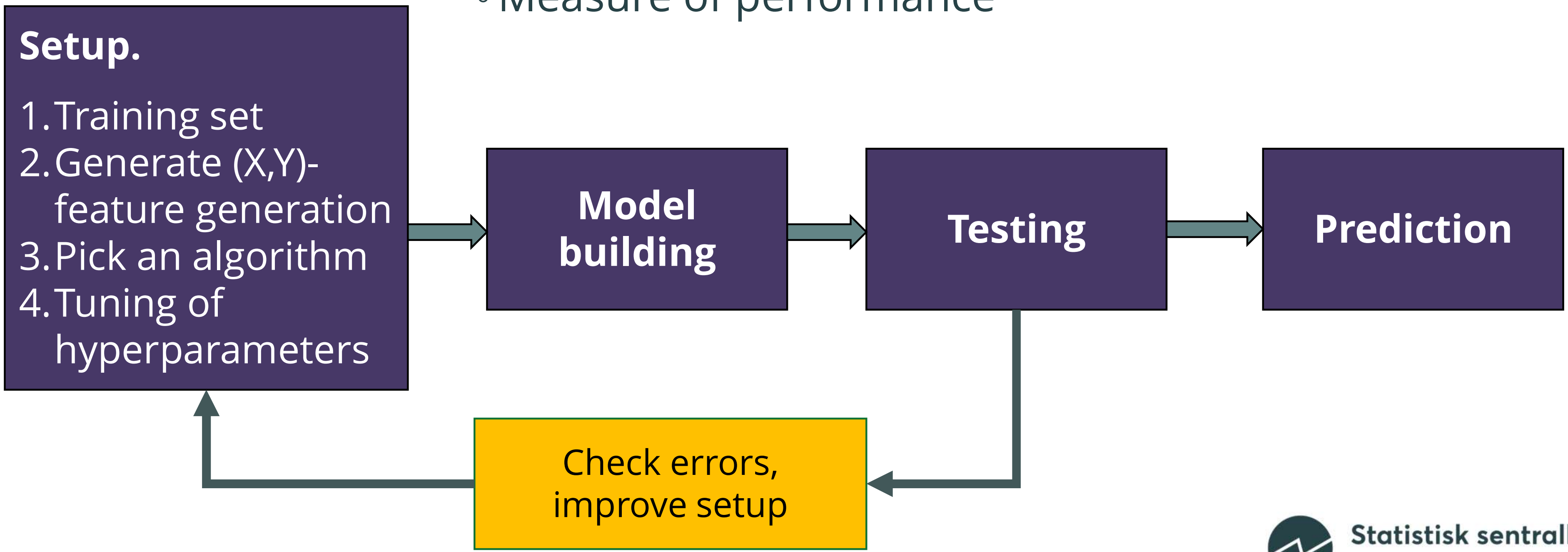
# Performance measures

- Numeric
  - Root mean square error $\sqrt{\frac{1}{n}\sum\left(X_{pred} - Y\right)^2}$
  - Mean absolute error $\frac{1}{n}\sum\left|X_{pred} - Y\right|$
  - **$R^2$** (percent of the variance in y that is explained by X)
- Categorical
  - Accuracy (% correct)
  - But not informative if for example Y : {97% True / 3% False} →

F1 score - GeometricMean(Recall, Precision)

# ML development diagram

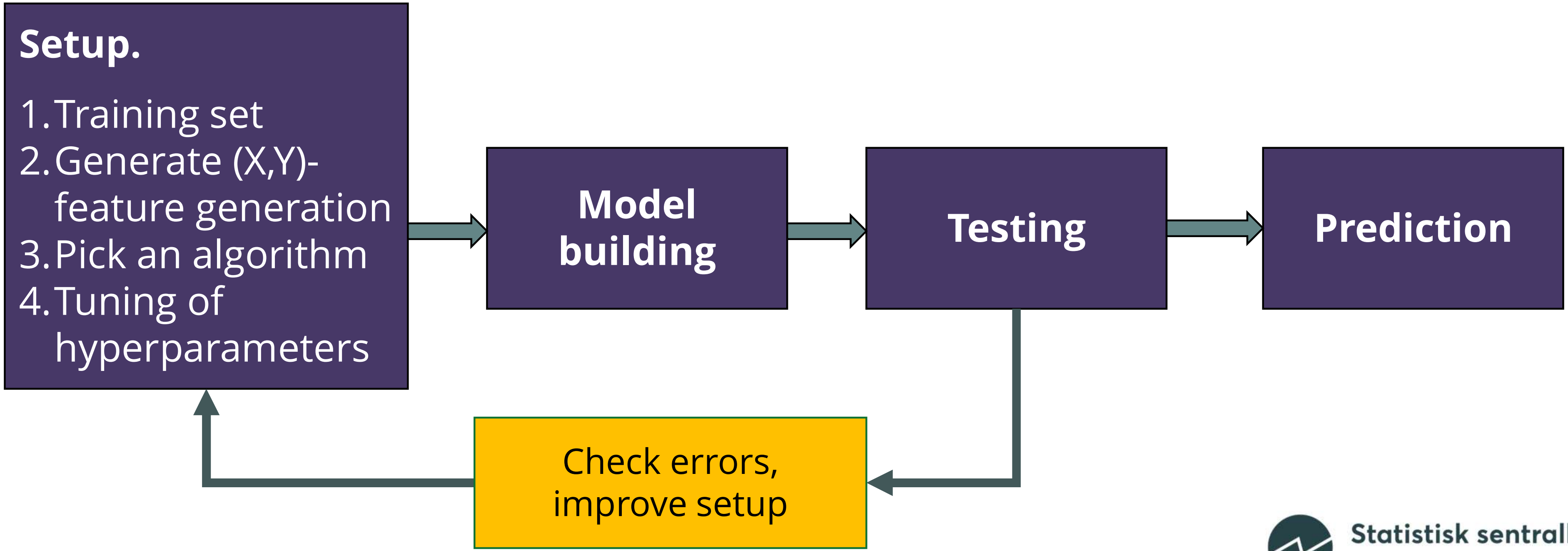
- Define-
  - The problem: what do we want to predict for what sample
  - Measure of performance



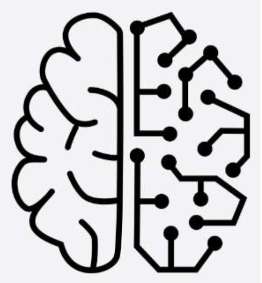

Statistisk sentralbyrå
Statistics Norway

# Procedure

- The four components of the setup determine everything!
  - From there, model building to find **f(X)** is just a numerical optimization problem

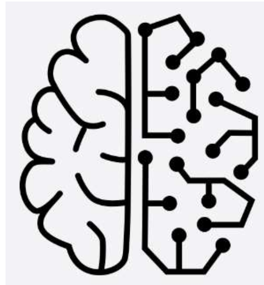


# Procedure



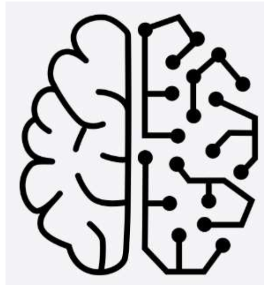
- All main steps can be done as a «black box»!
- Excellent programming utilities to prepare training and test data, make features, fit models, get performance me.asures, etc
- For example in python: classifier.fit(), classifier.predict()

# 1. Training set



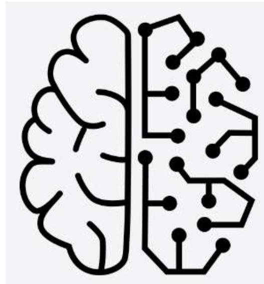
- Labelled data ideally comes from same source as what will be predicted
- Split into training and test set at random, training data requires much larger sets, 80%-20% split typical
    - N=1000 for testing often enough
- In practice: not enough labelled data from the right source for training, so make use of several sources
    - Check quality, balance of datasets of different sizes (reweight?)

# 2. Features (variables)



- Edit and convert - crucial for good performance
- Preprocessing/editing of X,Y.
- Numeric variables:
  - Some algorithms need standardization ( $X \rightarrow (X-\mu)/\sigma$ )
  - Missing values filled with an imputed value (i.e. mean or median)
- Categorical variables:
  - Many algorithms do not work with string values (integer or binary)

# 2. Features (variables)



- (Binary) indicator variables ('dummy' variable)

Store_name → [1_Rema, 1_Kiwi, …, 1_Clas_Ohlson]

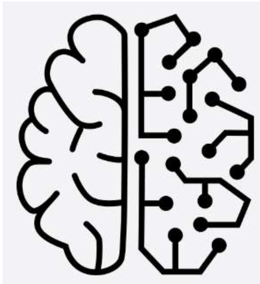
- Vectorizing: text to numeric vectors.

- Split up and use indicator vars (character n-grams, word n-grams)

| Varetekst | ree | eeb | ebo | bok | lær | … |
|---|---|---|---|---|---|---|
| Reebok shoes size 42 | 1 | 1 | 1 | 1 | 0 | … |
| lærebok | 0 | 0 | 1 | 1 | 1 | … |

- Weighting: higher weight to rarer n-grams

# 2. Features



- Crossing features: $X_i*X_j$ (for binary X, this means «both Xi and Xj» )

  *[«coffee», «Egon»]  → 1_coffee_and_Egon*

- Try including more or fewer- feature selection, «pruning»
  - Often enormous number of variables when we have text in the data
  - Statistical methods to choose

# BREAK

Statistisk sentralbyrå
Statistics Norway

# Review

What is supervised learning?

Statistisk sentralbyrå
Statistics Norway

# Review

What is supervised learning?

Learning to make predictions from examples.

# Review

What is a model?
What is an
algorithm?

# Review

What is a model?
What is an algorithm?

Algorithm: a mathematical procedure for building **f(X)**
vs model: prediction function **f(X)** itself, algorithm fitted to data

Statistisk sentralbyrå
Statistics Norway

# Review

What is the difference between a parameter and hyperparameter?

# Review

What is the difference between a parameter and hyperparameter?

Parameters: values that show how the algorithm is fitted to data

Hyperparameters: values that determine how the algorithm works

# Review

What do overfitting and underfitting mean?

Statistisk sentralbyrå
Statistics Norway

# Review

What do overfitting and underfitting mean?

Underfit: the algorithm lacks sufficient complexity to capture patterns in the data well

Overfit: the model reflects the particular patterns seen in the training data too closely, and will not perform well on a new sample

# Review

What is a feature?

# Review

What is a feature?

An input variable to a machine learning algorithm- often a special, converted variable

# Review

What are the four components that affect performance?

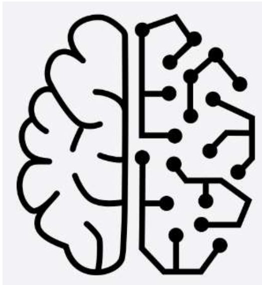
Statistisk sentralbyrå
Statistics Norway

# Review

What are the four components that affect performance?

Training set, generating (X,y), algorithm, and tuning

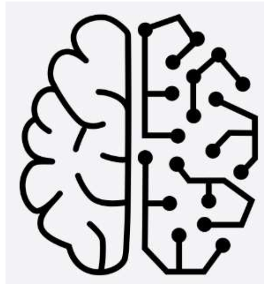
# 3. Algorithms



- Neural networks are the big rage
  - Especially large neural networks (large language models, chat-GPT)
- Most pracitioners use «classic» models
  - Much easier to tune, need less data
- Not important to understand exactly how algorithms work- but:
- Good to know: how are predictions Y are computed from inputs X?
  - What kind of functions can be learned?

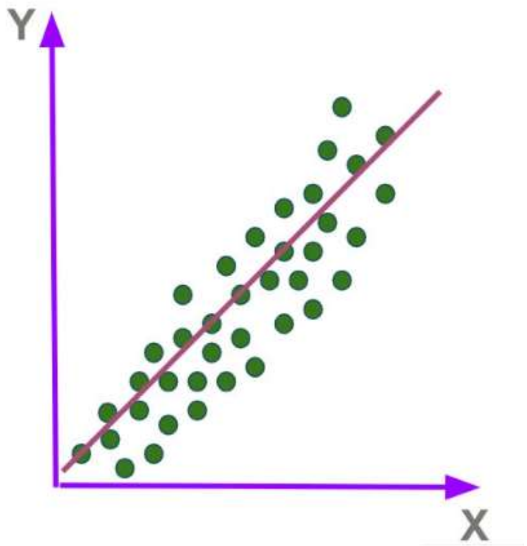
Statistisk sentralbyrå
Statistics Norway

# 3. Algorithms



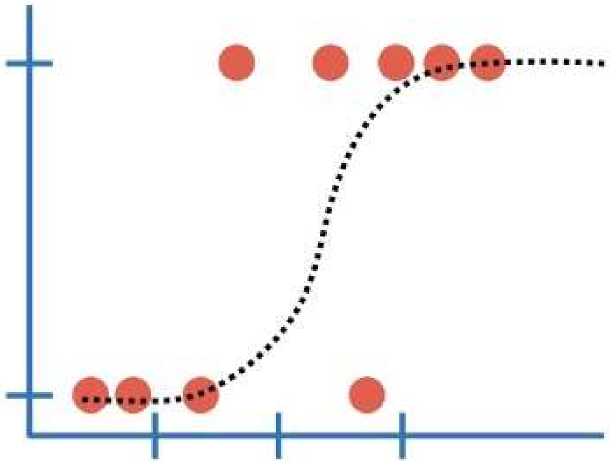
- Considerations:
  - Type of data
  - Structure of the function
  - Control for overfitting, «good algorithm»
  - Interprebility- easy to understand f(X)
  - Sample size needed
  - Can handle many features
  - Resources: Computation time, memory
- Try numerous!

# 3. Algorithms

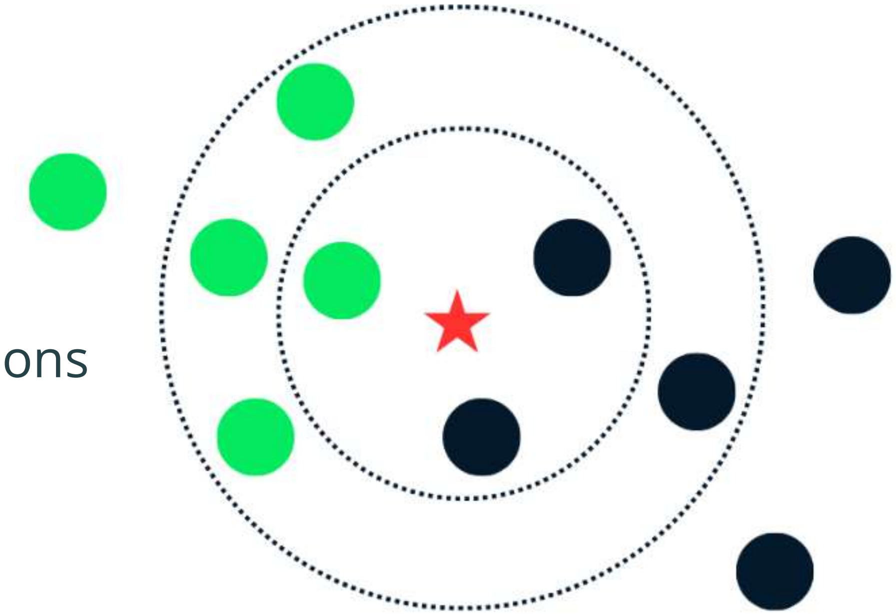



- Linear regression
  - Assumes a very simple structure for f(X), no control for overfitting
  - BUT good variations (ridge regression, lasso regression, polynomial coefficients) that solve these limitations
  - Easy to train and very easy to interpret

- Logistic regression
  - Like linear regression adapted for classification
  - (binary) Probability of Y=1 is related to a linear function of X
  - Used for categorical, not just binary, data



# 3. Algorithms

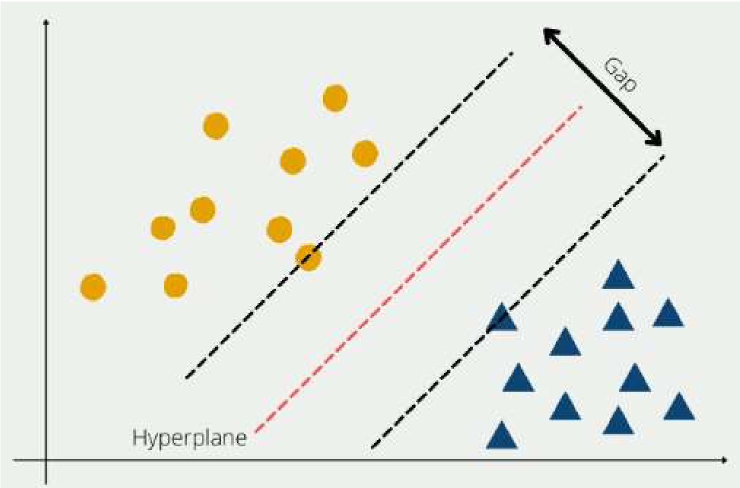


- K-nearest neighbors
  - Any data, Easy to understand, no assumptions
  - Imputation, no ability to generalize
- Naive Bayes
  - For classification
  - Probabilistic model for Pr(Y|X): unrealistic assumptions about independence of variables in X
  - Simple but relatively powerful

    -Doesn’t require much data, also fast to fit on large data

Statistisk sentralbyrå
Statistics Norway

# 3. Algorithms

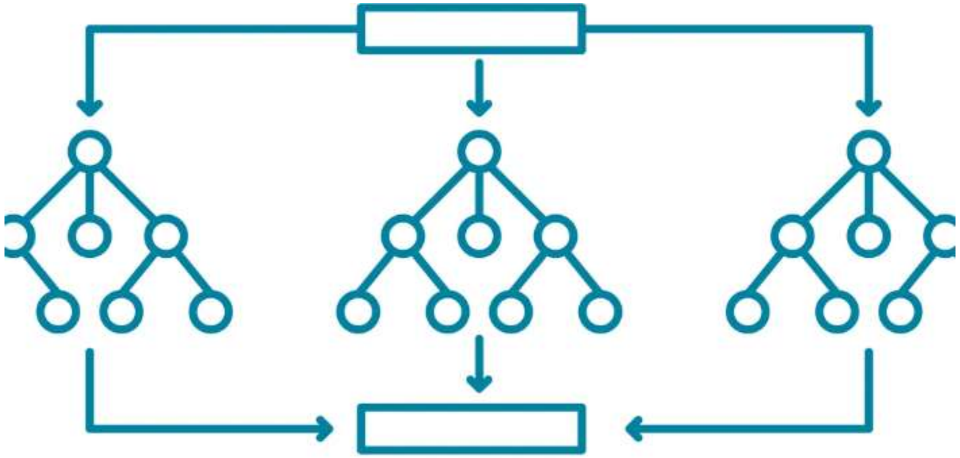



Source: Databasecamp

- Support vector machines
  - Linear function (in simplest version), but advanced properties
  - Can work with small samples and very high dimensional features
  - Computation and memory challenges with large datasets

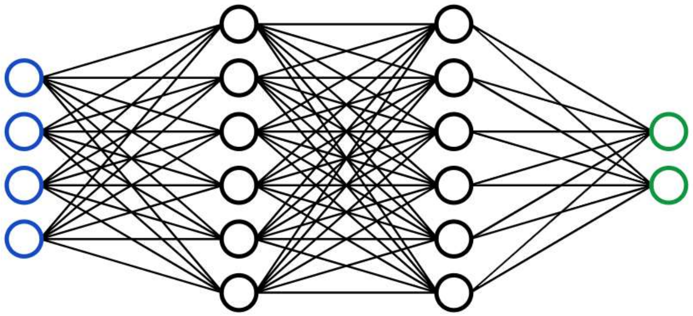
Source:Turing.com

- Random forest
  - Classification or regression
  - Very good performance
  - Controls for overfitting
  - Lots of CPU power, training time, memory



# Other algorithms

- Neural networks
- Large neural networks- deep learning, basis for LLM's (large language models), chatGPT
- Ensemble methods: combine many classifiers



Statistisk sentralbyrå
Statistics Norway

# Machine learning vs traditional statistics

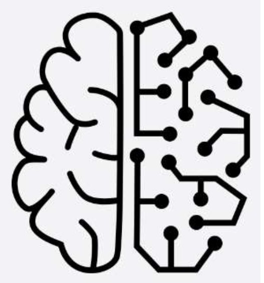
- Much less emphasis on probabilistic assumptions- if empirically we see good performance, we are done
- Good news for beginners ☺
- No longer have to code much, but good to understand certain concepts
- No such thing as a best algorithm, just try many

# 4. Model tuning- hyperparameters

- Hyperparameters: values that determine how the algorithm works
- *RandomForest(n_estimators=100, max_depth=None, min_samples_leaf=1)*
- Cross validation: hold out part of the training set to test some setting of hyperparameters.


Statististisk sentralbyrå
Statistics Norway

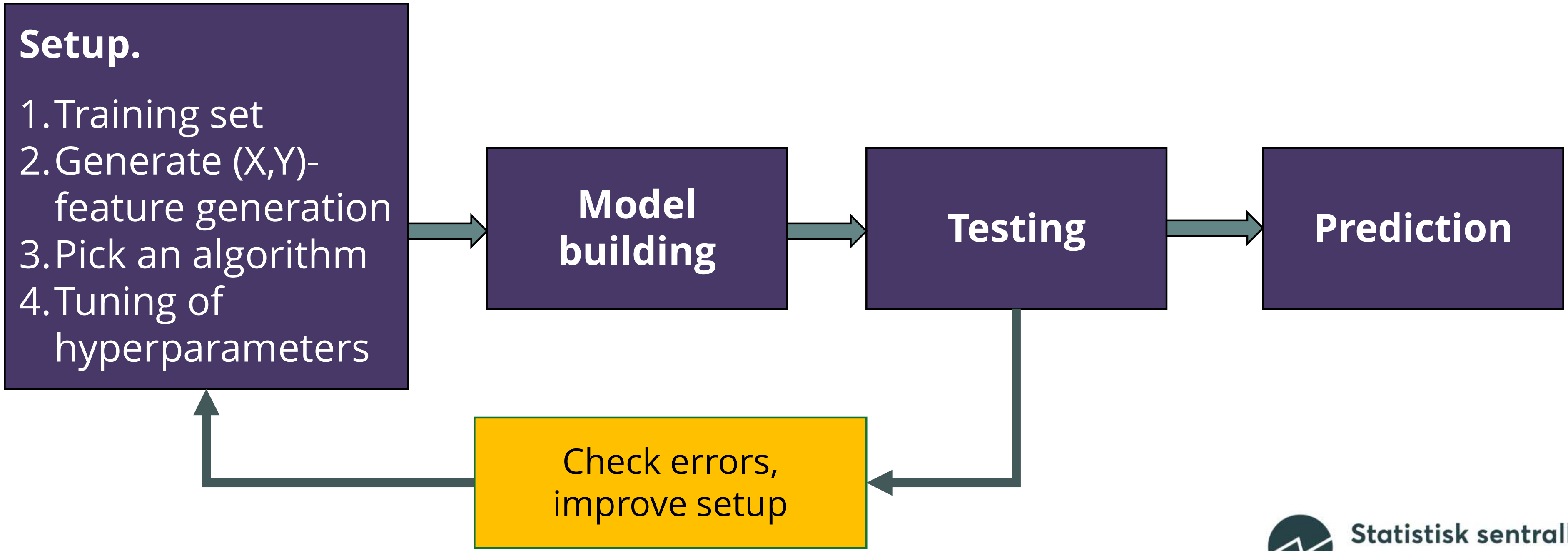
# Testing and predicting



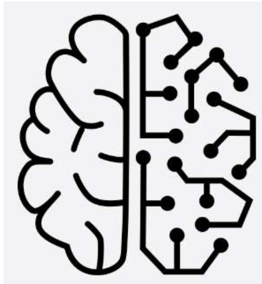
- Test on an unseen set, not so easy in practice!
- NO decision taken after having seen test set (i.e. editing of variables) when checking performance
- Check performance manually
  - Peformance in various groups is interesting (foods vs non-foods)
  - Manually inspect errors

# Procedure

- The four components of the setup determine everything!
  - From there, model building to find **f(X)** is just a numerical optimization problem



# Testing and predicting



- Supplement machine learning with rule-based?
- Human-in-the-loop setup for coding: machine learning and human work in synergy
- For classification problems, algorithms return the predicted class and prediction probabilities
- Machine learning : picks out most useful examples for manual coding (low prediction probability), sorts items by predicted labels
- Human: expands the training set

# Diagnosing- what went wrong?

- The first question- can I expect very high accuracy?
- $Y = \beta X + \varepsilon$,  where $\varepsilon$ is random noise
- Other factors: subjectivity in labelling, double-coder experiment
- I run out of memory or time
  - Dapla options at SSB: cluster vs local, *arbeidsmiljø* server vs *stor maskin*
  - Feature pruning
  - Use another algorithm

# What went wrong?

- Testing performance is mediocre- is training performance also mediocre?
- Training performance mediocre:
  - Choice of algorithm is a bad fit
  - Hyperparameters off, ie model complexity too low
  - Need better features (noisy features are typically ignored)
  - Data quality issues (i.e. incorrect labellings, preprocessing, …)
- Use subset of the training set?

Statististk sentralbyrå
Statistics Norway

# What went wrong?

- Testing performance (surprisingly) mediocre
  - Training set too small
  - Training set not representative
  - Data quality issues in the test set
  - Overfitting, hyperparameters

# Summarizing

(Putting aside data quality issues)

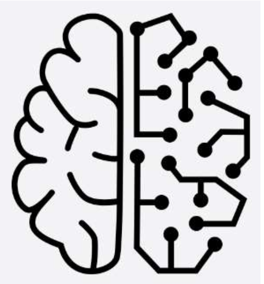
- Mediocre performance on the training set → algorithm as fitted doesn’t capture patterns in the data well. Another approach may or may not be better.
- Decent performance in training, big drop when testing on data → training and test data fundamentally different, OR overfitting

Statistisk sentralbyrå
Statistics Norway

# Good tools

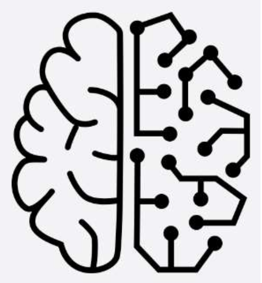
- Python (sklearn)
  - Pipeline object-
    - build complex machine learning systems with very little code
    - very transparent and organized,
    - Makes sure training and test data are kept separate
  - Huge range of functions available: hyperparameter tuning, feature generation, analysis, ...
- R, also Google, Microsoft

# Takeaways



1) Test on fresh unseen data after ANY changes in the workflow (preprocessing or machine learning)
2) Don't be intimidated by all the theory, empirical performance is everything!
   - Need a large representative test set
   - «black box» approach

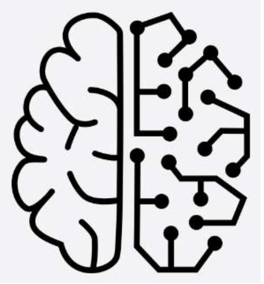
# Takeaways



3) Excellent programming support
   - Sklearn in Python, classifier.fit(), classifier. Predict()
   - Easy to try many possible algorithms, features, hyperparameter settings just changing a few lines of code!
4) The 4 factors: training set, features, algorithm, tuning.

   Trial and error!

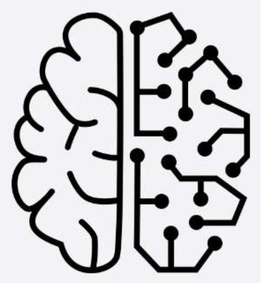
# Takeaways



5) OK to combine with manual labelling

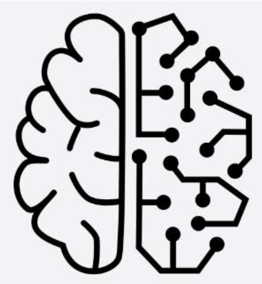
- Human-in-the-loop, predictions can help greatly
- Use prediction probabilities from the model.

# A few resources



- SSB's group on AI
- Hackathon SSB

# Lab session

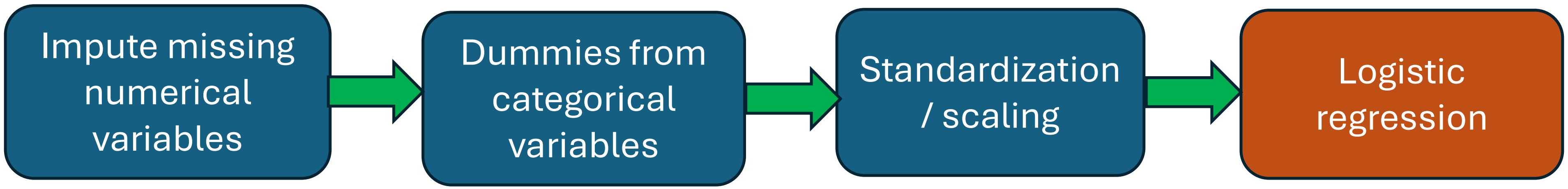


- Python sklearn package
  - Classification and regression
  - All steps of a machine learning workflow on real data
- Today: 12:30-15:00 Befolkning møtesenter
- Next Friday 13.9: 12-15 Arbeid møtesenter

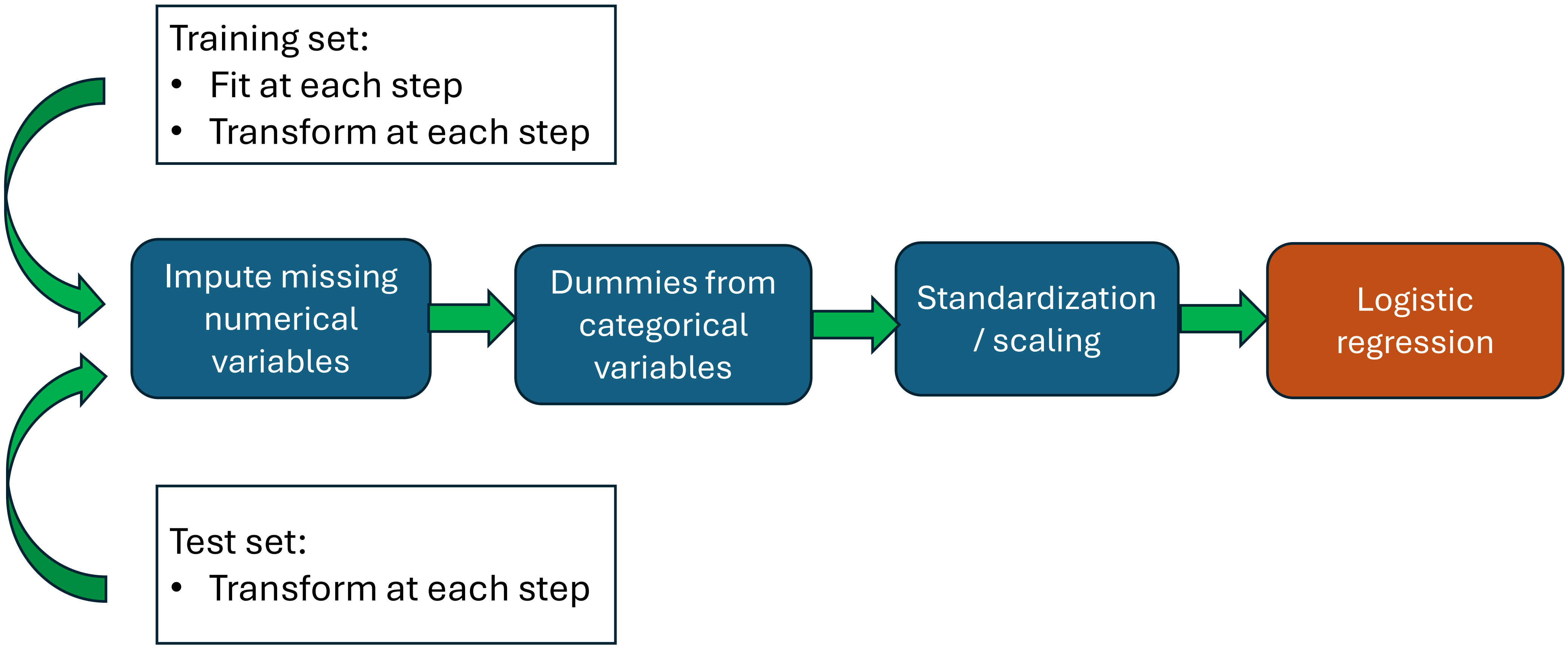
# A small taste of pipelines in Python


- We typically need some preprocessing steps before fitting a model

# A small taste of pipelines in Python



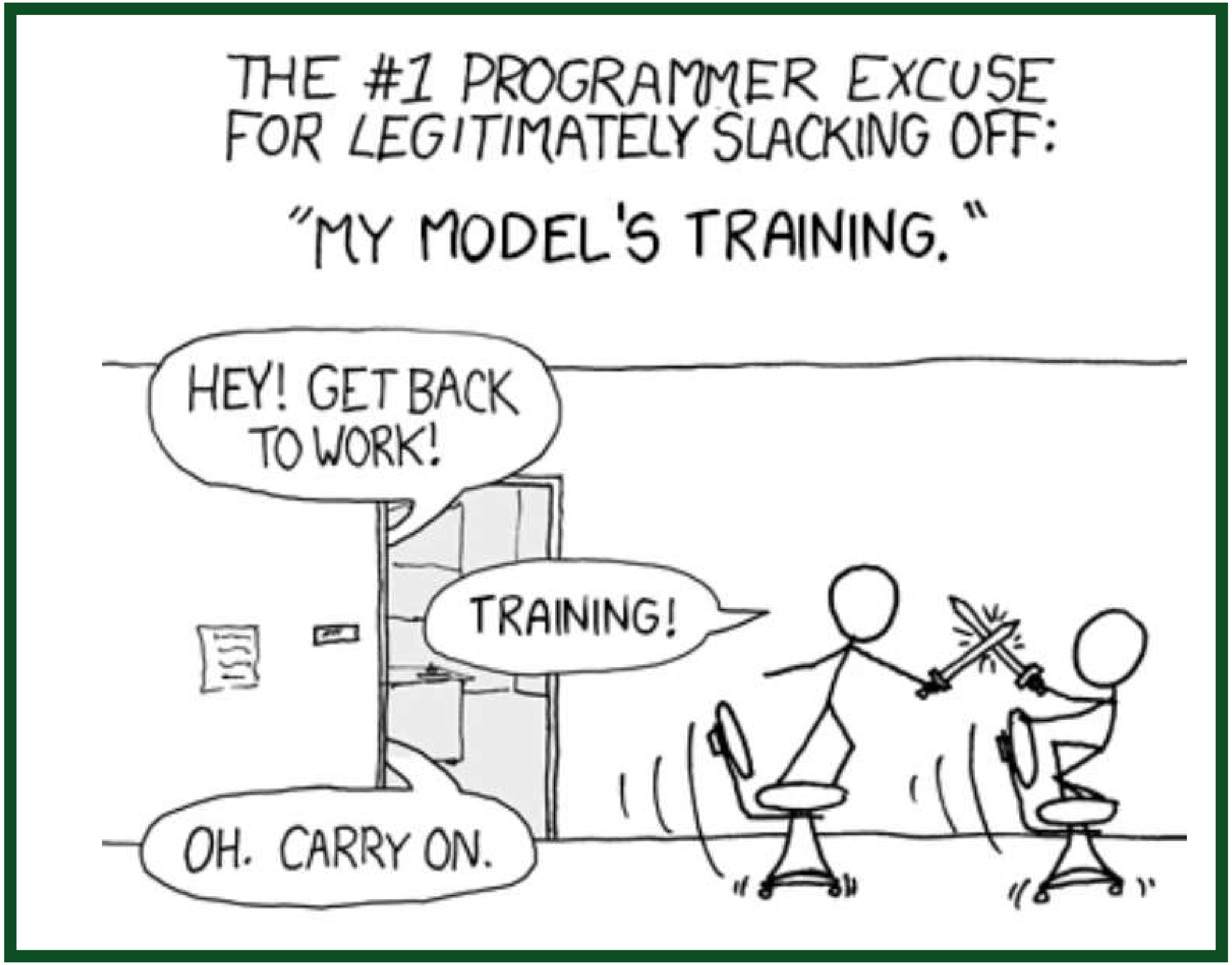
# A small taste of pipelines in Python



# A small taste of pipelines in Python

```
mymodel = Pipeline([   ('imputer', SimpleImputer(strategy ='mean')),
                       ('second_order_interactions’, PolynomialFeatures(2)),
                       ('LinReg', LinearRegression())    ])
```

- Clean and elegant code
- Easy to run different experiments
- Makes sure test data is never leaked when adjusting the preprocessing steps or training the model!




Source: Ifunny.co / mimipedia