UiO : University of Oslo

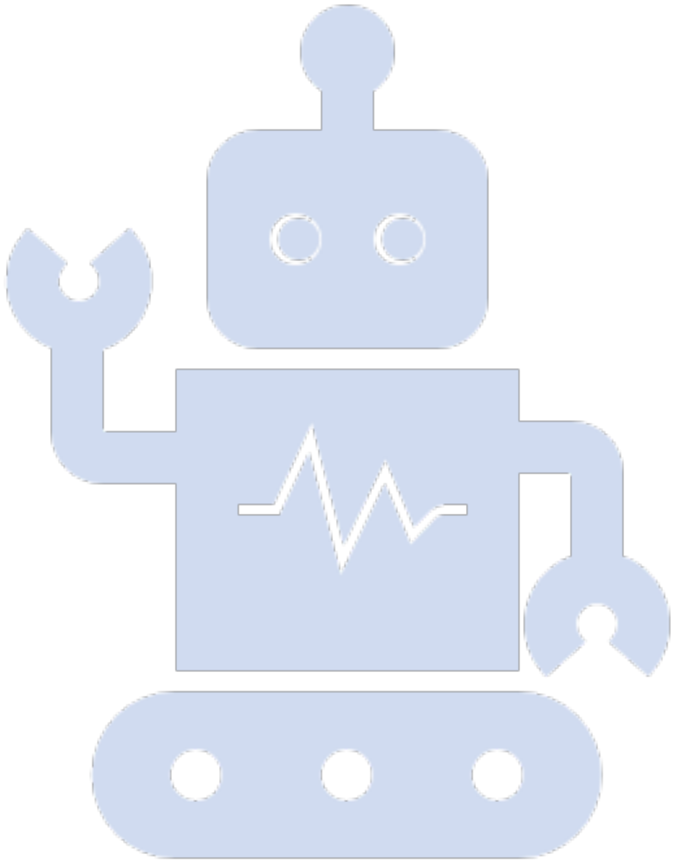
institutt for informatikk

IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning

Lecture 3

Supervised learning 1

Ali Ramezani-Kebrya



5.1 Machine learning

IN3050/IN4050 Introduction to Artificial Intelligence

and Machine Learning

Automated Learning of meaningful patterns in data (understanding Machine Learning)

Machine Learning

Converting experience in to expertise/knowledge

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. (Wikipedia)

- (various more or less precise and more or less equivalent definitions)

What does it mean to learn?

- If humans learn:
  - all the timetables for all the public transport in Oslo, would you say that they have learned?
  - all the entries in a bilingual dictionary?
- If a machine does the same, has it learned?

Generalization Matters
(unseen data)

T-bane Metro
Soria Moria
Take T-line 1 here
Frognerseteren
Sognsvann
Ringen
Vestli
Ellingsrudåsen
Østerås
Kolsås
Bergkrystallen
Mortensrud
Sentrum
City centre

Machine Learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. (Wikipedia)

In particular :

- **Generalization:** Provide sensible outputs for inputs not encountered during training
- **extracting relevant information** from data and **applying it to analyze new data.**

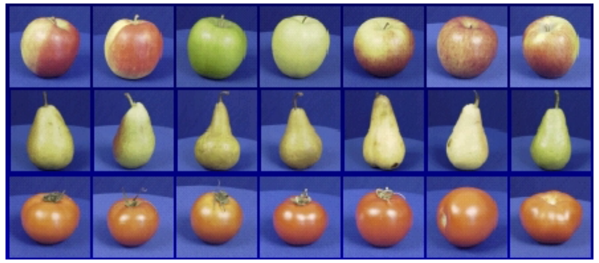
Formalizing the Learning Problem (Remember?)

- Learning = Improving with experience at some task
  - Improve over task *T*
  - with respect to performance measure *P*
  - based on experience *E*

Types of ML (Remember?)

Supervised learning

- Learn from labeled data



-

?

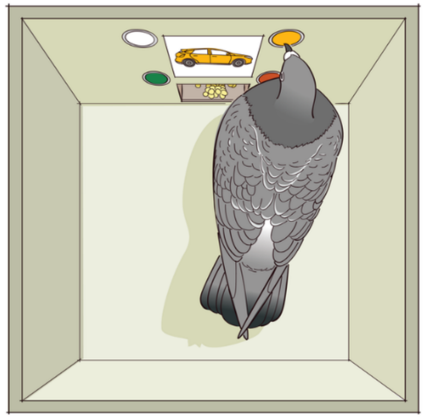
Unsupervised learning

- No labeled data

- Task: identify similarities and categorize together

Reinforcement learning

- Training with rewards (and punishments)

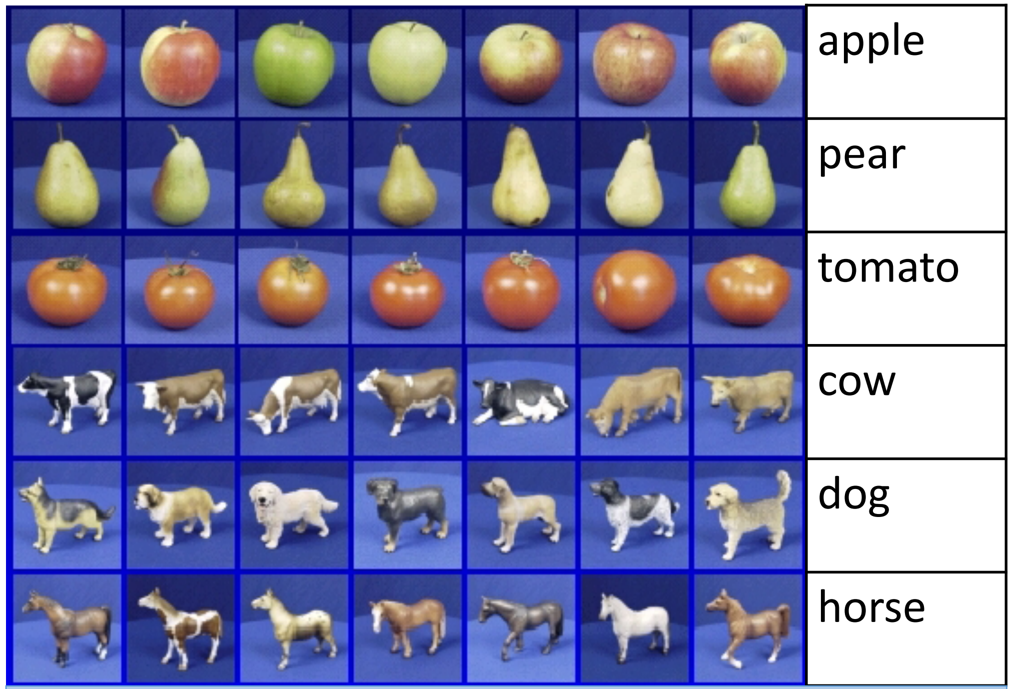


Source: Wikipedia

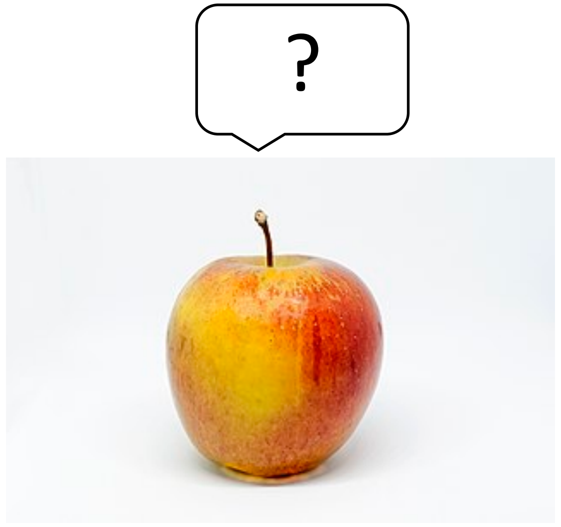
Supervised learning

= Learning from exemplars

Training data



Each item is labelled



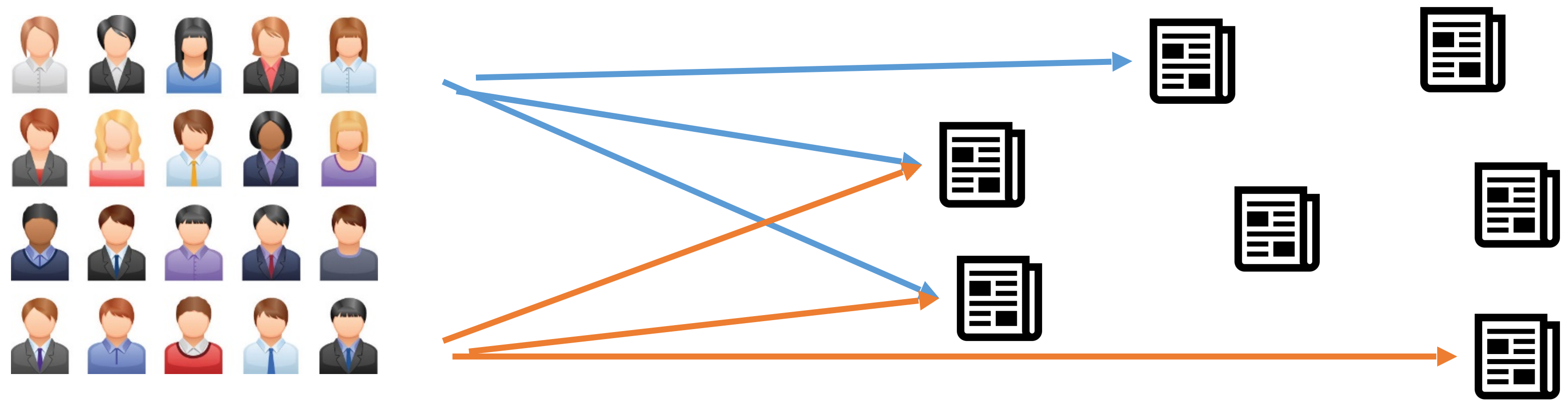
Task:

Predict the label on unseen items

Unsupervised learning

- *Can you sort the Lego bricks?*
    - (No instruction on how)
- You may choose sorting on
    - Color, or
    - Size, or
    - Shape, or
    - A combination
- I cannot know beforehand what you choose, but
- The result might be helpful

Unsupervised learning, example 2

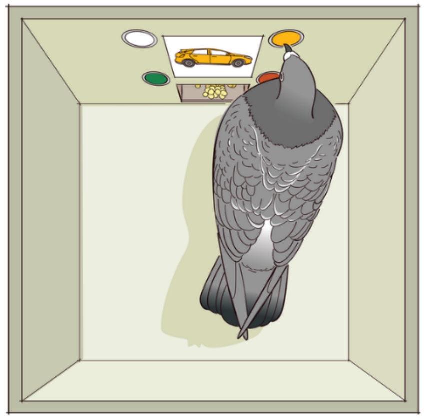


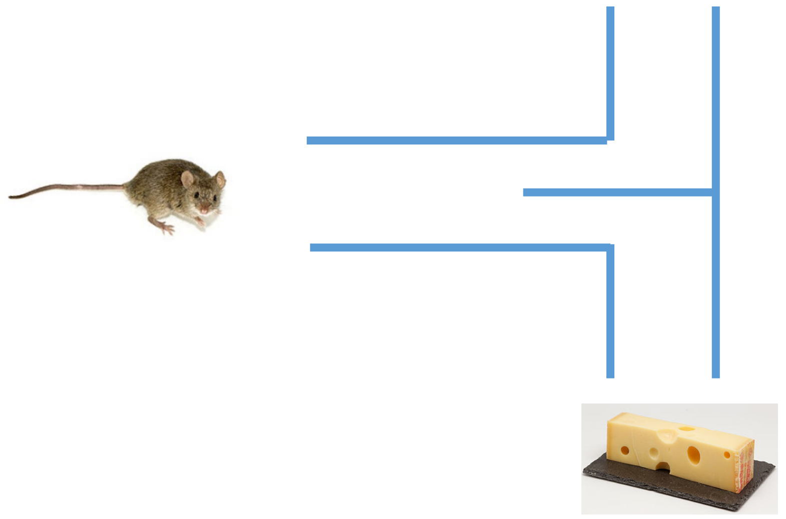
- Everybody (Facebook, Schibsted, ..) collects what you are reading
- Readers who read the same stories are considered similar
- Unsupervised learning can cluster readers based on what they read

- Recommend new stories based on what others in your cluster have read
- Convenient for you
- Danger: Echo chambers

population is divided into multiple cultures
not reading others / interacting with others

Reinforcement learning (from Operant Conditioning in Psychology)



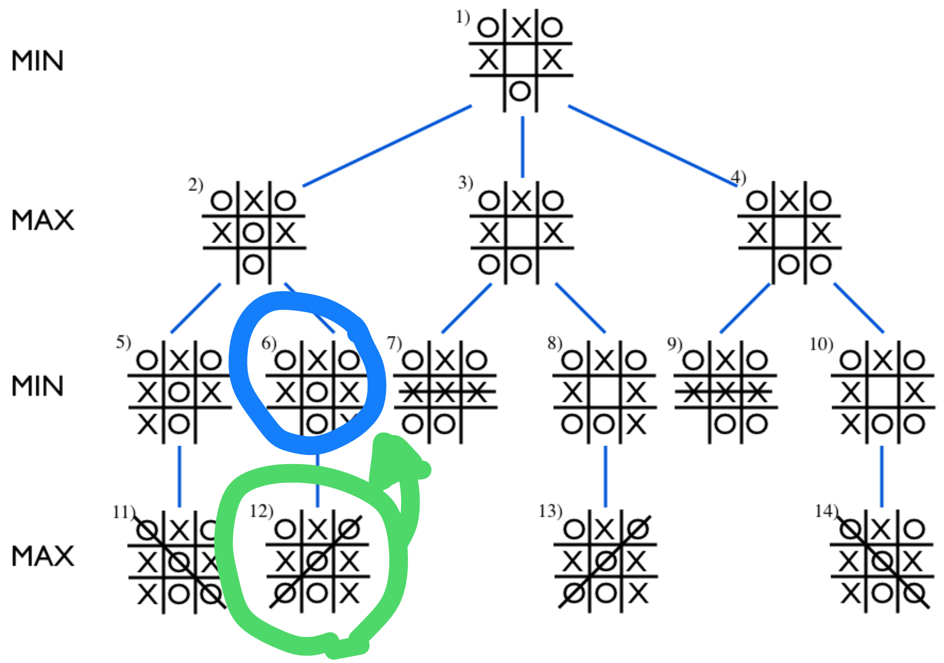


- The pigeon is given a reward if it pecks the right button when it sees the image.
- It learns this task

- An animal can learn the way around a maze if it gets a reward at the end and practices repeatedly
    - More complex mazes

Game playing

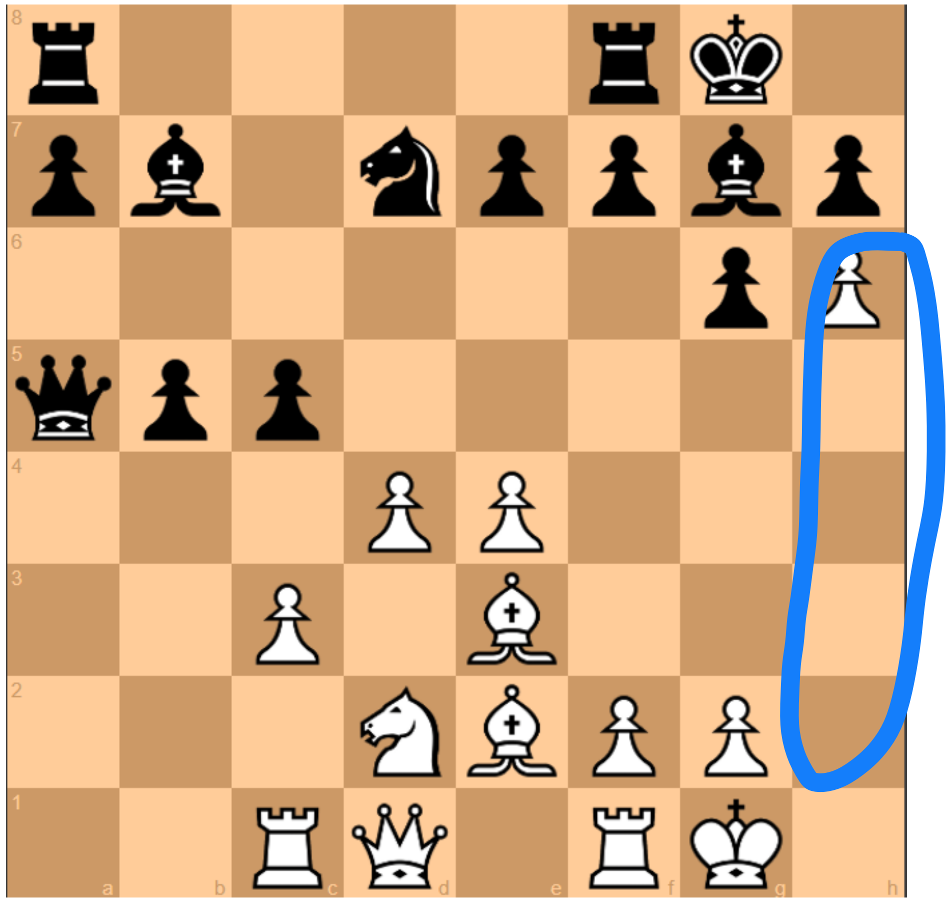
- A game tree is like a maze
- By playing many times you may recognize first a winning position
- Next time you may recognize a position immediately before a winning position etc.



https://materiaalit.github.io/intro-to-ai/part2/

AlphaZero

- It beats humans in Go and Chess
- Totally self-learned by playing against itself:
  - Reinforcement learning
  - Neural nets to generalize over game states
- Human plays, e.g., Carlsen, has learned new strategies from the program
  - ''The *h* pawn''



https://www.aftenposten.no/sport/sjakk/i/K3K31M/ny-oppfinnelse-er-en-sjakk-sensasjon-slik-paavirkes-carlsen

The next weeks: supervised learning

Learning algorithms

1. Nearest neighbors
2. Perceptron
3. Linear regression
4. Simple neural networks
5. Multi-layer neural networks
   - Backpropagation

- and more

Aspects of the algorithms

- What are the underlying ideas?
- How are they implemented?
  - (1-5)
- What are the use cases?
- How can we apply them practically?

Next weeks: The machine learning process

Steps in the process

1. Data Collection and Preparation
2. Feature Selection and extraction
3. Algorithm Choice
4. Model Selection
5. Parameter Selection
6. Training
7. Evaluation

- Back and forth. 2-6

What we will do

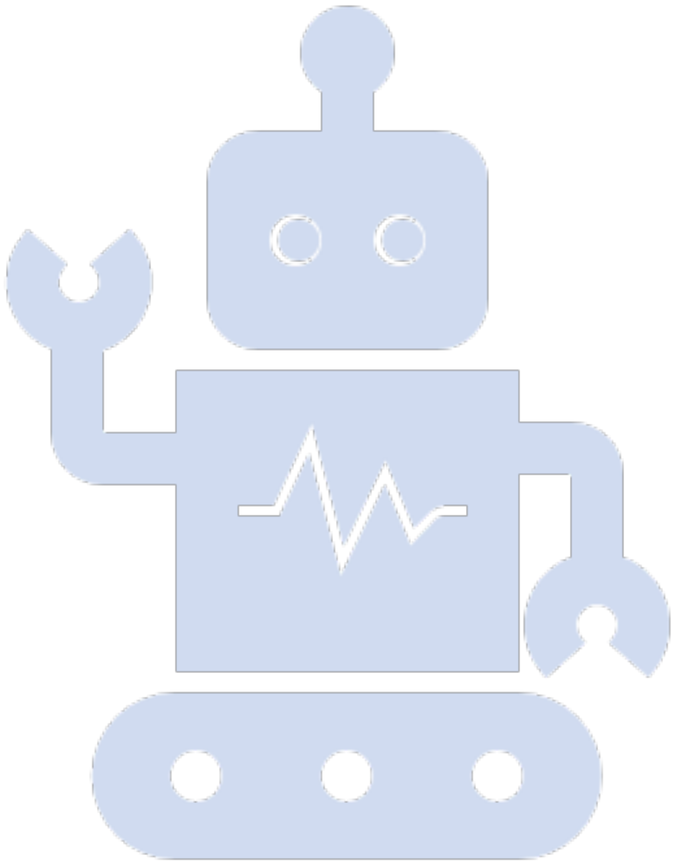
- Equally important as the algorithms.
- We will do this in parallel to the algorithms
- Starting with the basics and then introduce refinements

Most and foremost supervised learning

- Techniques from supervised learning basis for the other two, e.g.,
  - Classification $\rightarrow$ clustering (e.g., *k*-means clustering)
  - Recognizing similarities
  - Neural networks
- Experiments and evaluation are similar
  - Features
  - Test and training set
  - Evaluation measures
- Supervised learning is ‘’the biggest field’’
  - Most algorithms

5.2 Classification and Features

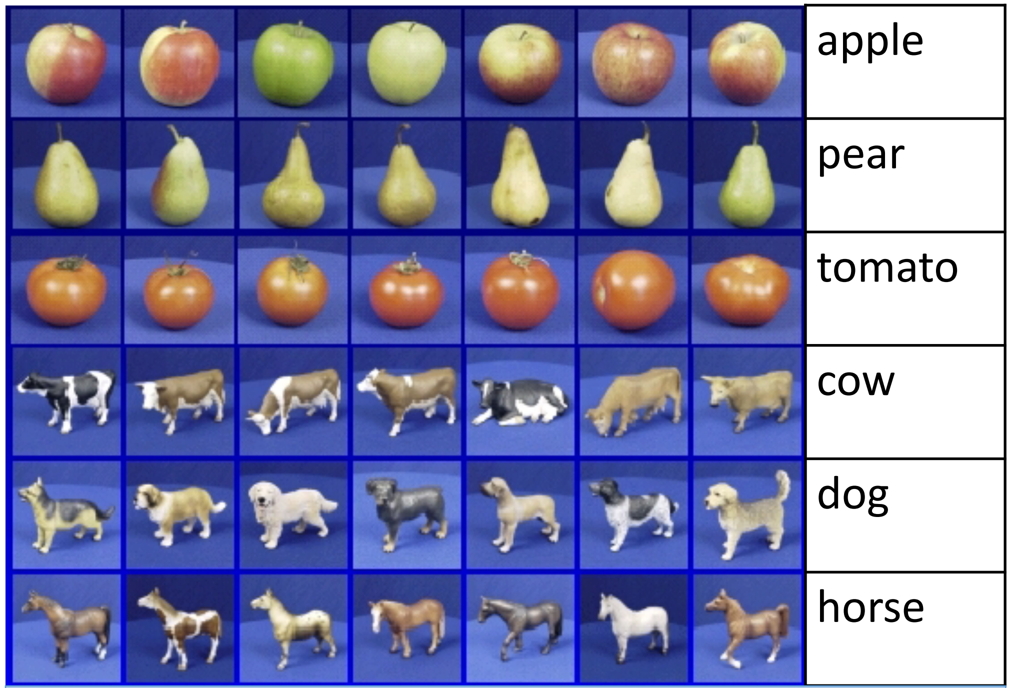
IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



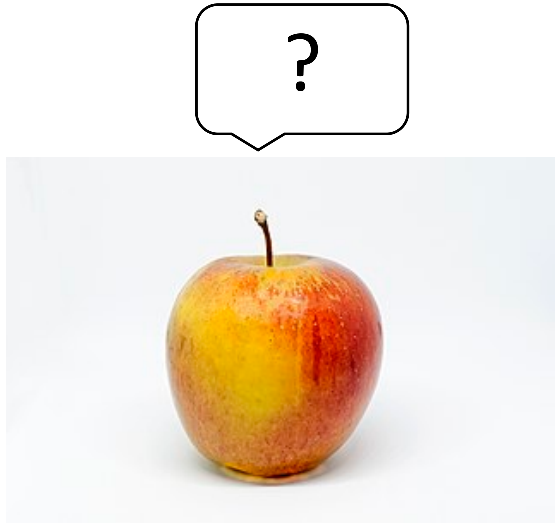
Supervised learning

= Learning from exemplars

Training data



Each item is labeled



Task:

Predict the label on unseen items

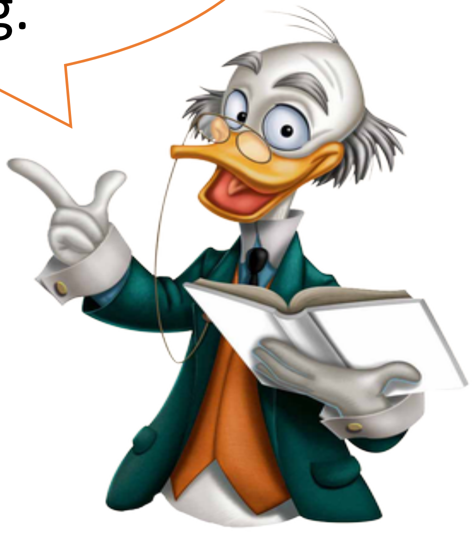
What is a classifier?

- A domain of objects/input features we are to classify
- Labels: A finite set of labels
- Classifier: A mapping which maps each object to a unique label

Example 1: (Decision Trees)

Professor, do you think I will enjoy IN3050?

I can give you a scientific answer using machine learning.



Example 1: (Decision Trees)

Domain

- (Prospective IN3050/IN4050) Students

Labels

- Will enjoy the course (yes/1)
- Will not enjoy the course (no/0)

Example 2: Spam mail

Domain

- E-mails

Labels

- Spam: yes
- Spam: no

Example 3: The MNIST data set

Domain

- Hand-written digits

https://en.wikipedia.org/wiki/MNIST_database

Labels

0
1
2
3
4
5
6
7
8
9

- Task
- To each hand-written picture of a digit, predict the correct digit
- There are 10 different classes

Example 4: Language model

The cat is on the …

- Task: Predict the next word!
- Labels:
  - A vocabulary of words:
    - *aardvark, …, mat, …, roof, …*
    - E.g., 100,000 words/labels
- Domain:
  - Finite sequences of words

- Properties: Large training data
  - 1 billion training instances
  - Supervised learning
    - But you do not have to hand-label the training data.

Classes

- Binary classification:
  - Two classes
- Multiclass classification:
  - Three or more classes

- Observe:
  - Some learning algorithms are by nature binary (e.g., the perceptron) and have to be adapted to multiclass classification
  - Other algorithms naturally admit any number of classes (e.g., decision tree)

Features

To be able to classify objects, we have to make a representation of them:

1. Decide on a set of attributes/ features we can observe
2. Decide on the set of possible values for each attribute
3. Extract the values of the attributes for each object

Example 1: (Decision Trees)

Domain

- (Prospective IN3050/IN4050) Students

Features

A questionnaire:

- Do you like mathematics?
  - Yes
  - no
- Do you have programming experience?
  - None
  - some (1 or 2 courses)
  - good (= 3 or more courses)
- Have you taken advanced machine learning courses?
  - Yes
  - no

Results of the 2020 survey: a data set

Cand no	Enjoy maths	Programming	Adv. ML	Enjoy
1	Y	Good	N	Y
2	Y	Some	N	Y
3	N	Good	Y	N
4	N	None	N	N
5	N	Good	N	Y
6	N	Good	Y	Y
….				

Transforming the classification task

- The task of predicting from a student

$\longrightarrow$ yes/no

- Is transformed to the task of predicting from some features

(Maths: Yes, Programming: Good, Adv.ML: No) $\longrightarrow$ yes/no

Example 2: email spam

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
…							
50	no	15,829	242	0	no	html	small

From OpenIntro Statistics
Creative Commons license

There are more variables (attributes) in the data set

- Data are typically represented in a table
- Each column one attribute
- Each row an observation (n-tuple, vector)
- (cf. Data base)

Transforming the classification task

- The task of predicting from an e-mail

$\longrightarrow$ yes/no

- is transformed to the task of predicting from some features

(Chars: 21,705, Lines: 551, ‘dollar’: 0,
‘winner’: no, format: html, number: small) $\longrightarrow$ yes/no

The larger picture

- This is how data sets are presented in texts on statistics or machine learning.
- But in real life, you want to apply ML to new tasks, then there is a lot of work before you have a data set like that:
    1. Data Collection and Preparation
    2. Feature Selection and extraction
- And for supervised learning, in particular
    3. Label the data, e.g., whether an x-ray shows cancer

In this course

- We will mainly follow the trend and use pre-made data sets
- Concentrate on the more algorithmic and experimental parts of ML
- Techniques for, e.g., preprocessing and feature selection is to a certain degree domain specific. Hence this is left to e.g., courses in
    - Natural Language Processing
    - Image Processing
    - Bio Informatics
    - Robotics

Example 2: email spam

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
…							
50	no	15,829	242	0	no	html	small

🟩 50 observations, rows

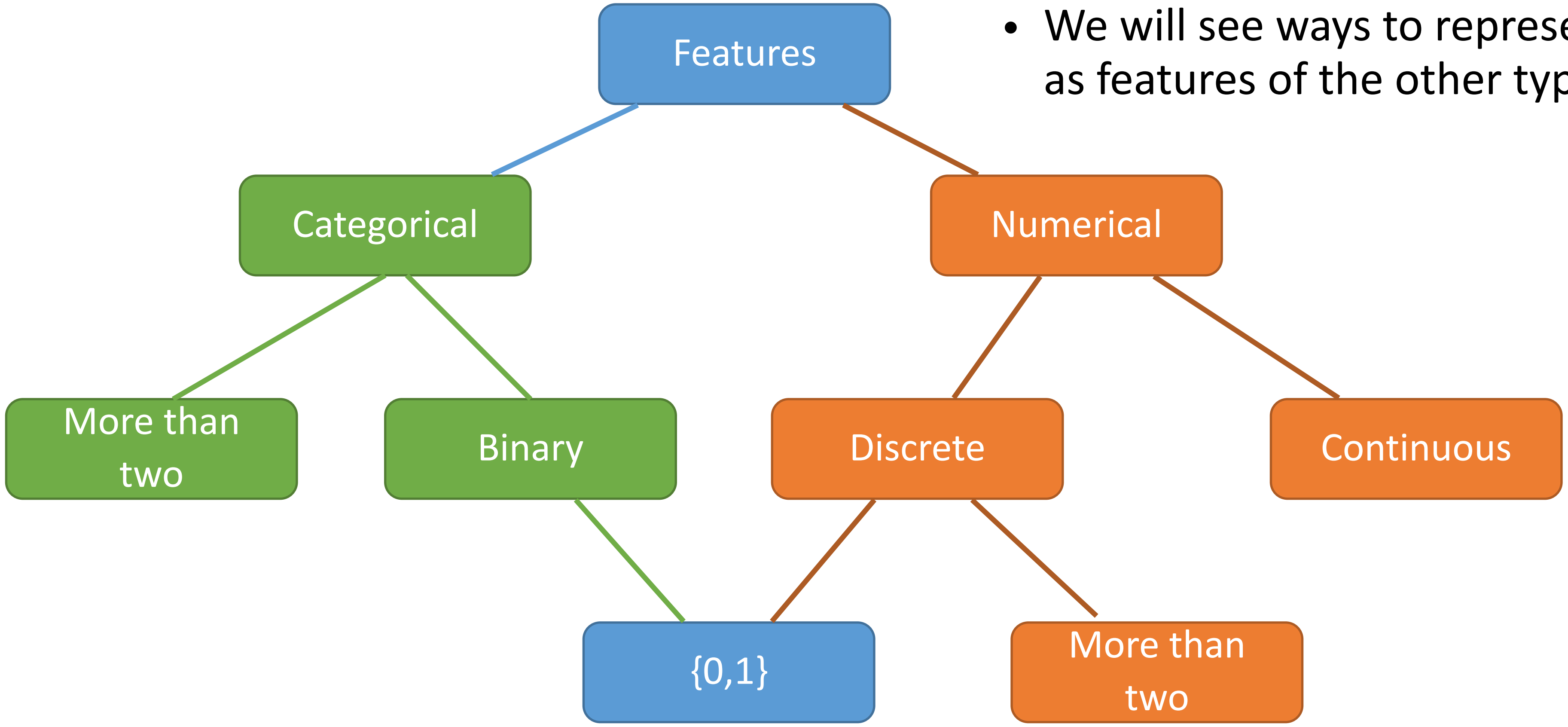
🟦 7 variables, columns

🟦 4 categorical variables

🟧 3 numerical variables

Feature types

- Some algorithms can only take numerical features
  - *k*NN, perceptron,
- Some algorithms take categorical features
  - Decision trees
- Some software can only take numerical features
  - Scikit learn
- We will see ways to represent a feature of one type as features of the other type



From data set to features

- In some cases, the attributes of a data set can be used directly as features in an ML-algorithm.
- In other cases, it is necessary with some further preprocessing, e.g.
  - Transforming categorical data to numerical data
  - Scaling the data
  - Combining features
  - ‘’Feature engineering’’
  - Fixing the positions of the features
- The result is a fixed shape numerical vector

$$\boldsymbol{x} = (x_1, x_2, \dots, x_n)$$

Transforming the classification task

- The task of predicting from an e-mail

  → yes/no

- is transformed to the task of predicting from some features

  (Chars: 21,705, Lines: 551, ‘dollar’: 0, ‘winner’: no, format: html, number: small) → yes/no

- is transformed to the task of predicting from a numerical vector to a number

  $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \longrightarrow y \in \{0, 1\}$

Some words on notation

- The input as a numerical feature vector $\boldsymbol{x} = (x_1, x_2, \dots, x_m)$
    - Bold face for vector
    - Sometimes start counting from 1, sometimes from 0
- Initially we consider classification as predicting a one-dimensional label: $\boldsymbol{x} \rightarrow y$
- Marsland (ch. 2, p. 15) considers the output as a vector $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$
- We will use vector when the output has more than one dimension

A word on notation

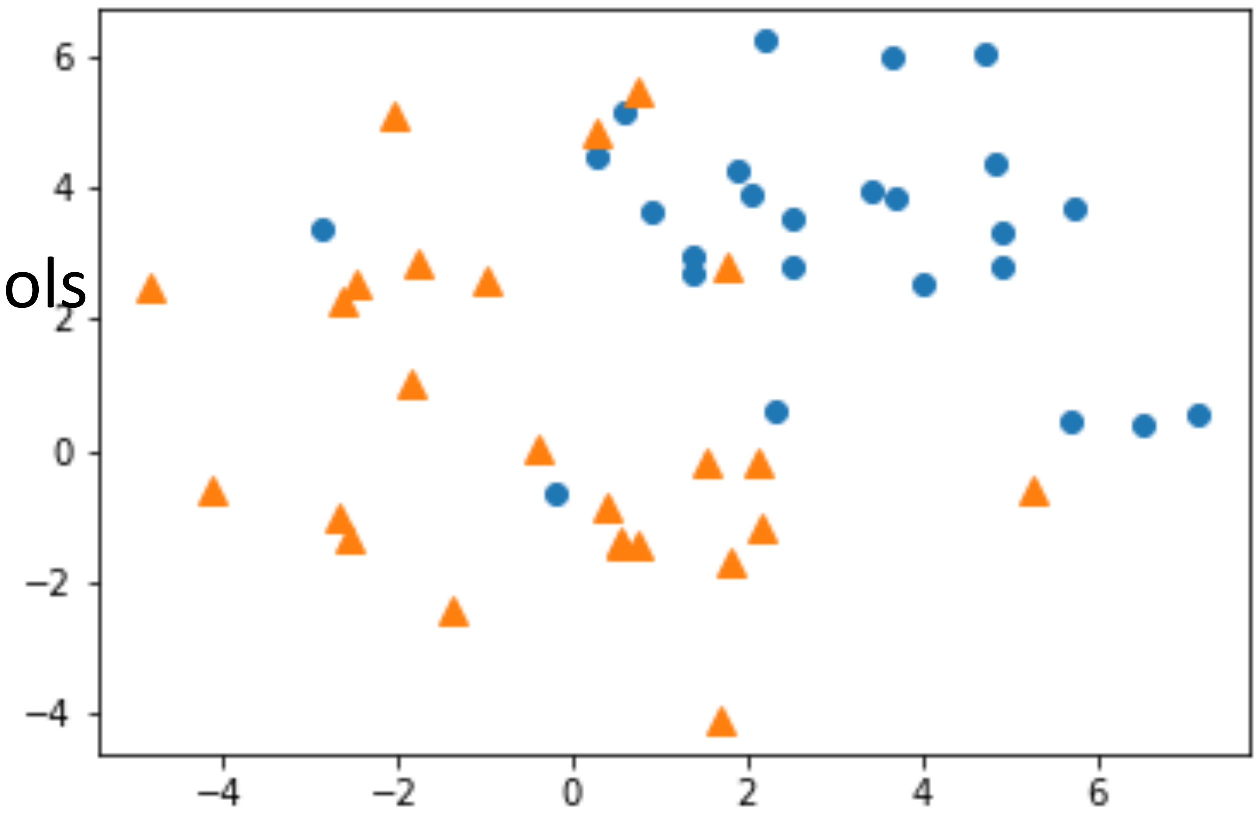
We will try to follow Marsland and use

- $\boldsymbol{x}_j = \left(x_{j,1}, x_{j,2}, \ldots, x_{j,m}\right)$ for input point number *j*
- t_j for a target value
- $y_j = f(\boldsymbol{x}_j)$ for a predicted value
- But beware that some use
  - y_j for the target value and
  - $\hat{y}_j = f(\boldsymbol{x}_j)$ for the predicted value (or even $\hat{y}_j = \hat{f}(\boldsymbol{x}_j)$)

ML algorithms and visualization

To illustrate ML-algorithms, it is easiest to use

- 2 numerical features
- show classes with colors, symbols (glyphs) or both



Multi-label classification

- Classify an object with respect to several binary classes
- E.g., *Who is in the picture?*
- (Uncle Tom: 1, Aunt Mary: 1, Grandma: 0, Grandpa: 0, etc.)

Supervised learning – two types

More on regression next week

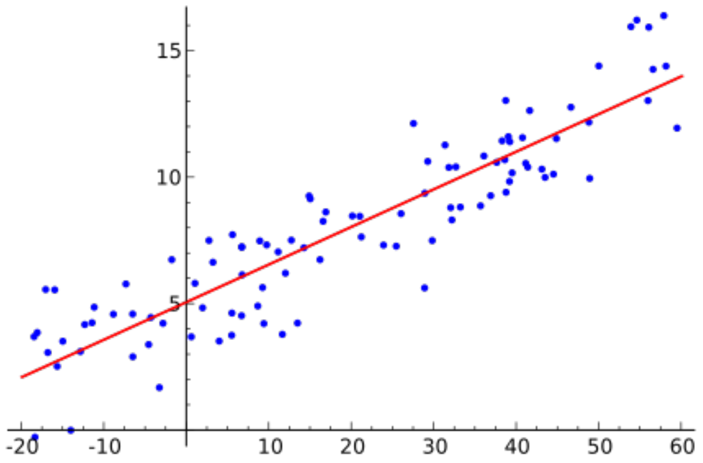
Classification

- Assign a label (class) from a finite set of labels to an observation

?

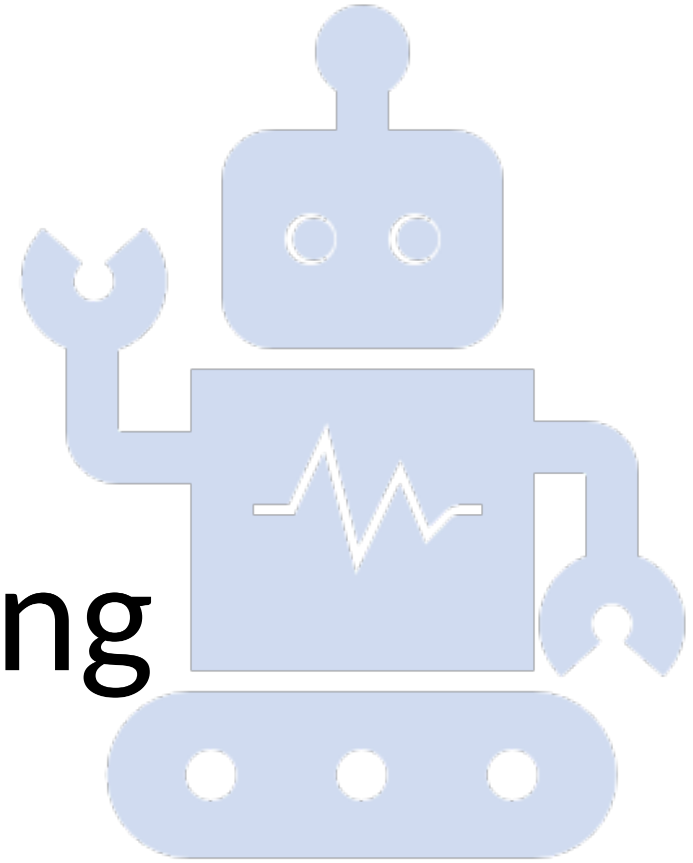
Regression

- Assign a numerical value to an observation
  - e.g., the temperature tomorrow



Important concepts

- Classifier, domain, label set
- Classifiers:
  - Binary
  - Multi-class classifier
- Multi-label classifiers
- Regression
- Features:
  - Categorical
  - Numerical



5.3 Supervised learning

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

Supervised Learning

- Structure:
  - a well-defined set of observations, O (possible inputs)
  - a set of label values, L (possible output values)
- Goal: determine a mapping , γ, from O to L
- A training set of examples from $O \times L$, $\left\{ \left(o_1, t_1\right), \left(o_2, t_2\right), \ldots, \left(o_n, t_n\right) \right\}$
- The training phase: Learn γ from the training set
  - Ideally, $\gamma\left(o_i\right) = t_i$ on the training data. = the "supervision" of the learning
- When γ is learned, it can be used to predict values for new items, $\gamma(o)$.

Supervised Learning

1. Feature extraction: We first decide on the features, their possible values, extract them from the observations, and replace each observation with its features, e.g.,

   o_j gets represented by $\boldsymbol{x}_j = \left(x_{j,1}, x_{j,2}, \ldots, x_{j,m}\right)$

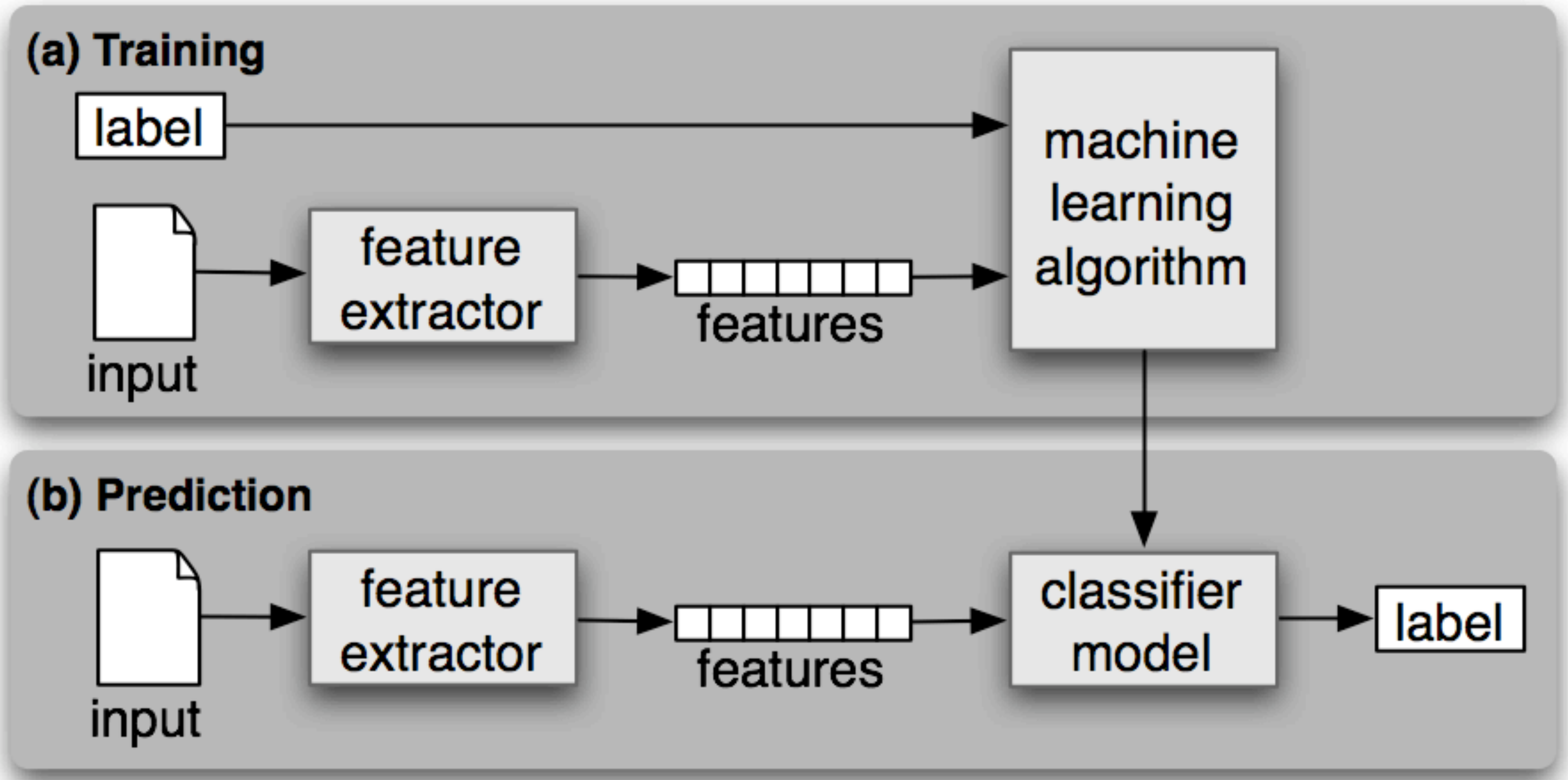
   - Revised goal: determine a mapping , f, from the set of features to L
   - A training set of examples from $\left\{\left(\boldsymbol{x}_1, t_1\right), \left(\boldsymbol{x}_2, t_2\right), \ldots, (\boldsymbol{x}_n, t_n)\right\}$

2. The training phase: Learn f from the training set

   - Ideally, $f\left(\boldsymbol{x}_j\right) = t_j$ on the training data. = the "supervision" of the learning

3. When f is learned, it can be used to predict values for new items, $f(\boldsymbol{x}')$

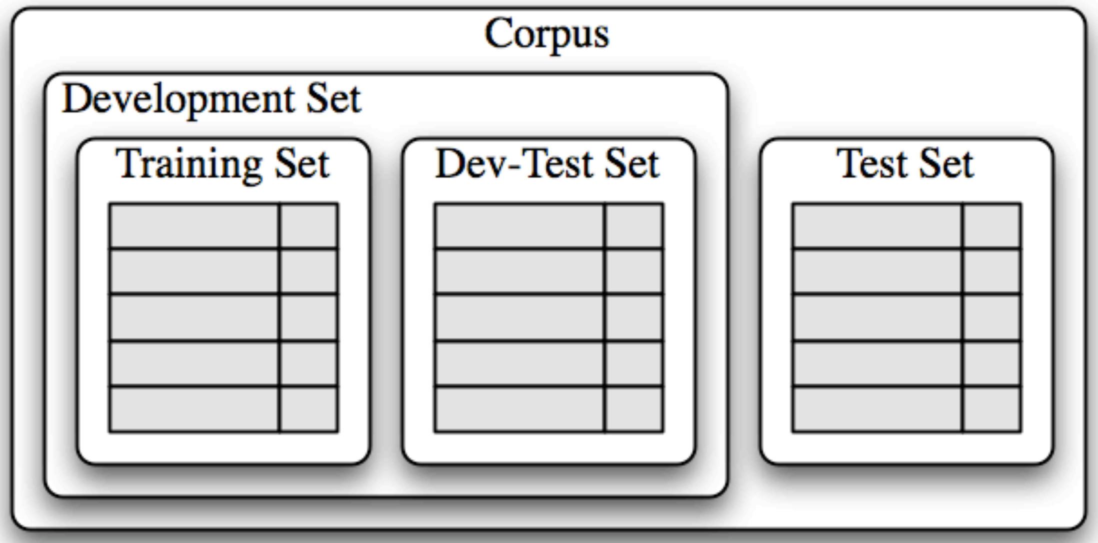
Classification



https://www.nltk.org/book/ch06.html

Training and test sets

- To measure improvement, we need (at least) two disjoint labeled sets:
    - Training set
    - Test set
- Train on the training set.
- Predict labels on the test set (after removing the labels)
- Compare the prediction to the given labels



https://www.nltk.org/book/ch06.html

- For repeated development we need (at least) two test sets.
    - One for repeated testing during development
    - One for final testing

Confusion matrix and accuracy

Goal: Evaluate our spam classifier

- We run the classifier on the labeled test set (without the labels)
- Compare the predicted labels to the example labels and count
- We can present the numbers in a confusion table

		True label	
		Yes	NO
Predicted label	Yes	tp=150	fp=50
	No	fn=100	tn=200

- True positives, tp=150
- False positives, fp=50
- False negatives, fn=100
- True negatives, tn=200
- Accuracy:

  `(tp+tn)/N = 350/500 = 0.7`

  - (Marsland (2.2) p.23 is wrong!)

More than two classes

		True label		
		spam	normal	urgent
Predicted label	spam	150	49	1
	normal	31	250	19
	urgent	19	31	50

Accuracy:

- (sum of the diagonal)/N
- $= \frac{\#\{y_i \mid y_i = t_i\}}{\#\{y_i\}} = \frac{450}{600} = 0.75$

Observe

- There is no consensus regarding what should be the columns and what should be the rows
- (Marsland p.22 does it differently from p. 23)

Important concepts

Steps:

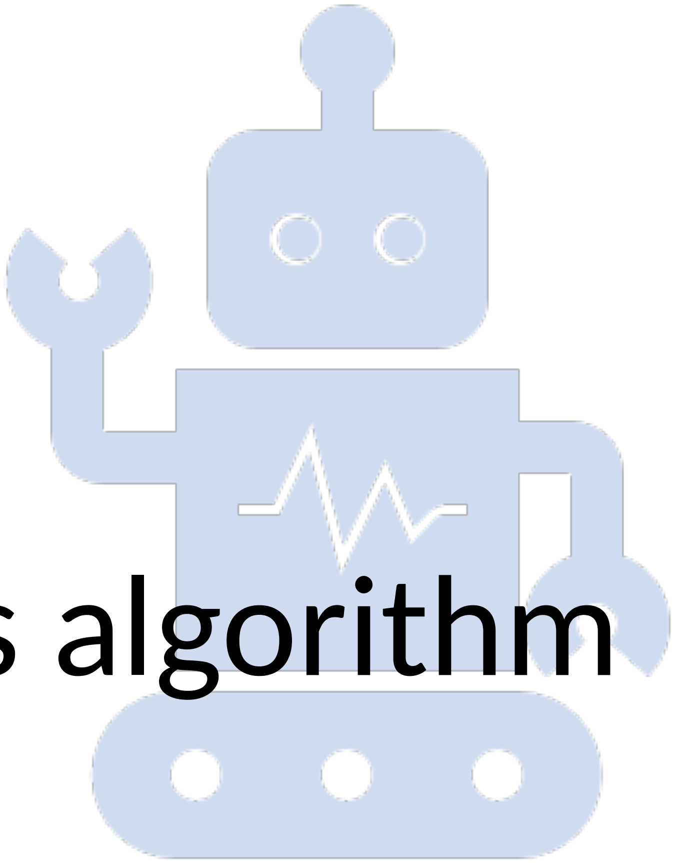
- Feature extraction
- Training on development data
- Predicting on new data

Split data:

- Development set:
  - Training set
  - Development test set
- Test set

Evaluation:

- Confusion matrix
- Accuracy



5.4 *k* Nearest Neighbors algorithm

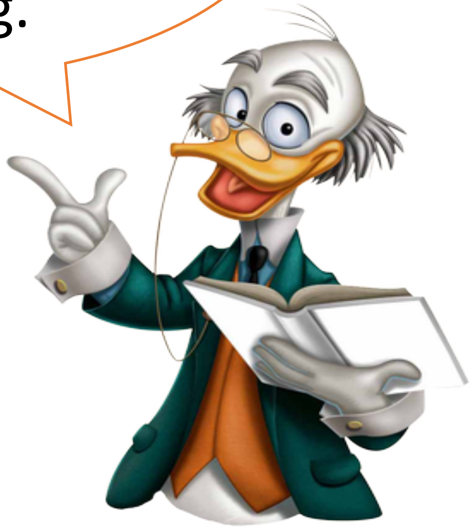
IN3050/IN4050 Introduction to Artificial Intelligence

and Machine Learning

Example:

Professor, do you think I will enjoy IN4050?

I can give you a scientific answer using machine learning.



Another student

I don't believe that professor
I rather ask the students who took the course last year

Source: http://pngimg.com/uploads/student/student_PNG165.png

- Ask, say, 7 students who took the course whether they liked it or not.
- Some *yes*, some *no*-s
- I trust most the students who resemble me most, e.g.:
  - Taken the same courses as me
  - Like the same courses as I do
- I pick the 7 students who resemble me most, and ask them
- I trust the majority vote of this group

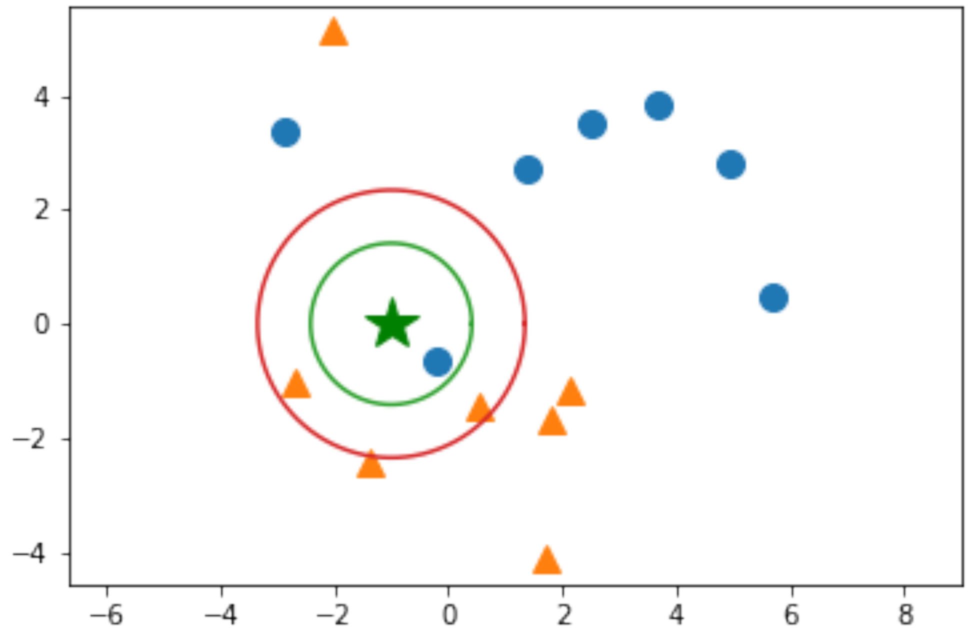
k Nearest Neighbors

I don't believe that professor
I rather ask the students who took the course last year

- This is an example of *k* Nearest Neighbor algorithm, with *k*=7
- Downside:
  - You have to consider every student who took the course last year to find the ones that resemble you most
  - Every student asking the question has to start the procedure from scratch

*k*NN algorithm

1. Calculate the distance to all the training instances
2. Pick the *k* nearest ones
3. Choose the majority class for this set



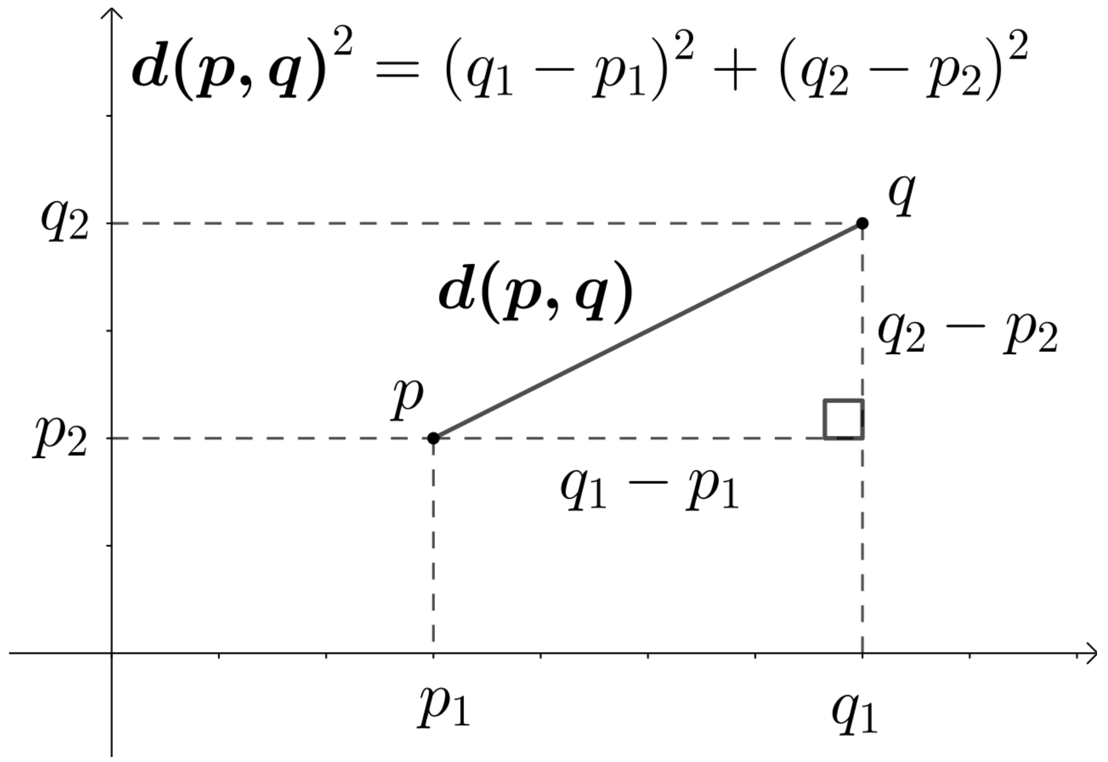
What is the result with

k = 1?

k = 3?

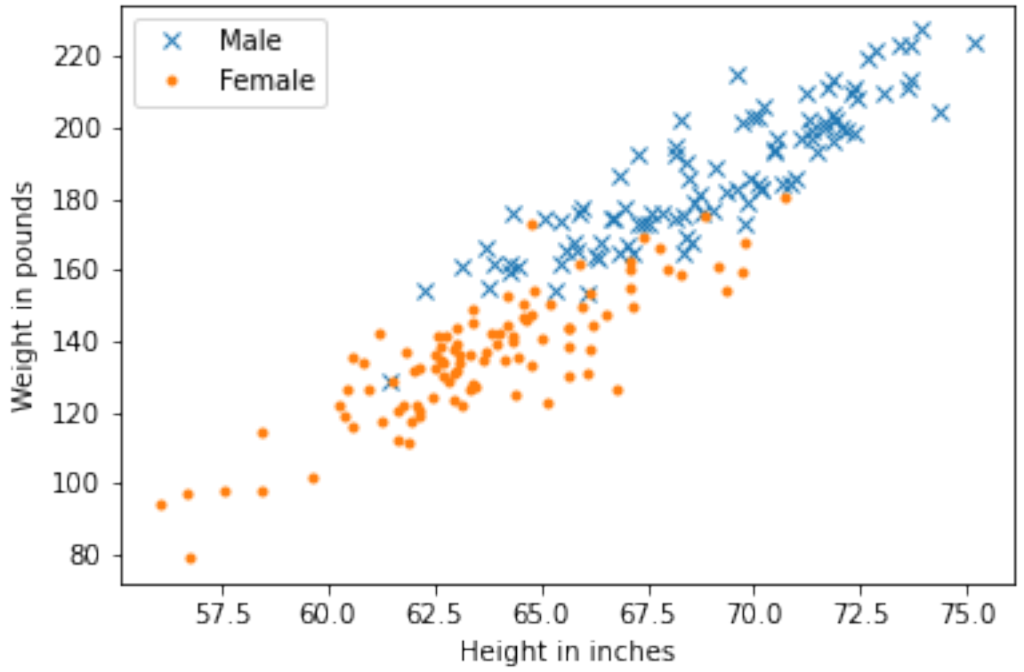
Distance

- Assumption:
  - Points that are close together in feature space are similar
- Some notion of distance in feature space, normally Euclidean distance (L2)



Example: Height, weight, gender

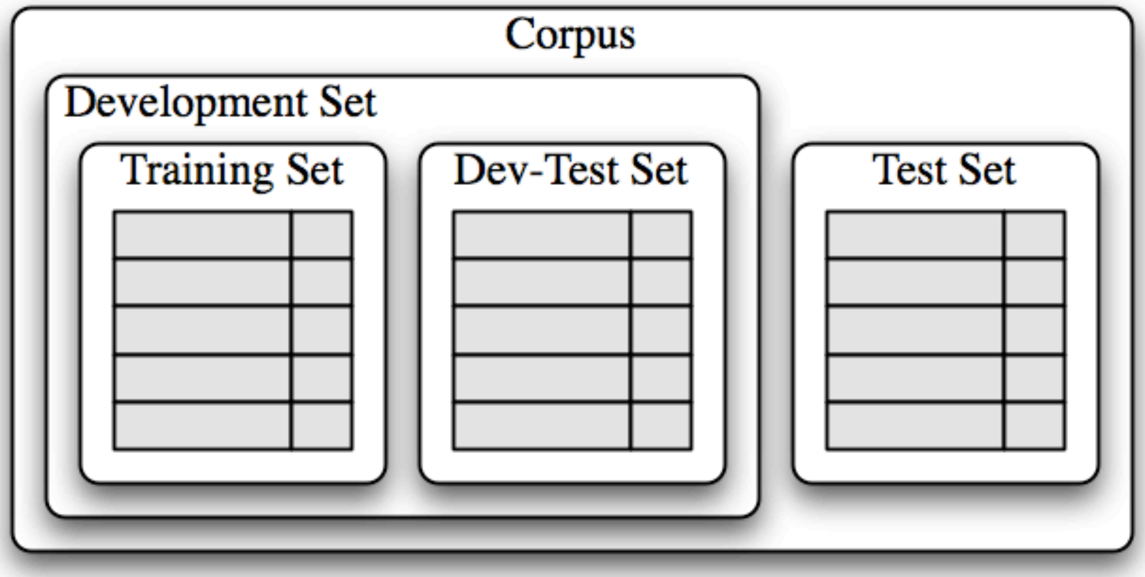
- Dataset:
  - 10,000 observations
  - 5,000 of each gender
  - Height in inches
  - Weight in pounds
- https://www.kaggle.com/mustafaali96/weight-height
- Processing:
  - Shuffled
  - Split:
    - 5000 for training
    - 2500 for development testing
    - 2500 for final testing



A random subset of 200 of the training data

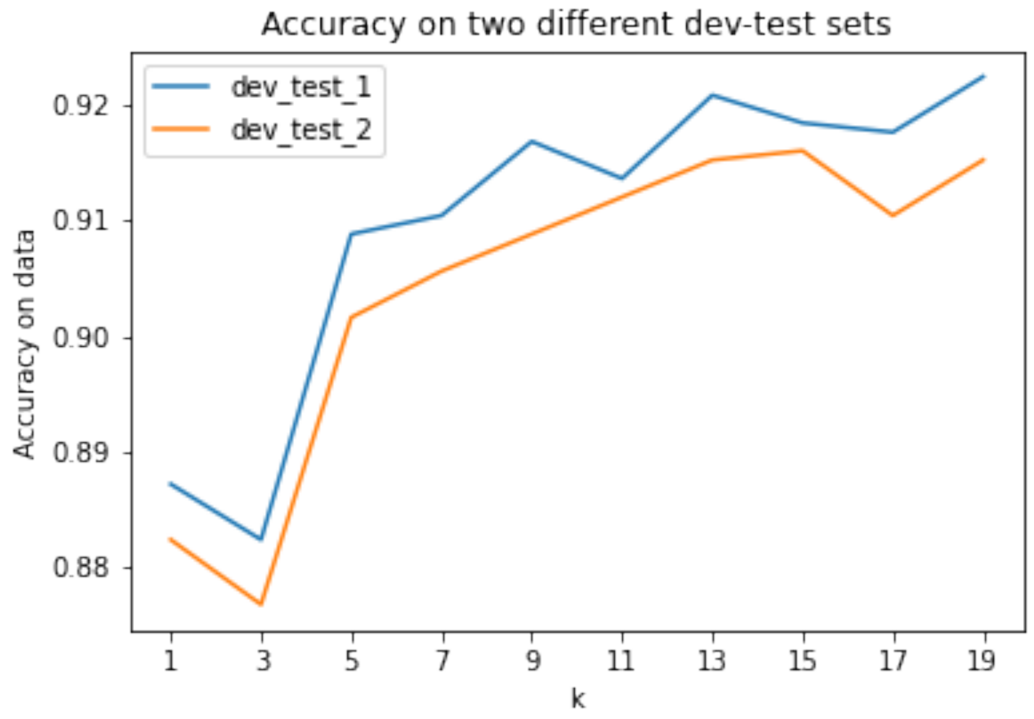
Choosing *k*

- For alternative values of *k:*
  - Train on the training set
  - Evaluate on the dev-test set
- Choose the *k* which yields the best accuracy
- You may test with this *k* on the final test set.



*k*NN with varying *k*

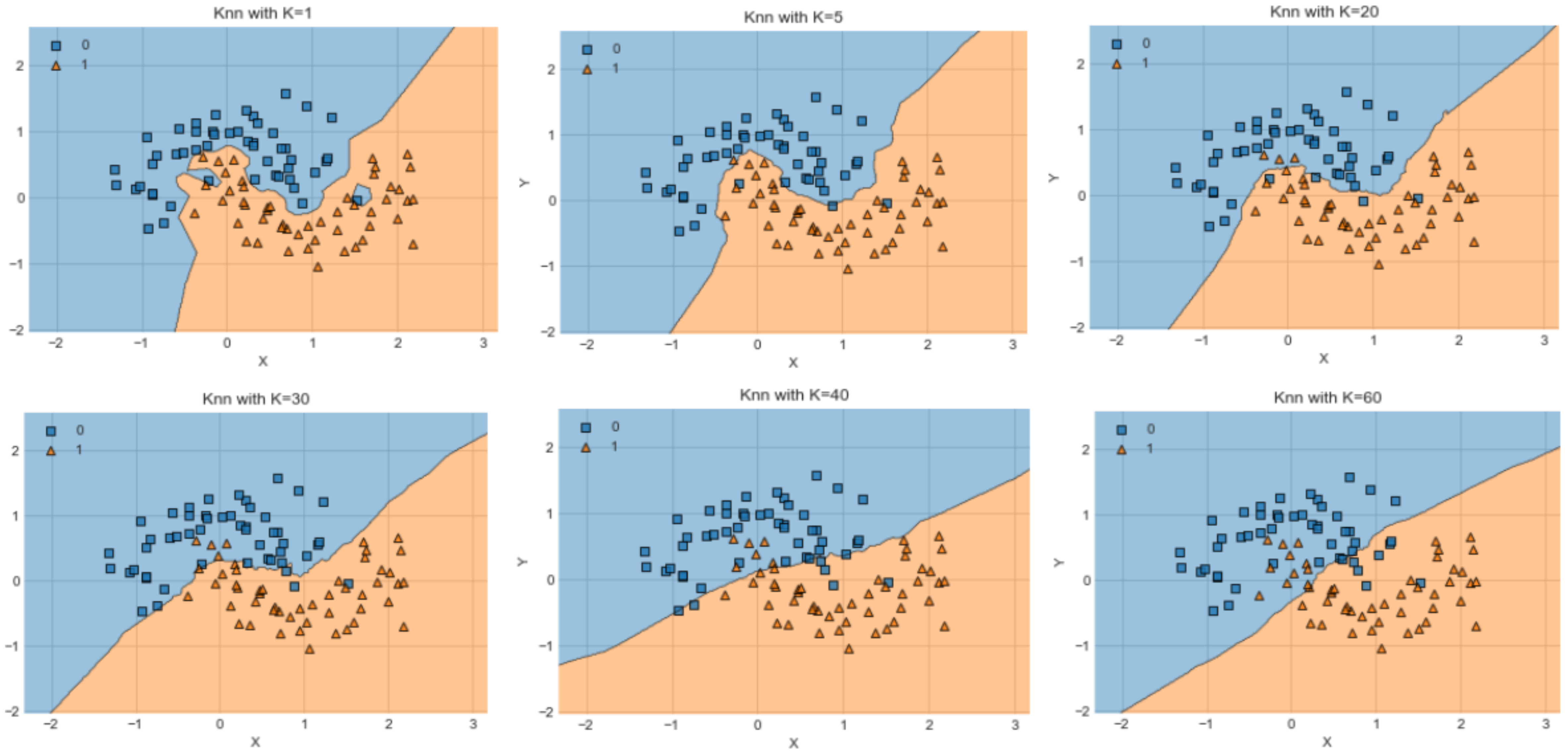
- Trained on the train set with various *k*'s
  - Accuracy on dev-test set
  - For this task: better with larger *k*
- Split the dev-test randomly in two equal parts
  - Same tendency
  - Not the exact same numbers:
    - To be expected
  - Numbers like accuracy will vary with test set



Is larger *k* always better?

- Small *k*:
  - Good fit to training data
  - Danger of overfitting
- Larger *k*:
  - More general

- Next slide:
  - The squares and triangles are the true classes
  - The background colors show the decision boundary for the classifier with various *k's*



KNN visualization for the U-shaped dataset

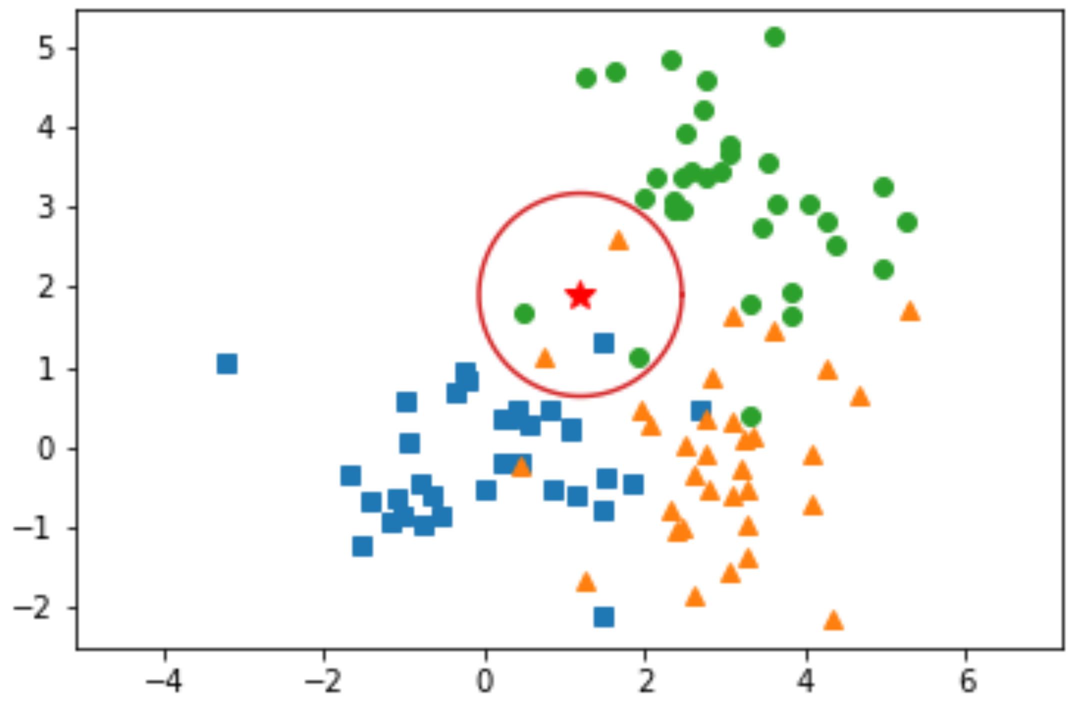
https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6

Properties of *k*NN

- Instance-based, no real training
  - (Fast to "train")
- Inefficient in predicting the label of new instances
  - Since it must consider all the training data
- One parameter: *k*
- The distance measure may influence the result
- The scaling of the axes might influence the result:
  - Next!

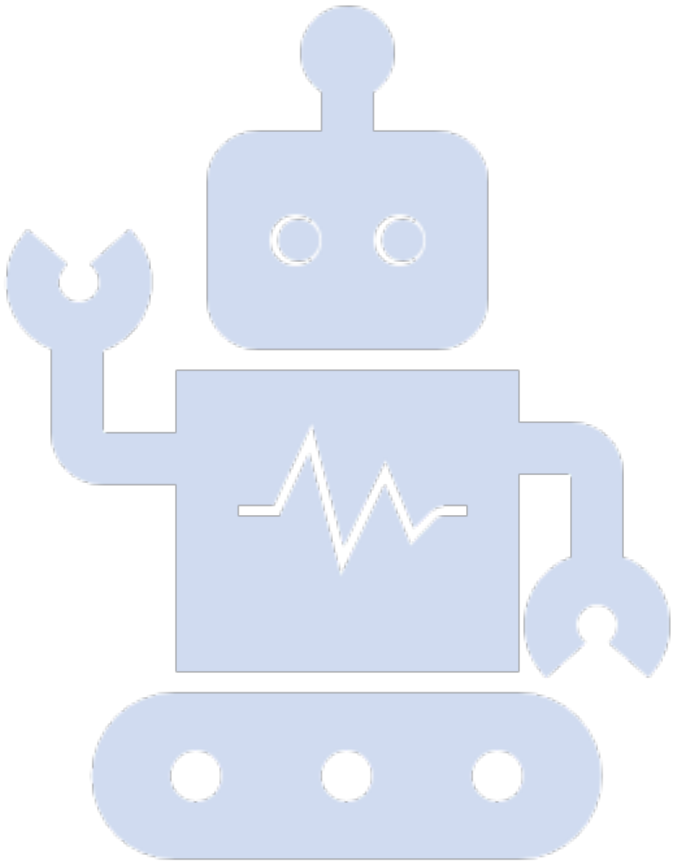
Footnote: More classes

- A binary classifier with odd *k* always reaches a decision
- With more than 2 classes, there might be a draw
- Possible ways out
  - Alt. 1: Back-off to *k*-1, etc. until there is a majority class
  - Alt. 2: Weight points by inverse distance from target, take the weighted max.



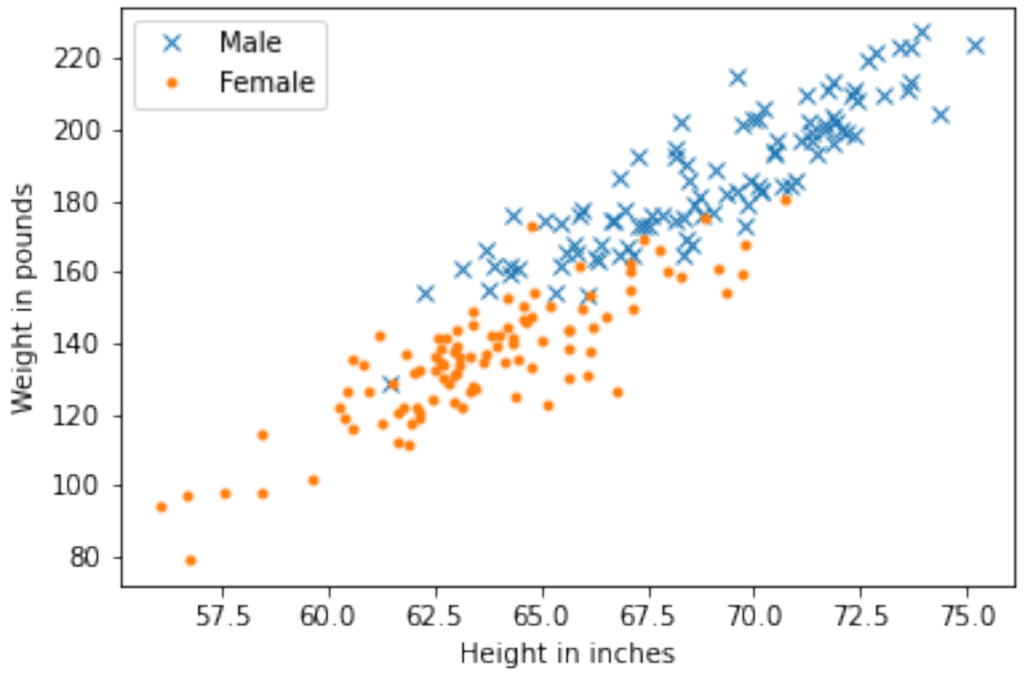
5.5 Feature scaling

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



Example: Height, weight, gender

- Dataset:
  - 10,000 observations
  - 5,000 of each gender
  - Height in inches
  - Weight in pounds
- https://www.kaggle.com/mustafaali96/weight-height
- Processing:
  - Shuffled
  - Split:
    - 5000 for training
    - 2500 for development testing
    - 2500 for final testing



A random subset of 200 of the training data

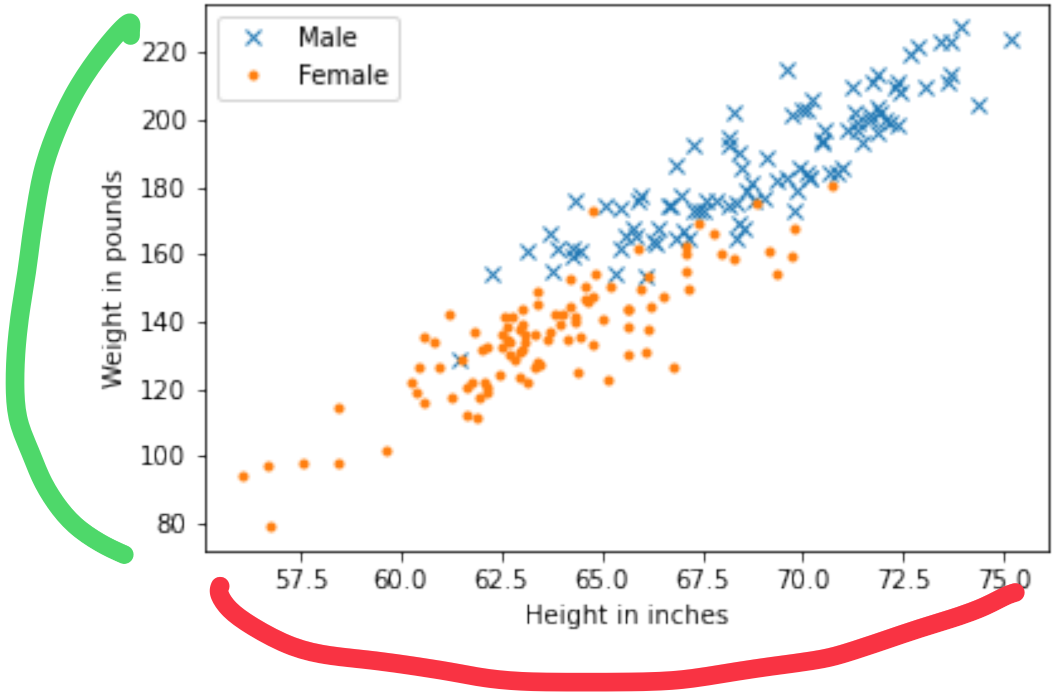
Example: Height, weight, gender

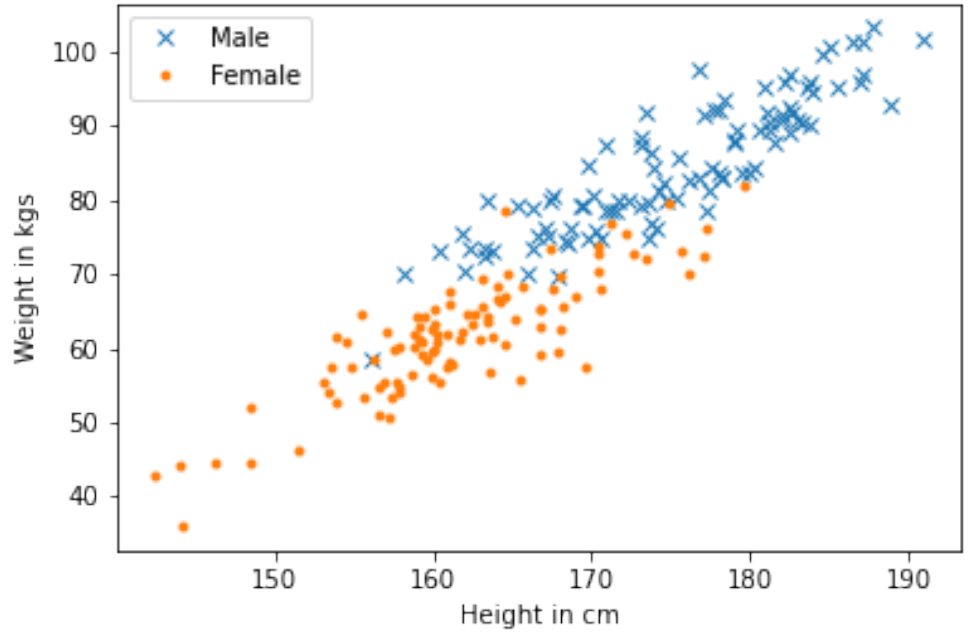
- Dataset:
    - 10,000 observations
    - 5,000 of each gender
    - Height in inches
    - Weight in pounds
- https://www.kaggle.com/mustafaali96/weight-height
- Processing:
    - Shuffled
    - Split:
        - 5000 for training
        - 2500 for development testing
        - 2500 for final testing

Data format	Accuracy
Inch-pounds	0.905
Cm-kilos	0.897

- We converted to cm and kgs using the same splits
- Trained and tested a 3NN-model
- Different results
- How come?

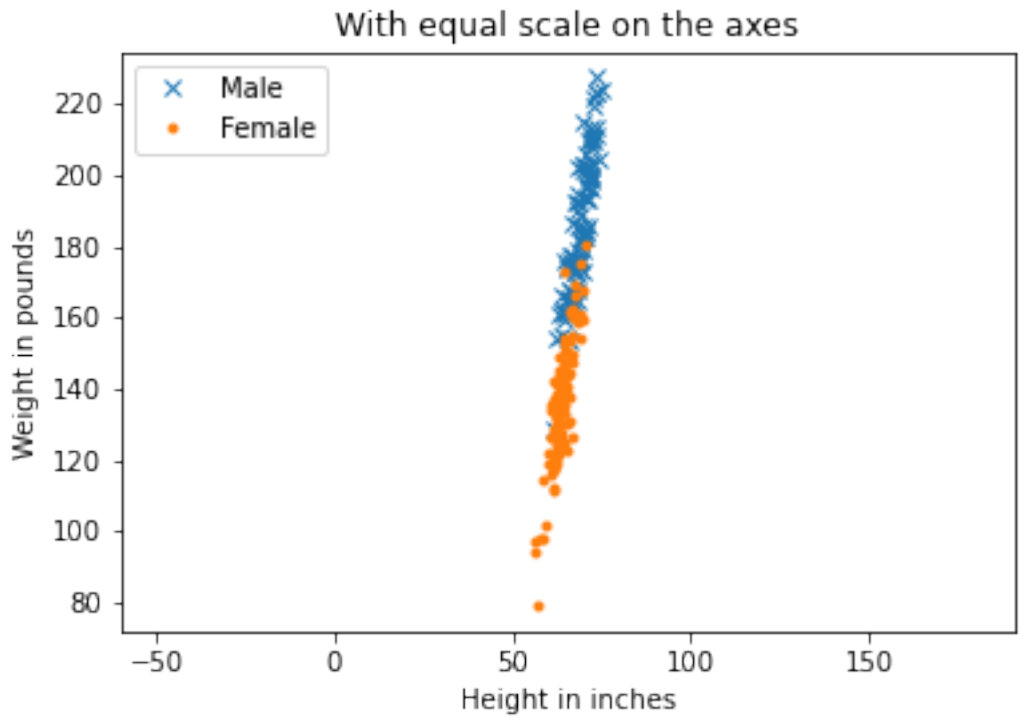
How come?

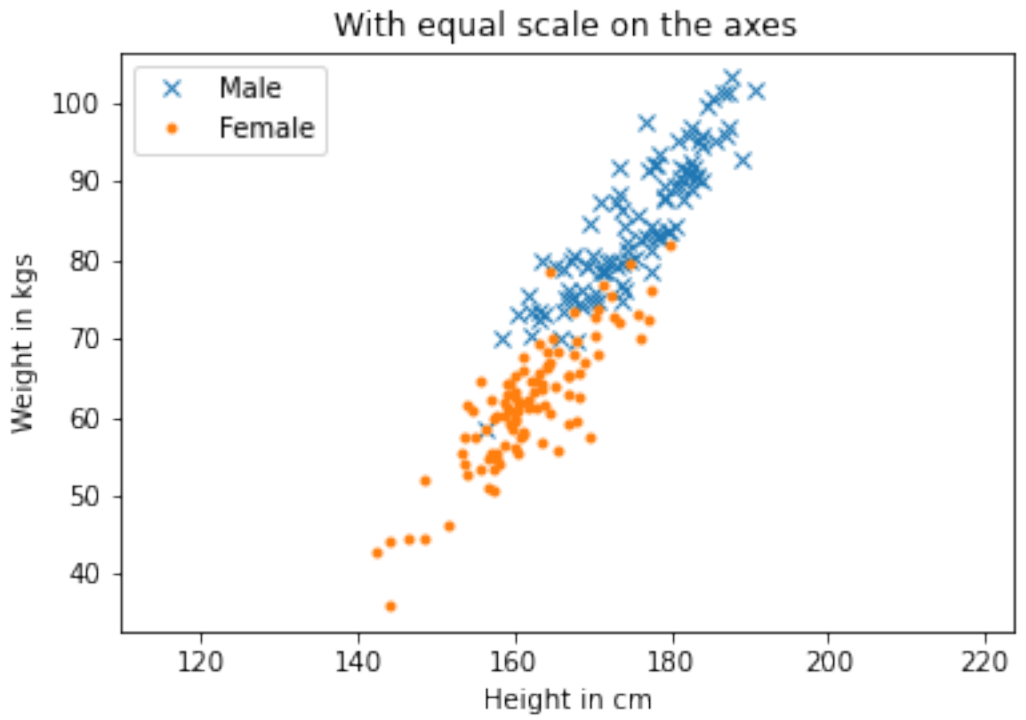




- The same random subsets of the training data
- Look the same

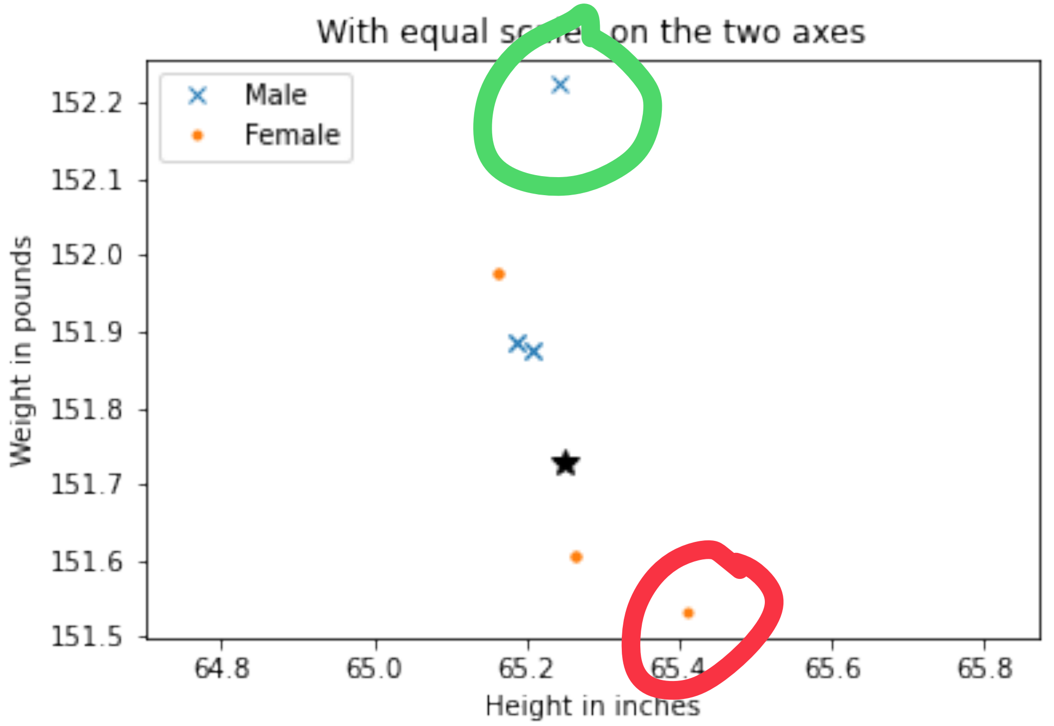
Using equal scale at the two axes

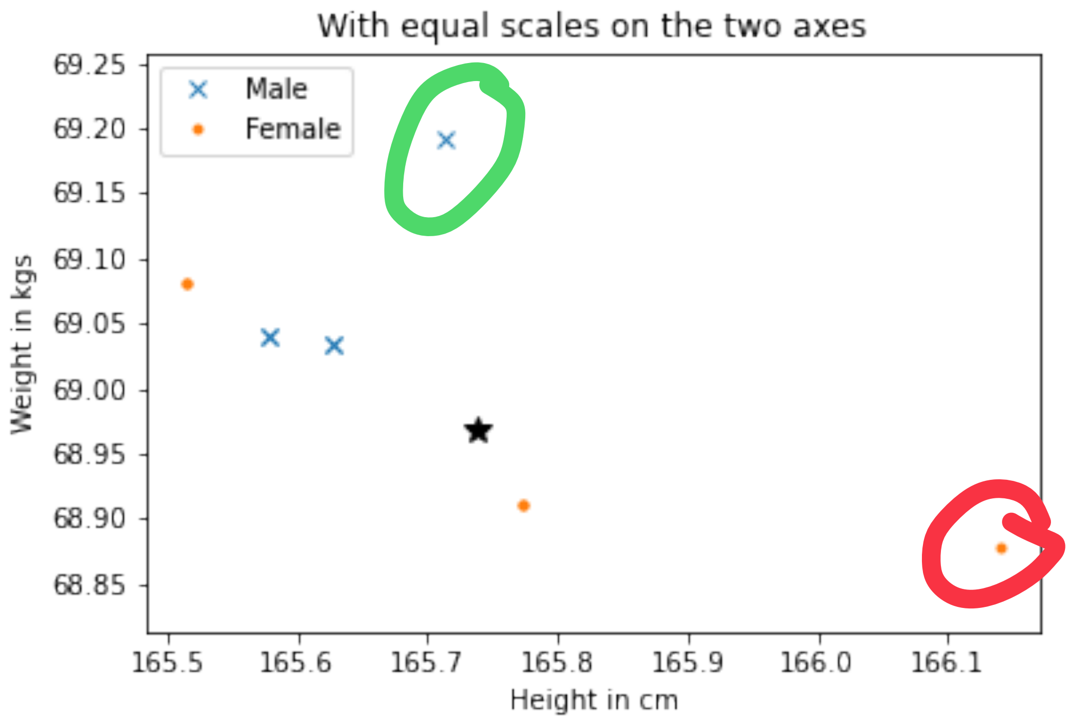




- These are the spaces in which we calculate the distance to the nearest neighbors

Using equal scale at the two axes





- Shows the same target and 6 training points in the two
- The 6 training points include the 5 nearest neighbors for each of them

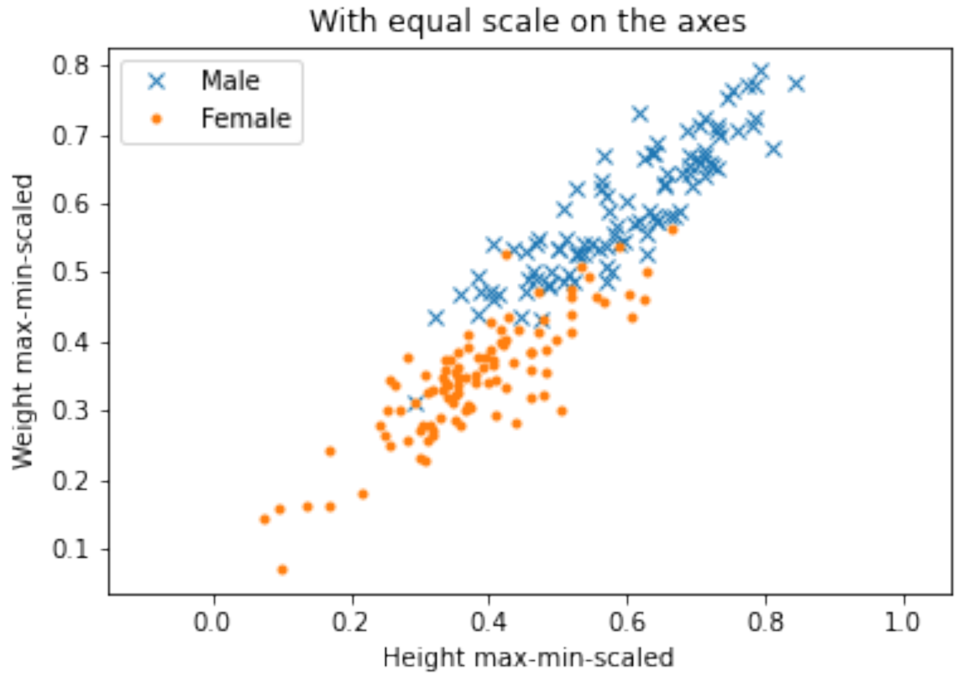
Scaling

- Observation:
  - The result of *k*NN depends on the scale we use on the feature dimensions.
- How do we determine the ``correct'' (i.e., best) scales?
  - We could experiment with various alternatives
- Another option is scaling

- The input data should:
  - have similar range for the various input variables
  - typically, be in the range (0,1) or (-1, 1) or thereabouts
- Several ways of scaling, most common
  - Max-min scaler
  - Normalization (standard scaler)

Max-min-scaling

- Training set: N data points, of the form:
  - $\boldsymbol{x}_j = \left(x_{j,1}, x_{j,2}, \ldots, x_{j,m}\right)$
    - m is the number of features
    - $x_{j,i}$ is the value of feature i for observation j
- For each feature $i = 1, \ldots m$:
  - Let
    - $min_i = \min(\left\{x_{j,i} \middle| j = 1,2, \ldots, N\right\})$
    - $max_i = \max(\left\{x_{j,i} \middle| j = 1,2, \ldots, N\right\})$
  - Define $scale_i(x_{j,i}) = \dfrac{x_{j,i} - min_i}{max_i - min_i}$
- This will map all features in the training set to numbers in the interval [0, 1]



Test set

- It is not a good idea to normalize the dataset before splitting it into training and testing (as Marsland will have it, p. 64):
  - Do not touch your test data
  - The goal is to apply the system also to new unseen data beyond the test data
- Instead:
  1. Use the training data to determine the scaler/normalizer
  2. Scale the training data before training
  3. Whenever you apply the system: scale the data before they are given to predict, using the same scaler as on the training data

Normalizing (an alternative scaler)

- Sometimes called standard scaler
- Find the
  - mean $\mu_i = \text{mean}\left(\left\{ x_{j,i} \middle| \ j = 1,2, \ldots, N \right\}\right),$ and
  - standard deviation $\sigma_i = \text{std}(\left\{ x_{j,i} \middle| \ j = 1,2, \ldots, N \right\})$
  - of your training set.
- Define $scale_i(x_{j,i}) = \dfrac{x_{j,i} - \mu_i}{\sigma_i}$ for each feature $i = 1, \ldots m$
- Apply it to your training data and to the data to which you apply the system.

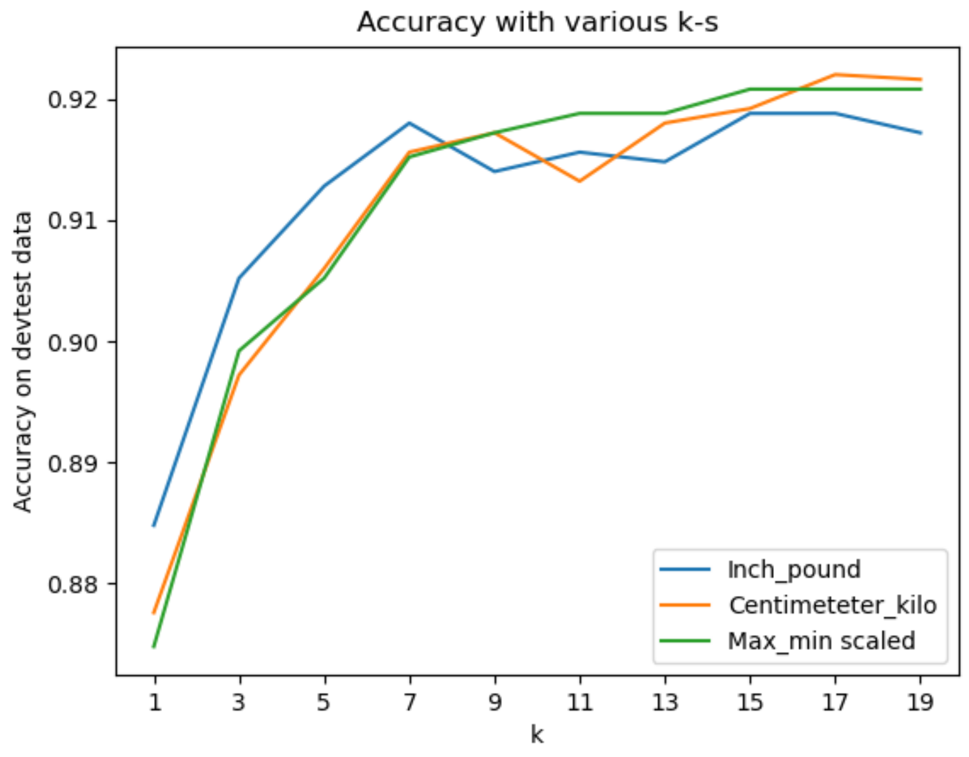
Scaling

- For some algorithms, scaling influences how items are classified, cf. *k*NN

- For other algorithms, the scaling
  - does not alter the result as the algorithm determine how much weight to put on various features
  - But scaling can be import for efficiency

Results on the weight-height data

- In this case the scaler does not give better results.
- No guarantee for giving better results



Take home

- The scale of the axes may influence the result of *k*NN
- Scaling can be important for efficiency for some classifiers
- Max-min-scaling
- Standard scaler
- Construct scaler from training set alone