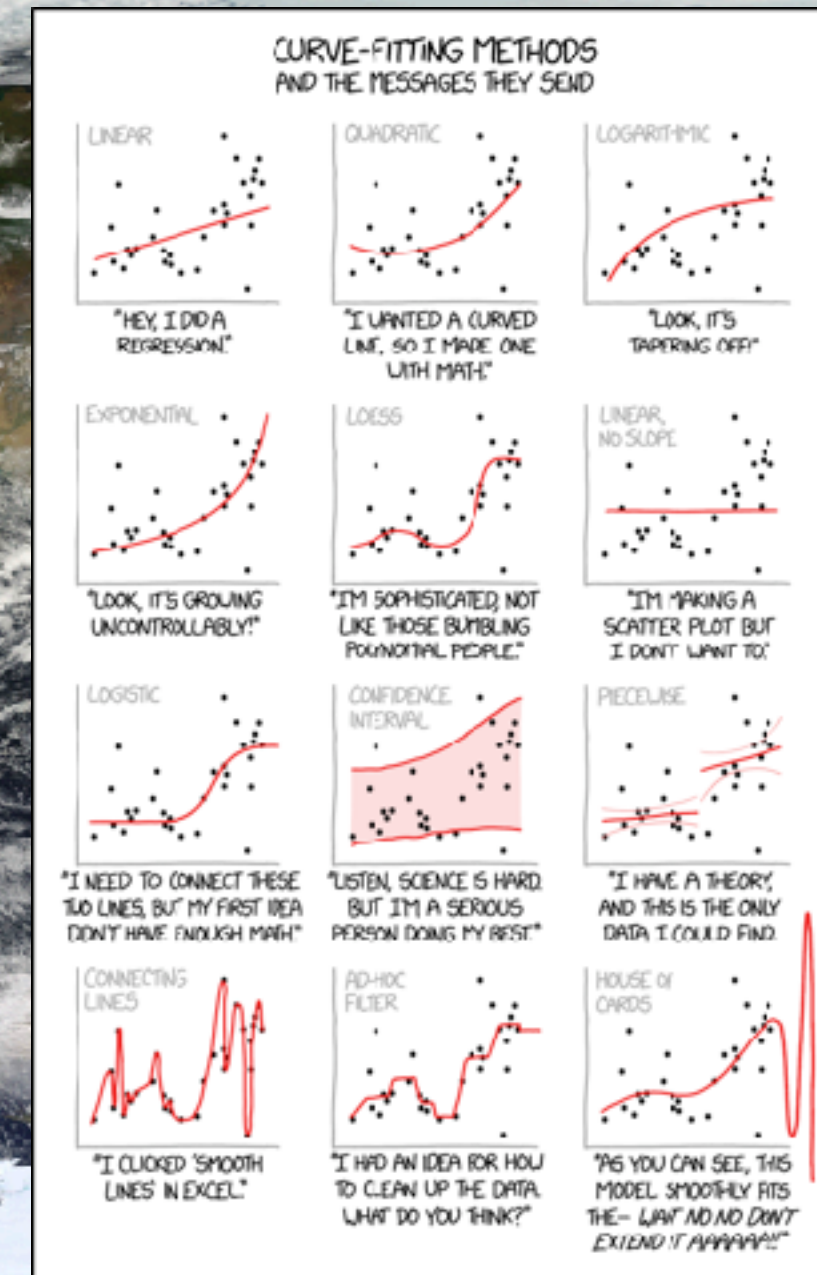


Earth Data Sciences

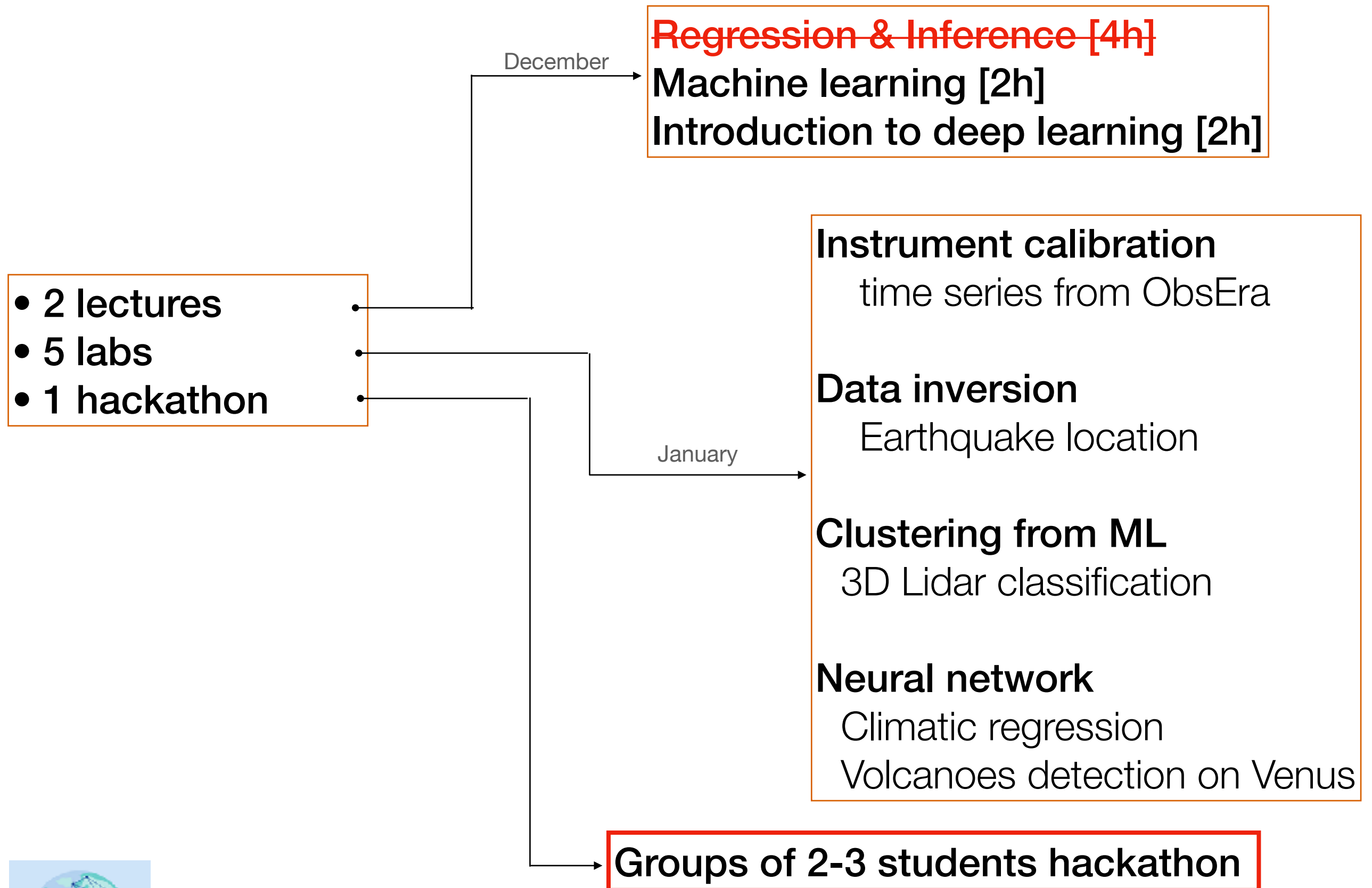


Antoine LUCAS (lucas@ipgp.fr)
Alexandre Fournier
Grégory Sainton
Léonard Seydoux
Éléonore Stutzmann



This mosaic is based on data observations from the *Moderate Resolution Imaging Spectroradiometer* (MODIS)

Organigram





Antwan's slack



- Fils de discussion
- Messages directs
- Mentions et réactions
- Brouillons et envoyés
- Slack Connect
- Plus

Canaux

eds

général

master_risqnat

tutored_project

+ Ajouter des canaux

Messages directs

Slackbot

Antoine vous

Canal

eds

+ Ajouter un marque-page

Ceci est le tout début du canal # eds
Vous avez créé ce canal le 25 novembre.
This is the Earth Data Science Slack channel [Modifier la description](#)

[Ajouter des personnes](#)

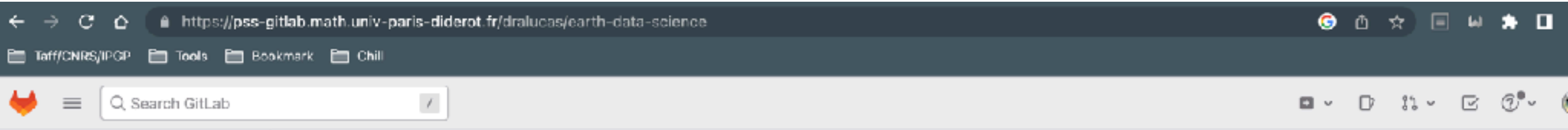
[Transférer les e-mails sur ce canal](#)

Vendredi 25 novembre

Antoine 13 h 30
a rejoint #eds, avec 2 autres personnes.

Antoine 13 h 30
a défini la description du canal : This is the Earth Data Science Slack channel

Lundi 28 novembre

Antoine Lucas › Earth Data Science

- 

Earth Data Science

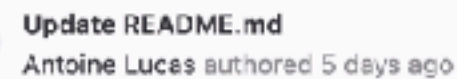
Project ID: 134 

 Star 0 Fork 0

 49 Commits 1 Branch 0 Tags 69.9 MB Project Storage

earth-data-science /

Web IDE

Clone 

874153b2









 README

MIT License

[⊕ Add CHANGELOG](#)

 Add CONTRIBUTING

 Configure Integrations

Name	Last commit	Last update
 Lectures	Upload New File	1 week ago
 Preparation	Update EDS_Python_Intro2Pandas.ipynb	10 months ago
 Useful	Upload New File	5 days ago
 binder	Add new file	11 months ago
 LICENSE	Update LICENSE	11 months ago
 Presentation_UE_EarthDataSci...	Upload New File	8 months ago

EDS_Python_Positionning_v1_1_no_sol.ipynb 14.67 KiB

</> 📄 Edit ⌵ Replace Delete 📄 📄 📄

Python: positionning test or refreshing exercises

G. Sainton (2020)

What are we talking about:

This notebook should allow you to evaluate your skills and your knowledges in Python and in particular your mastery of the **Numpy** and **Matplotlib** libraries which will be the basis of this [Earth]-Data Science course.

The following exercices are supposed to be fully understood to really take advantages of this data science lectures. If it's not the case, no problem, just go through this notebook.

It is impossible to cover all the fonctionnalités of each library. We just give very few examples. If you need help, besides the traditionnal "call a friend" "50/50" and "The public", one can use:

1. Official documentation of the library
2. Online help: ie -> print?,
3. Stack overflow
4. ...

Solutions will be given later... Training first

Table of content :

1. Few elements of Lists and Tuples
2. A bit of numpy
3. Matplotlib in a nutshell
4. Next step

Earth Data Science

- Project information
- Repository
- Issues 0
- Merge requests 0
- Security & Compliance
- Deployments
- Packages and registries
- Infrastructure
- Monitor
- Analytics
- Wiki
- Snippets
- Settings

EDS_Python_Intro2Pandas.ipynb 72.65 KiB

</> 📄 Edit Replace Delete 📄 📄 📄

Pandas: a convenient library for data science

In this notebook, it's out of questions to reinvent what has been already done thousands of times. Nevertheless, one can focus on the useful commands which will be used during the *Earth Data Sciences* practical activities.

Pandas is part of the *anaconda* toolbox. So if it's already installed, you have nothing to do but importing the Pandas library just writing:

In [1]:

```
import pandas as pd
```

Pandas is made of two core objects : The **Dataframes** and the **Series** Let's see that in details.

DataFrame

Roughly, DataFrame object is a table ie, an array with some entries made of values

In [2]:

```
myfirstdataframe = pd.DataFrame({"Hero": ["SpiderMan", "Wonder Woman", "Iron Man"], "Hair": ["Brown", "Blond", "Black"]}, )
display(myfirstdataframe)
```

Out [2]:

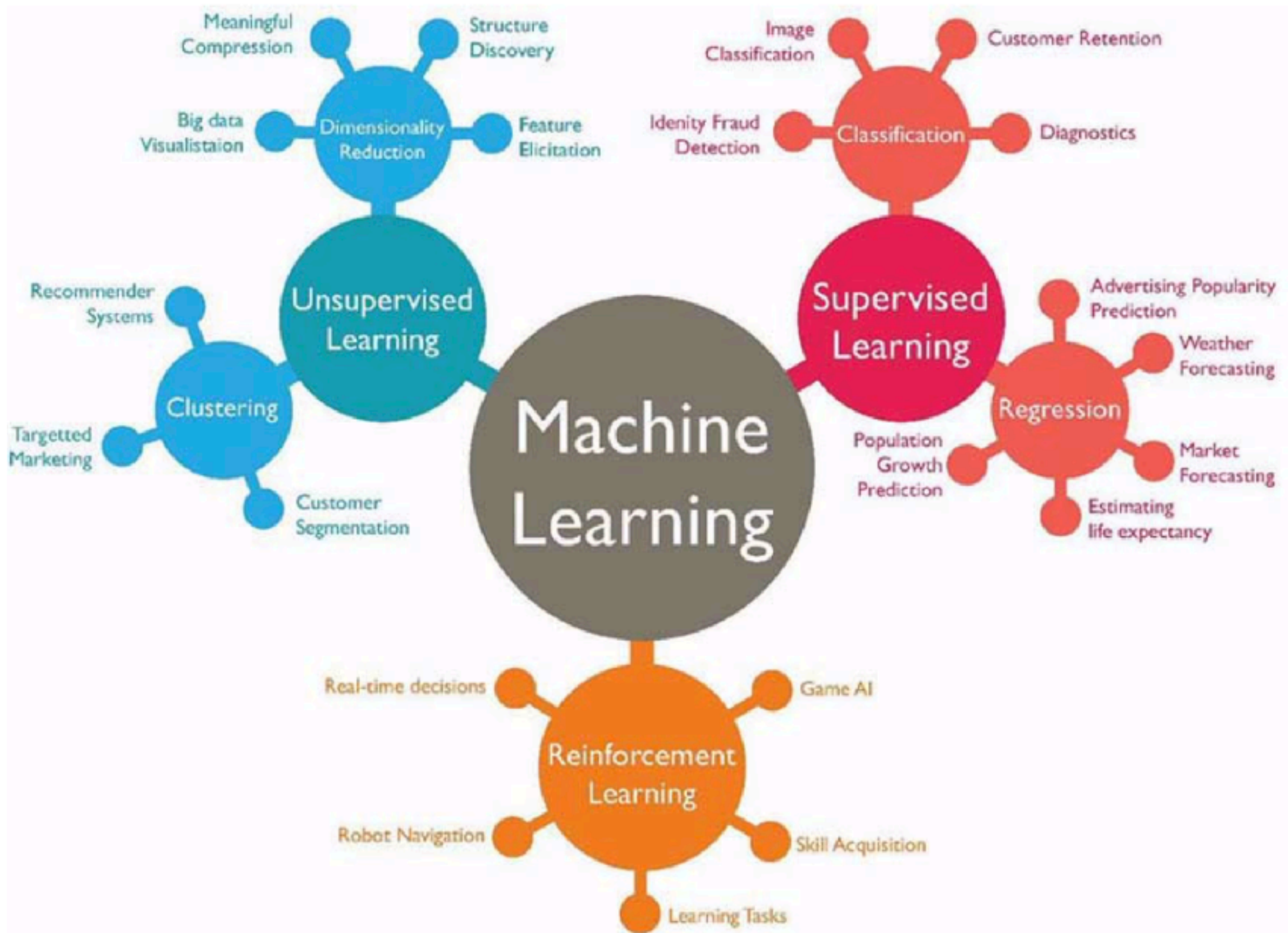
	Hero	Hair
0	SpiderMan	Brown
1	Wonder Woman	Blond
2	Iron Man	Black

Of course, entries can be either strings, integers, floats...

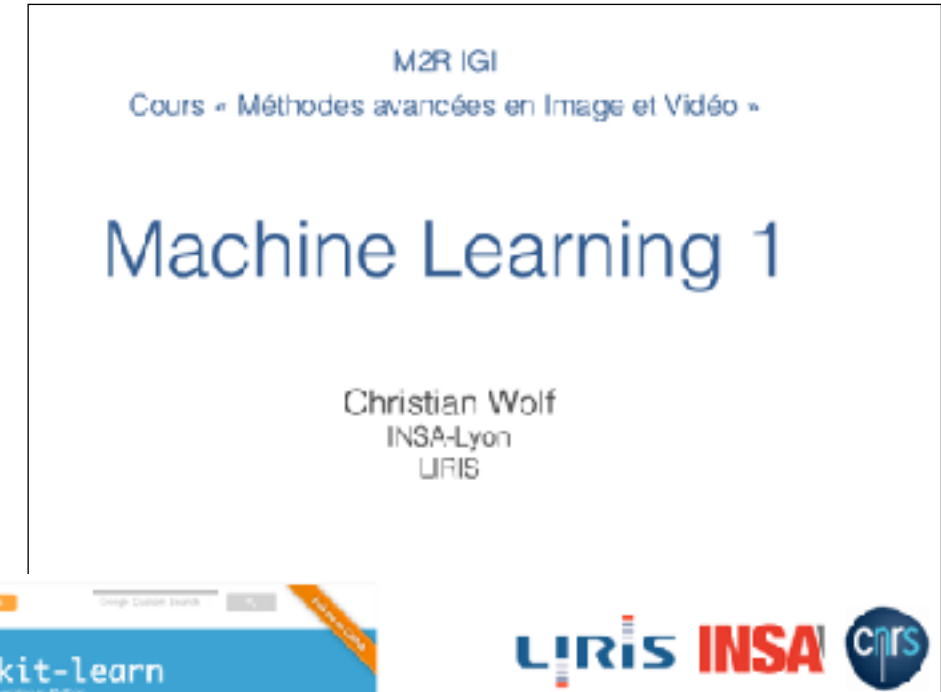
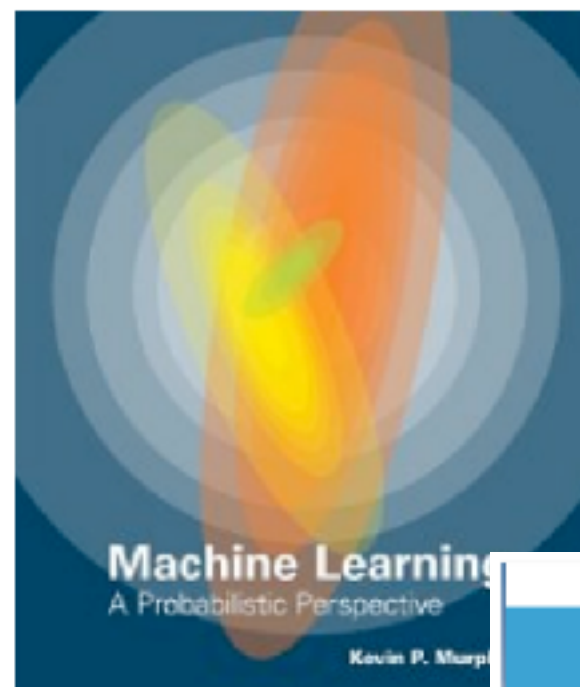
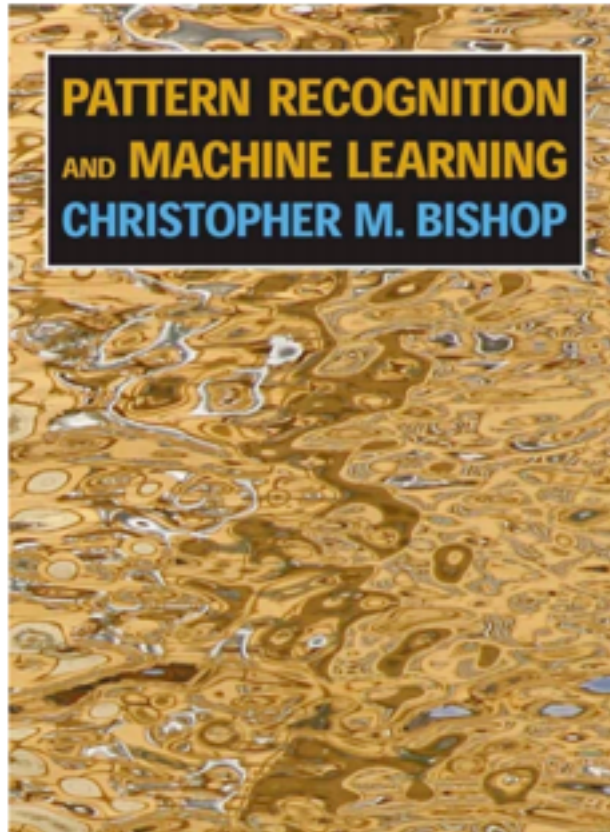
Lecture 2

Classification techniques are an essential part of machine learning and data mining applications.

Approximately 70% of problems in Data Science are classification problems.



Sources:



<https://www.3blue1brown.com/>

[https://fw.ipgp.fr/rhs0024/wiki/index.php/Atelier Machine et Deep learning%2C 20 mai 2019](https://fw.ipgp.fr/rhs0024/wiki/index.php/Atelier_Machine_et_Deep_learning%2C_20_mai_2019)

www.datacamp.com

<https://thedatafrog.com/>

<https://machinelearningmastery.com/>

<http://wikistat.fr/>

towardsdatascience.com

<http://perso.univ-lemans.fr/~berger/CoursTF/CoursTF>



Lecture 2

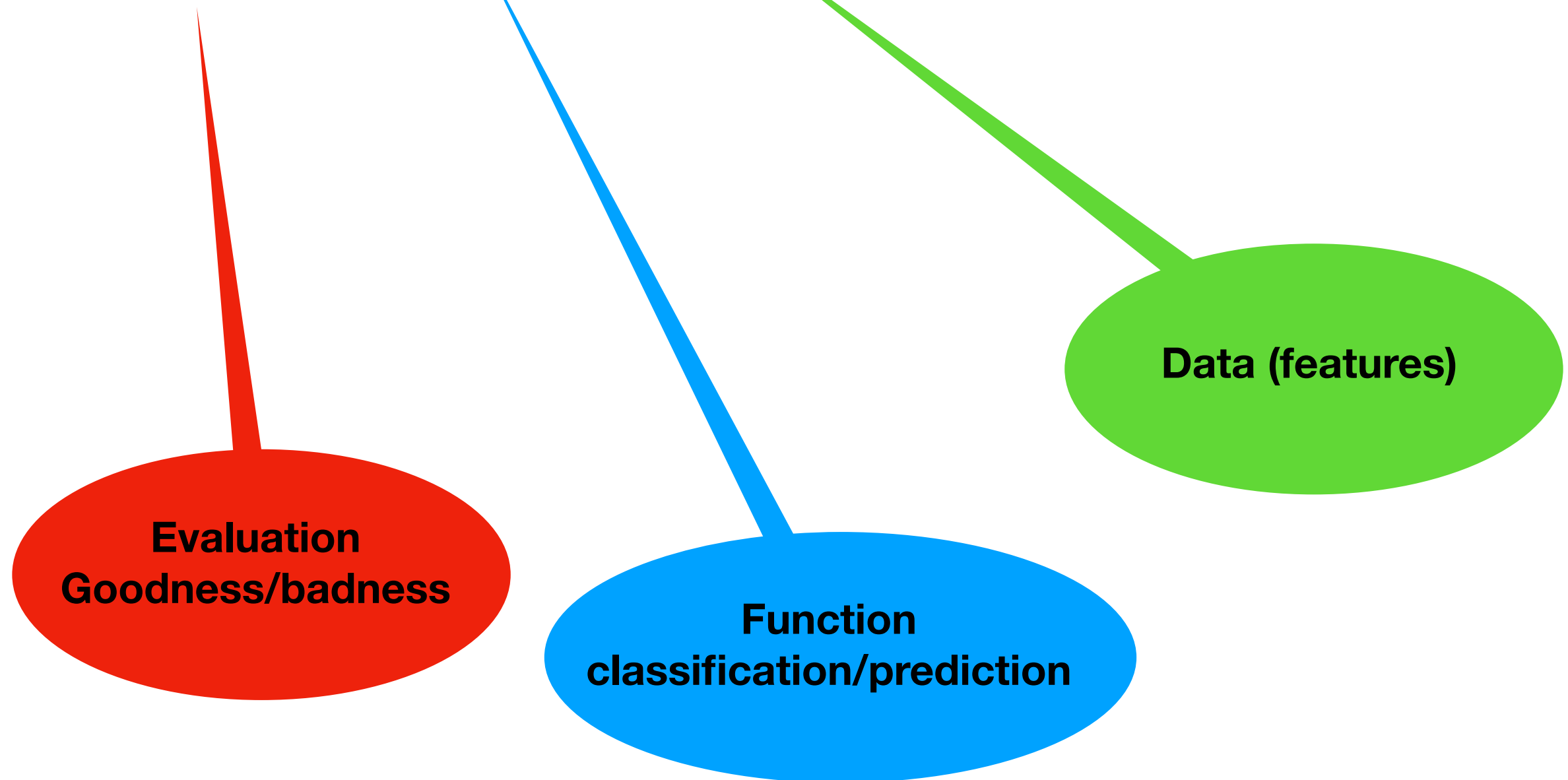
Lecture 2.1 Machine learning

Lecture 2.2 Clustering and the like

Lecture 2.3 Deep Learning with neural networks

Learning to perform a Task from Experience (like humans, animals) with a performance measure

Learning to perform a Task from Experience (like humans, animals) with a performance measure



Learning to perform a Task from Experience (like humans, animals) with a performance measure

Modelling the function that best represents the relationship between input data x_i and outputs y_i , i.e. a model of the data

Learning to perform a Task from Experience (like humans, animals) with a performance measure

Modelling the function that best represents the relationship between input data x_i and outputs y_i , i.e. a model of the data

- Prediction (task): x_i =features, y_i =output function (continuous, regression)
- Classify (task): x_i =features, y_i =output classes (discrete, categories)

Learning to perform a Task from Experience (like humans, animals) with a performance measure

Modelling the function that best represents the relationship between input data x_i and outputs y_i , i.e. a model of the data

- Prediction (task): x_i =features, y_i =output function (continuous, regression)
- Classify (task): x_i =features, y_i =output classes (discrete, categories)

Two techniques: supervised or unsupervised

Several algorithms = methods to model

How much better? A cost function and an optimization algorithm ("best")

Prediction

“La prévision est difficile surtout lorsqu'elle concerne l'avenir.”
Pierre Dac

Prediction

“La prévision est difficile surtout lorsqu'elle concerne l'avenir.”
Pierre Dac

In general, we would like to predict a t -value from an observation x

$$t = y(x, w)$$

Prediction

“La prévision est difficile surtout lorsqu'elle concerne l'avenir.”
Pierre Dac

In general, we would like to predict a t -value from an observation x

$$t = y(x, w)$$

If t is continuous: **Regression** ➡ Lecture 1

Prediction

“La prévision est difficile surtout lorsqu'elle concerne l'avenir.”
Pierre Dac

In general, we would like to predict a t -value from an observation x

$$t = y(x, w)$$

If t is continuous: **Regression** ✓ *Lecture 1*

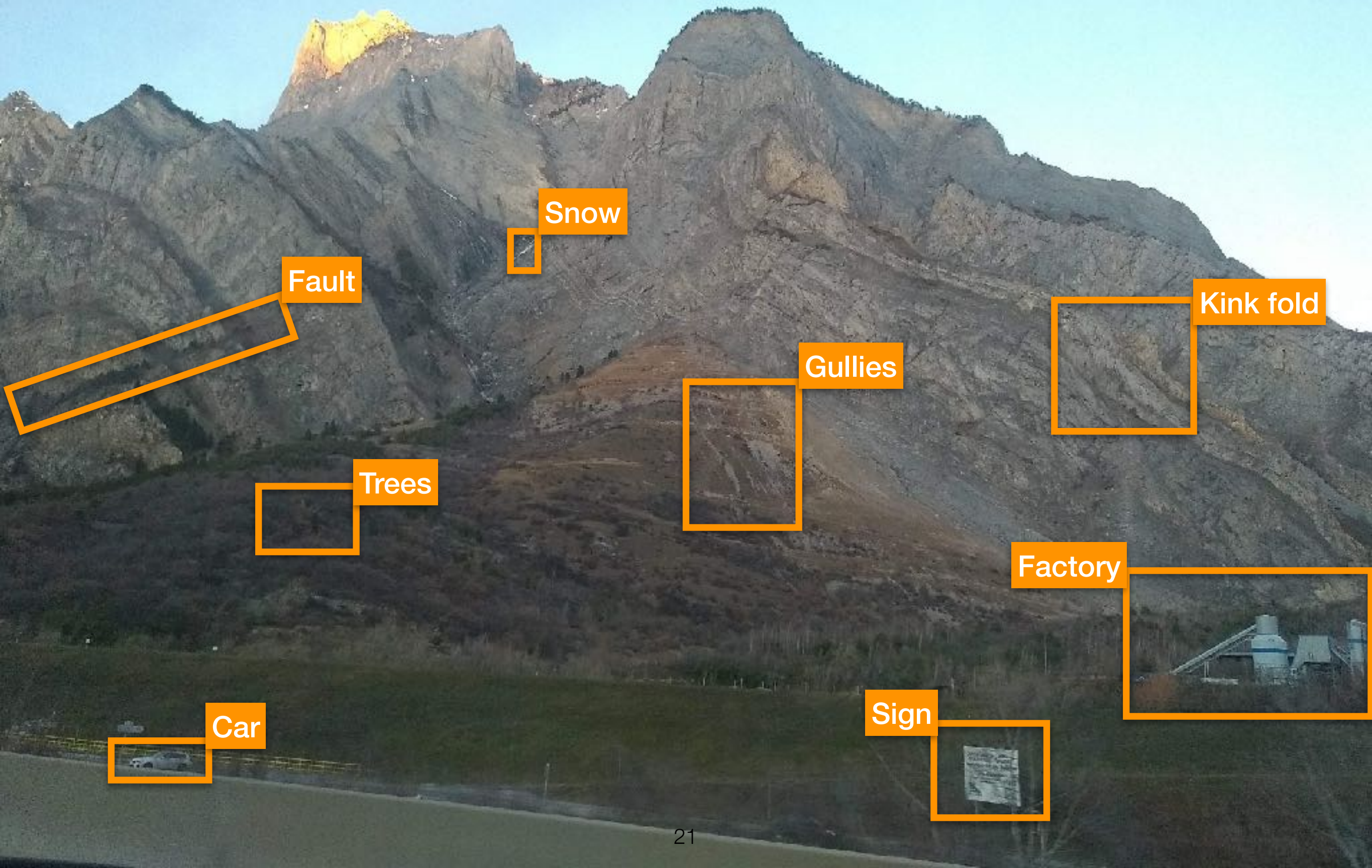
If t is discrete : **Classification** ➡ **Lecture 2**

{Machine} Learning of parameters w from observation x



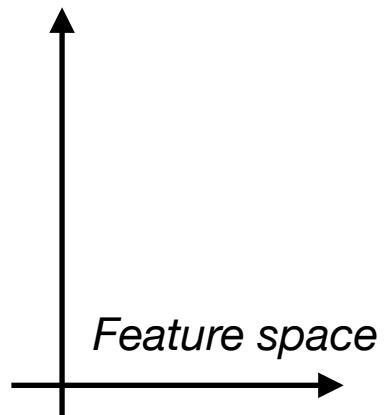
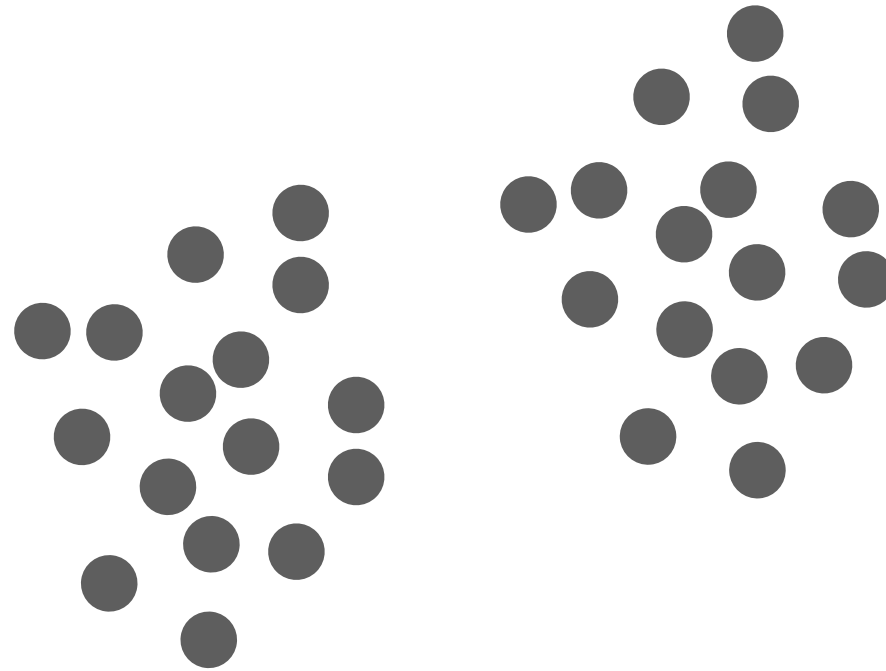
Supervises classification

Labeled features



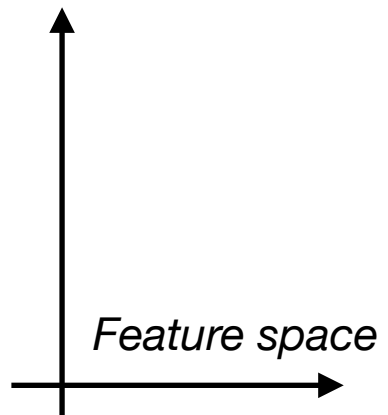
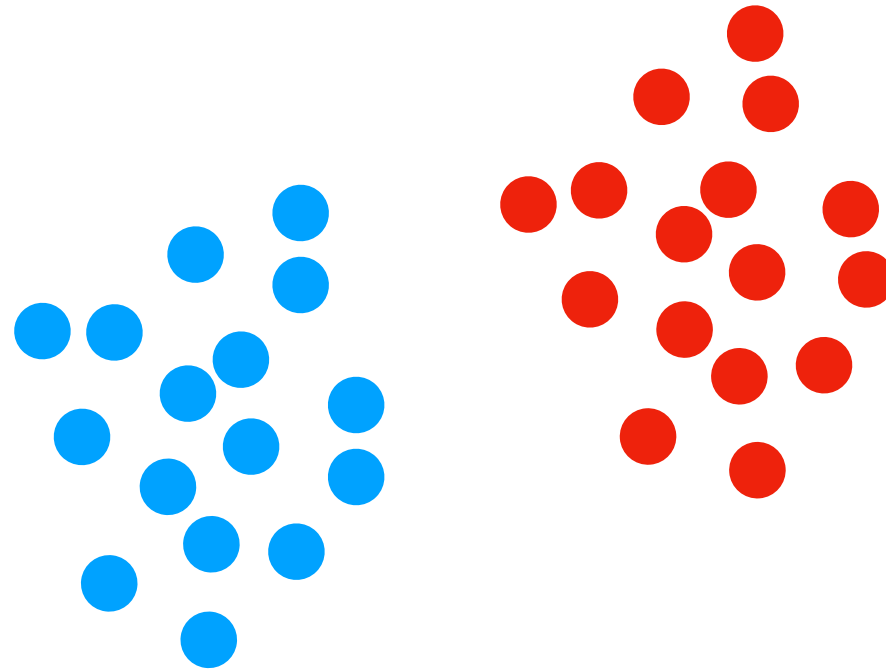
Learning and generalization

*Learning to classify data is like learning a function.
of decision: the boundary between classes.*



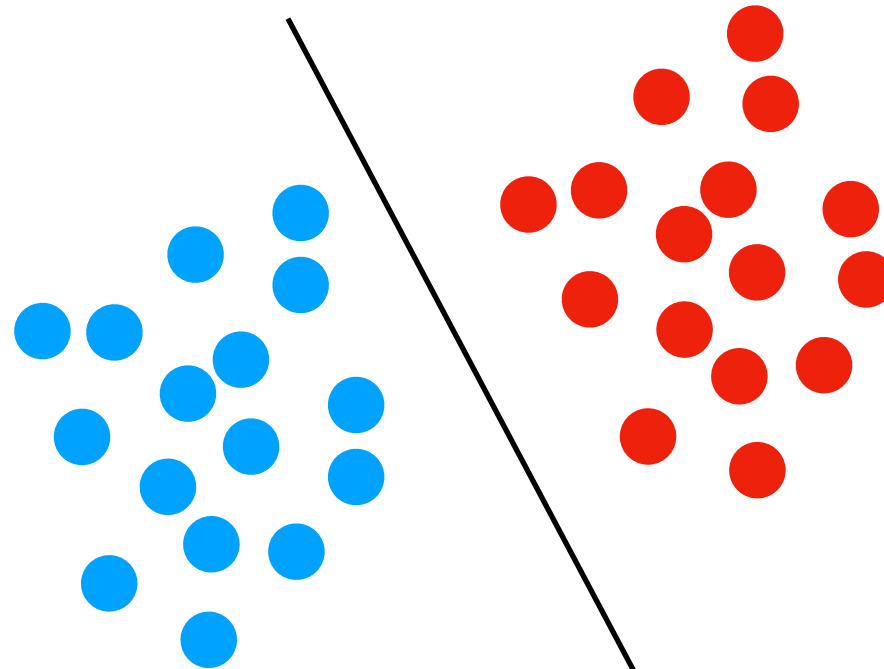
Learning and generalization

*Learning to classify data is like learning a function.
of decision: the boundary between classes.*

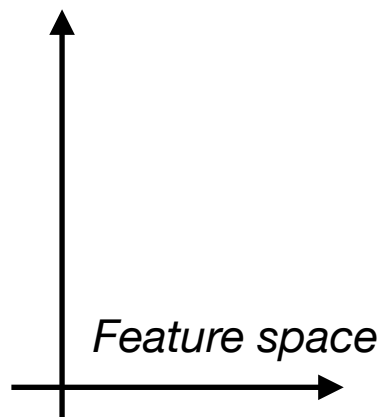


Learning and generalization

*Learning to classify data is like learning a function.
of decision: the boundary between classes.*

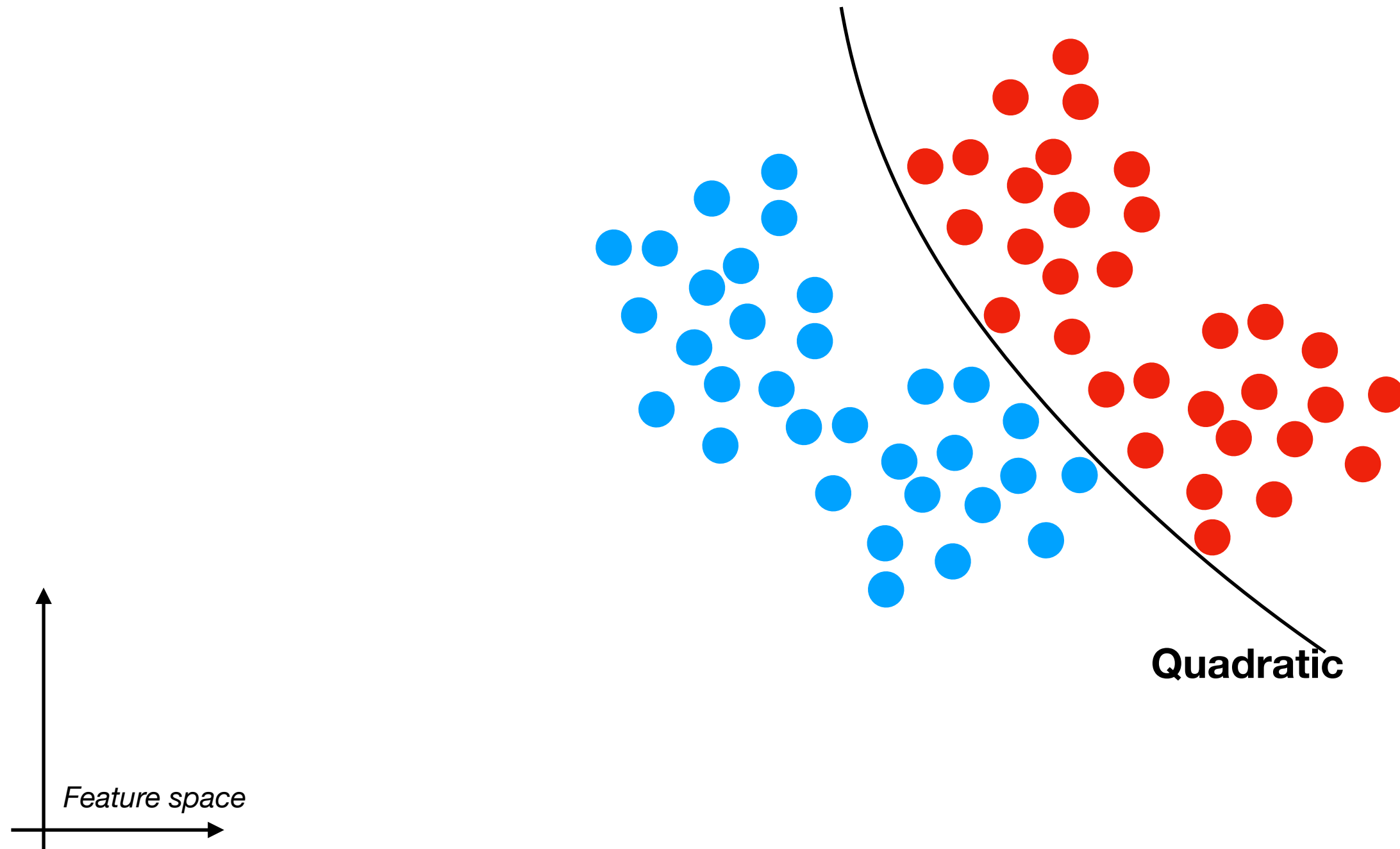


Linear



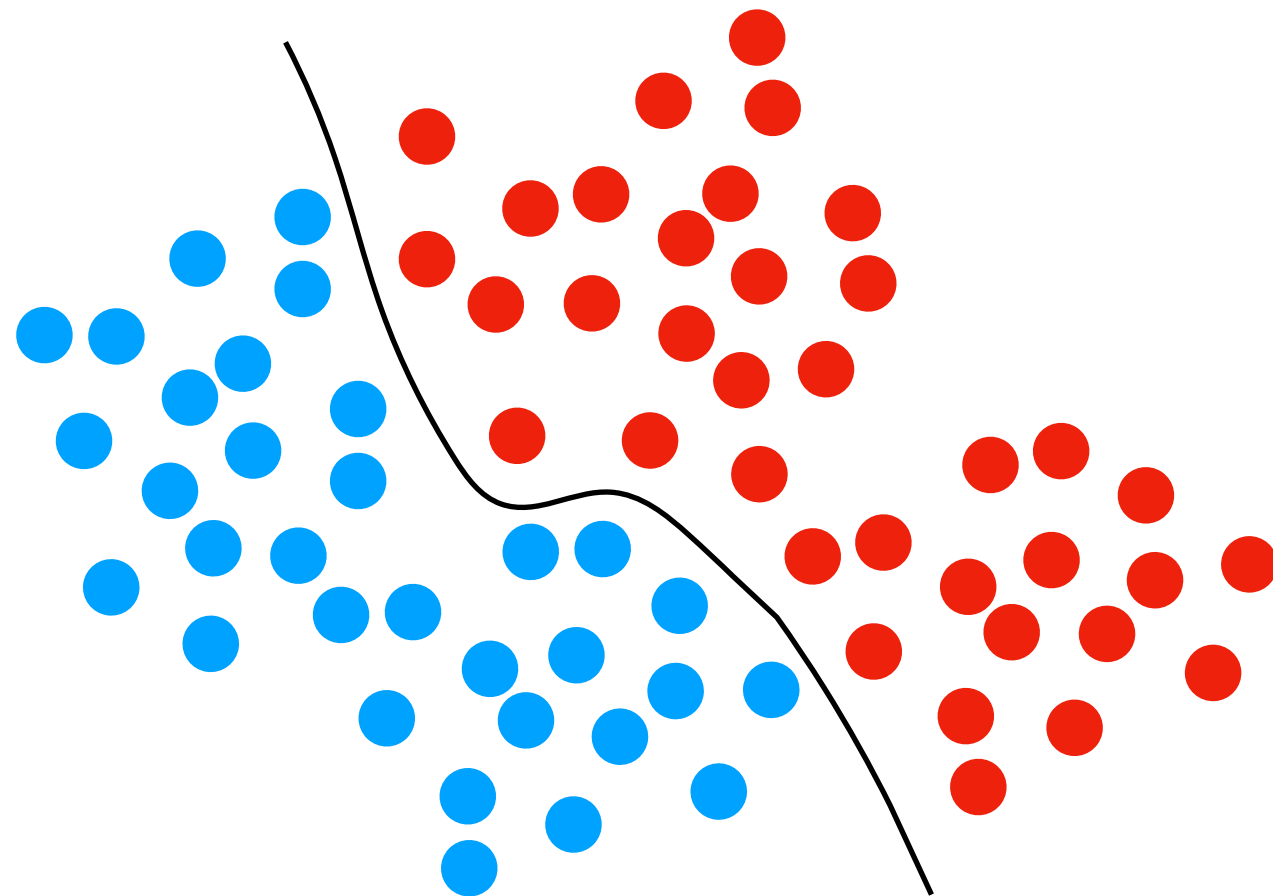
Learning and generalization

*Learning to classify data is like learning a function.
of decision: the boundary between classes.*

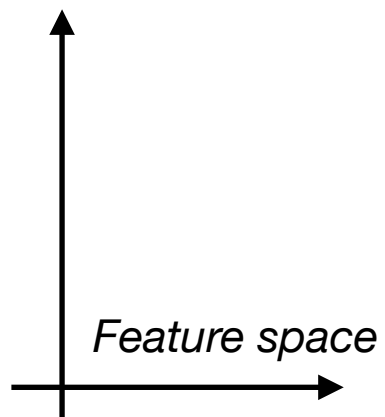


Learning and generalization

*Learning to classify data is like learning a function.
of decision: the boundary between classes.*

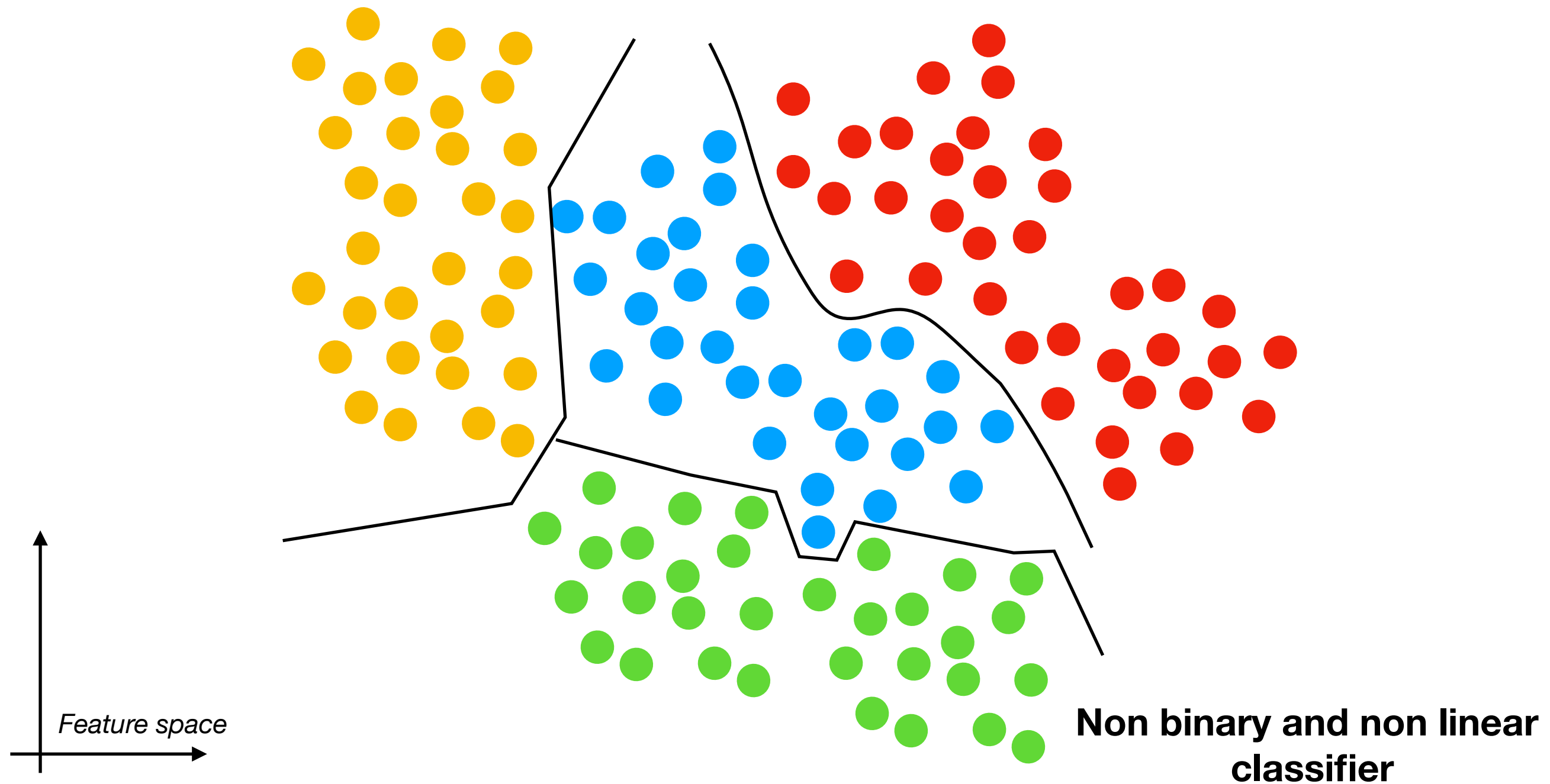


Decision tree



Learning and generalization

The complexity of a decision function depends on the complexity of the grouping of labels in the characteristics space (i.e. feature space)



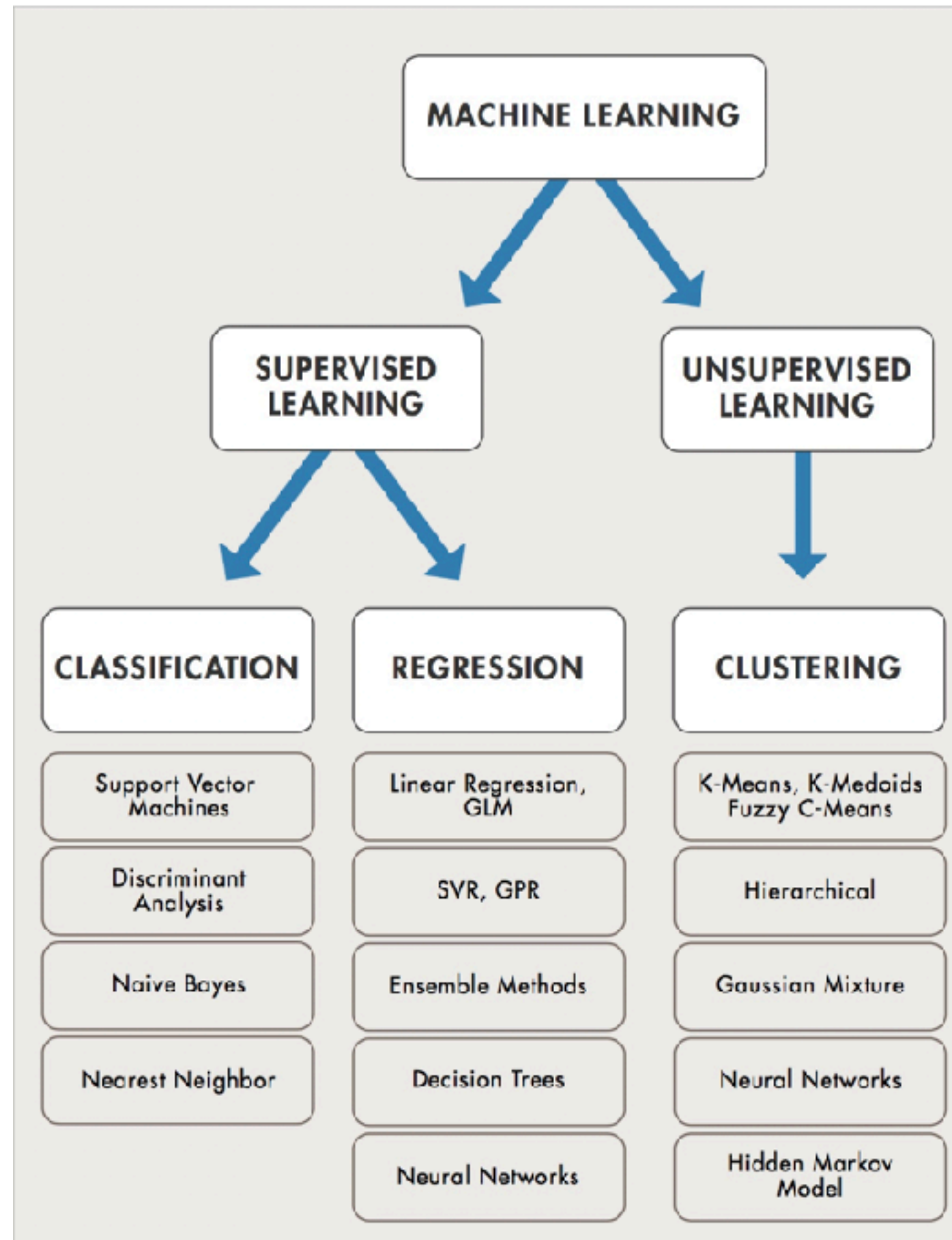
➡ Object's characteristics to identify which class it belongs

Supervised

Training a model on input and output data known exits so that he can predict the exit on future entries. An a priori the data set of learning

Unsupervised

Finding forms, an intrinsic structure in the data without *a priori*



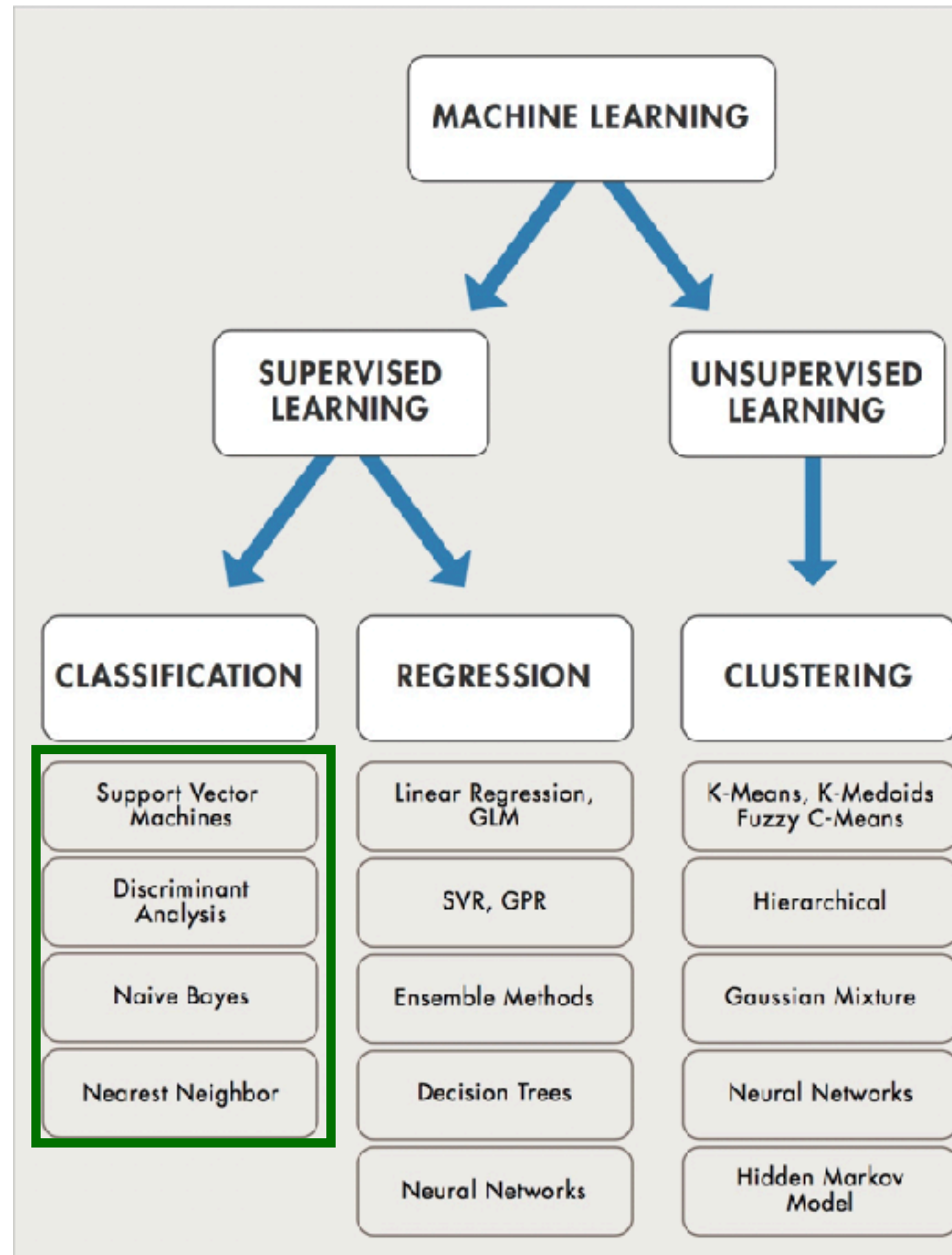
Supervised

Training a model on input and output data known exits so that he can predict the exit on future entries. An a priori the data set of learning

Unsupervised

Finding forms, an intrinsic structure in the data without *a priori*

Lecture #2.1



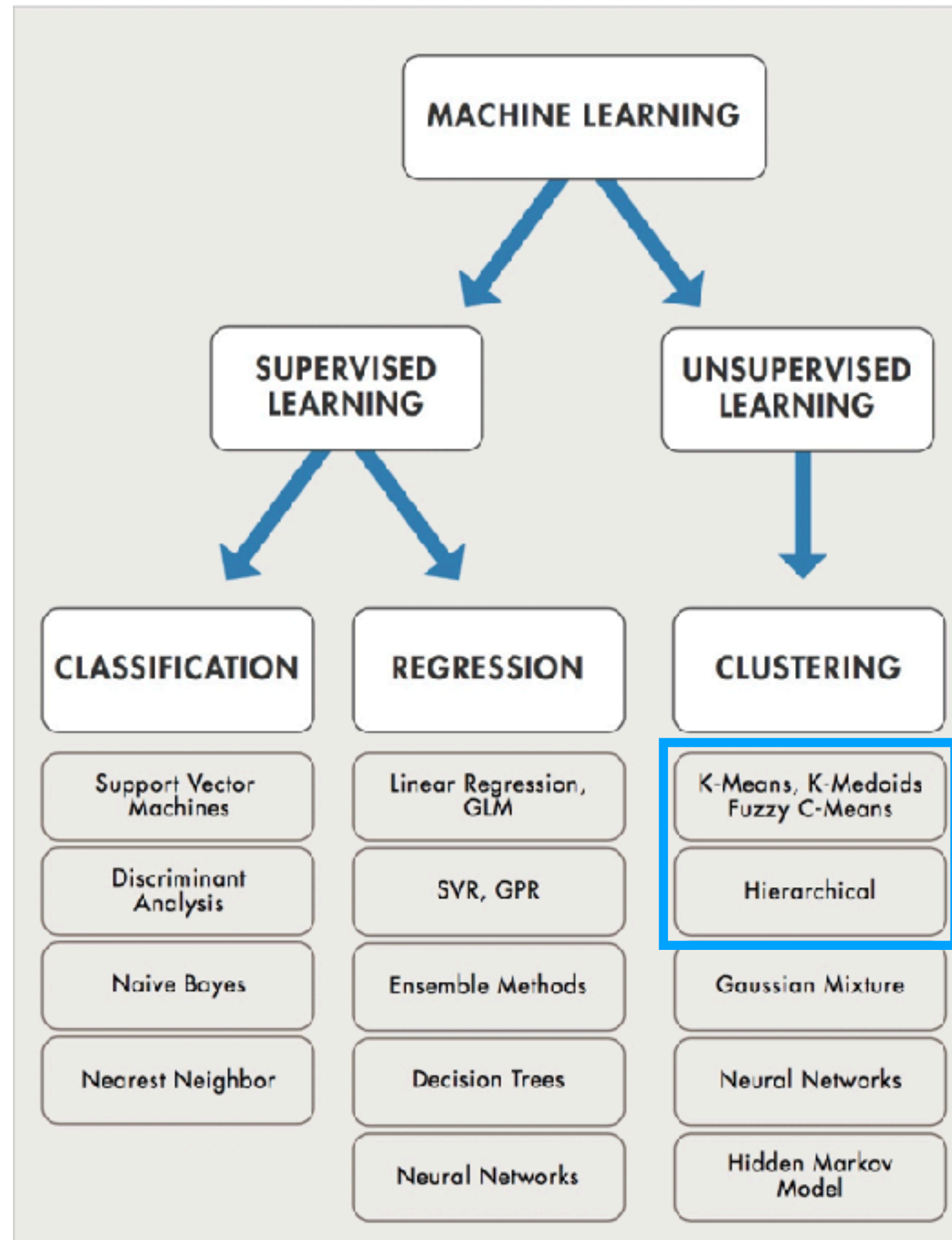
Supervised

Training a model on input and output data known exits so that he can predict the exit on future entries. An a priori the data set of learning

Unsupervised

Finding forms, an intrinsic structure in the data without *a priori*

Lecture #2.2



Supervised

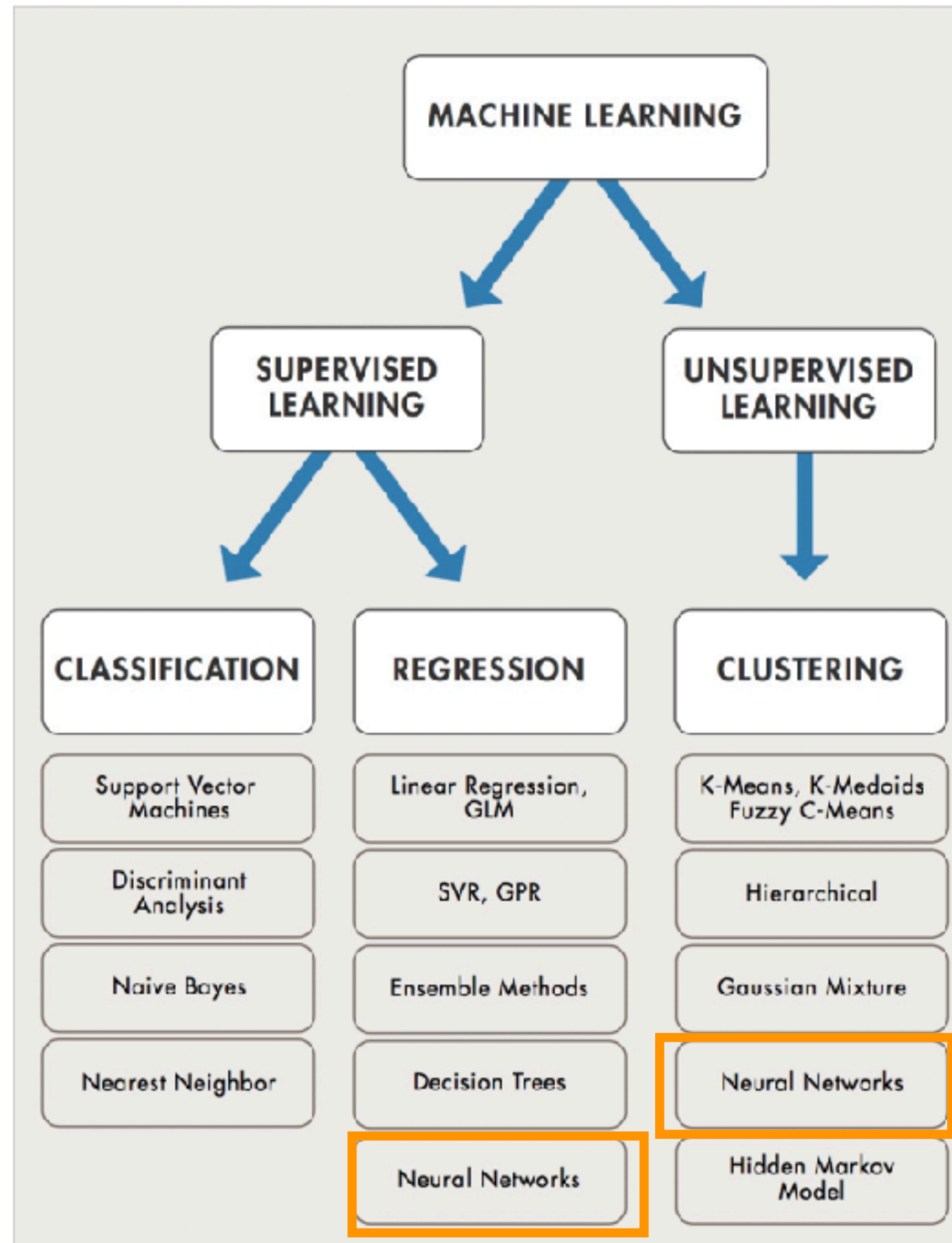
Training a model on input and output data known exits so that he can predict the exit on future entries. An a priori the data set of learning

Unsupervised

Finding forms, an intrinsic structure in the data without *a priori*

Lecture #2.3

with Greg in a few



Outline

I. Introduction to ML

- Generic ML workflow

II. Linear classifier

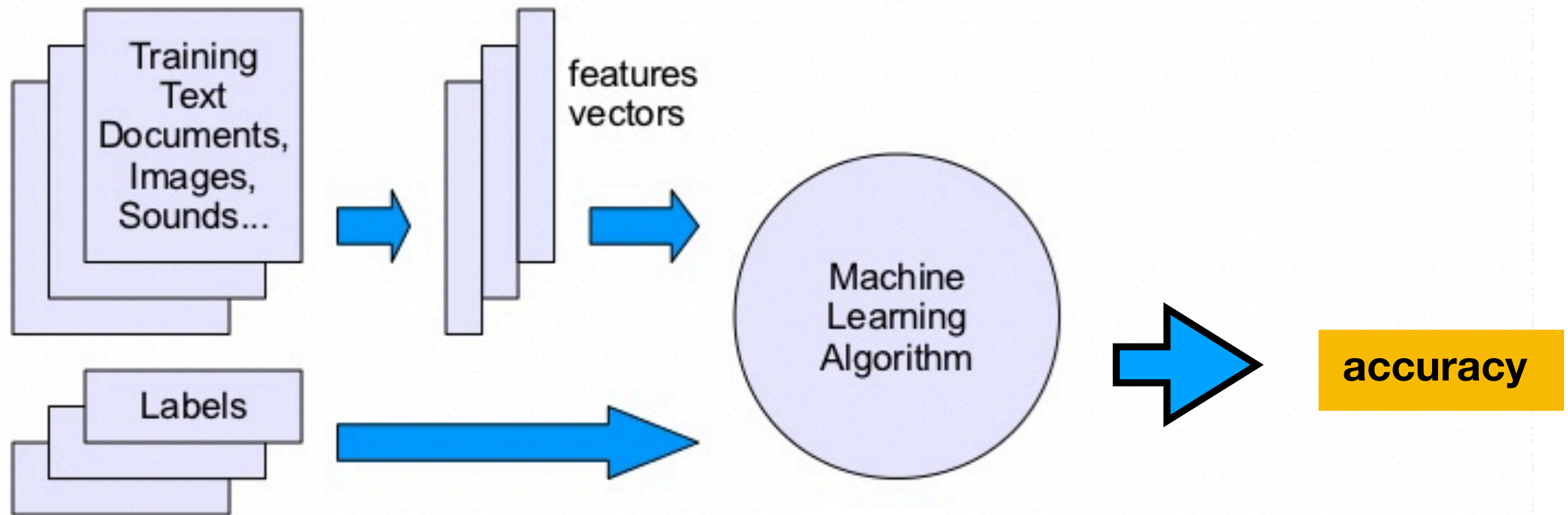
- Discriminative model,
example of Logistic Reg.
- Generative model,
example from LDA and Bayesian

III. Generalization

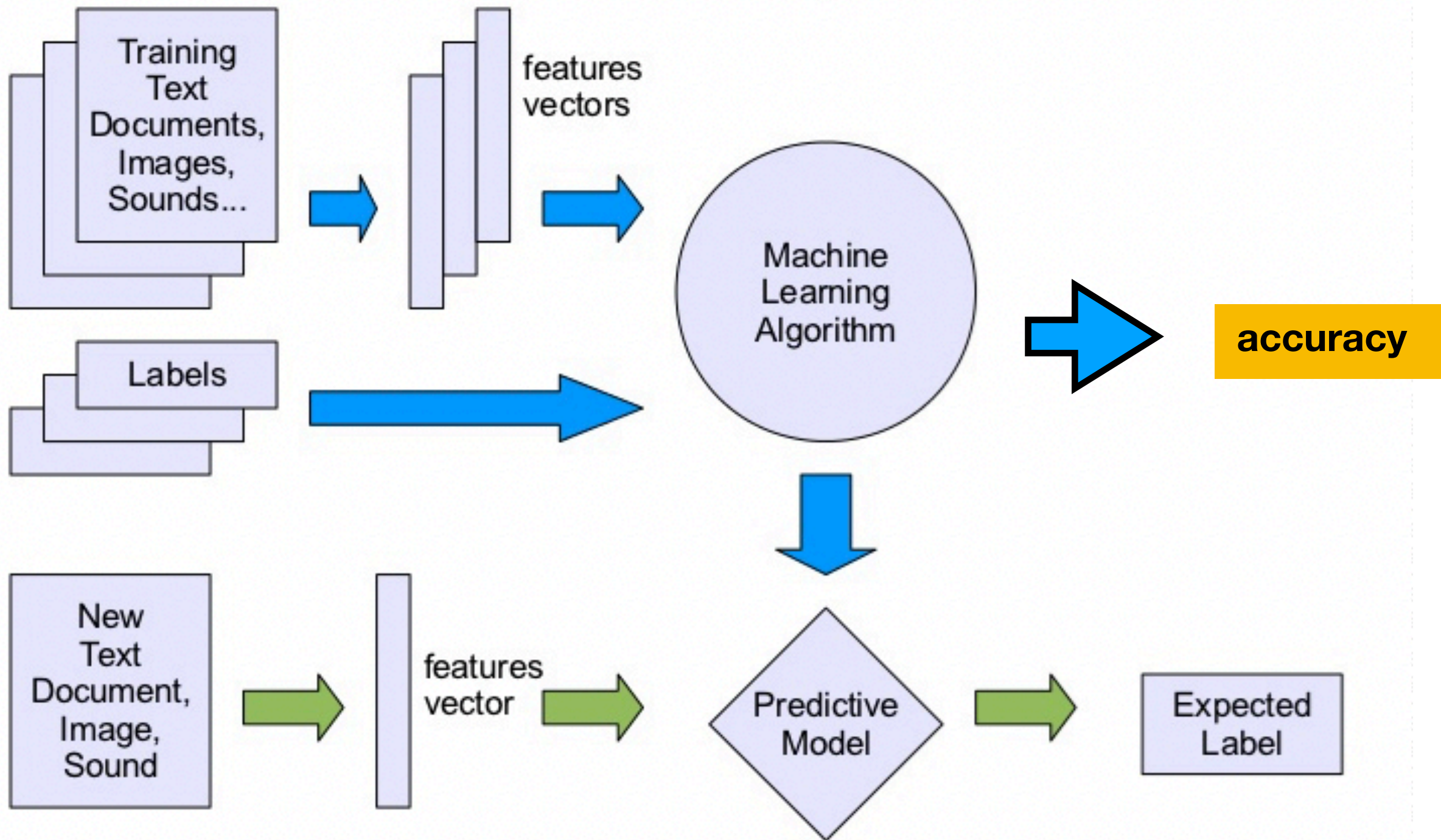
- Non-linear classifier
- Fitting problem
 - Regularisation
 - Cross-validation
- A word on performance and score

IV. Clustering

Workflow of supervised ML process



Workflow of supervised ML process



Linear classifier

linear function that assigns a score to each possible category k

$$\text{score}(\mathbf{X}_i, k) = \beta_k \cdot \mathbf{X}_i$$

Discriminative model

Attempts to maximize the quality of the output on a training set.

e.g., Logistic regression

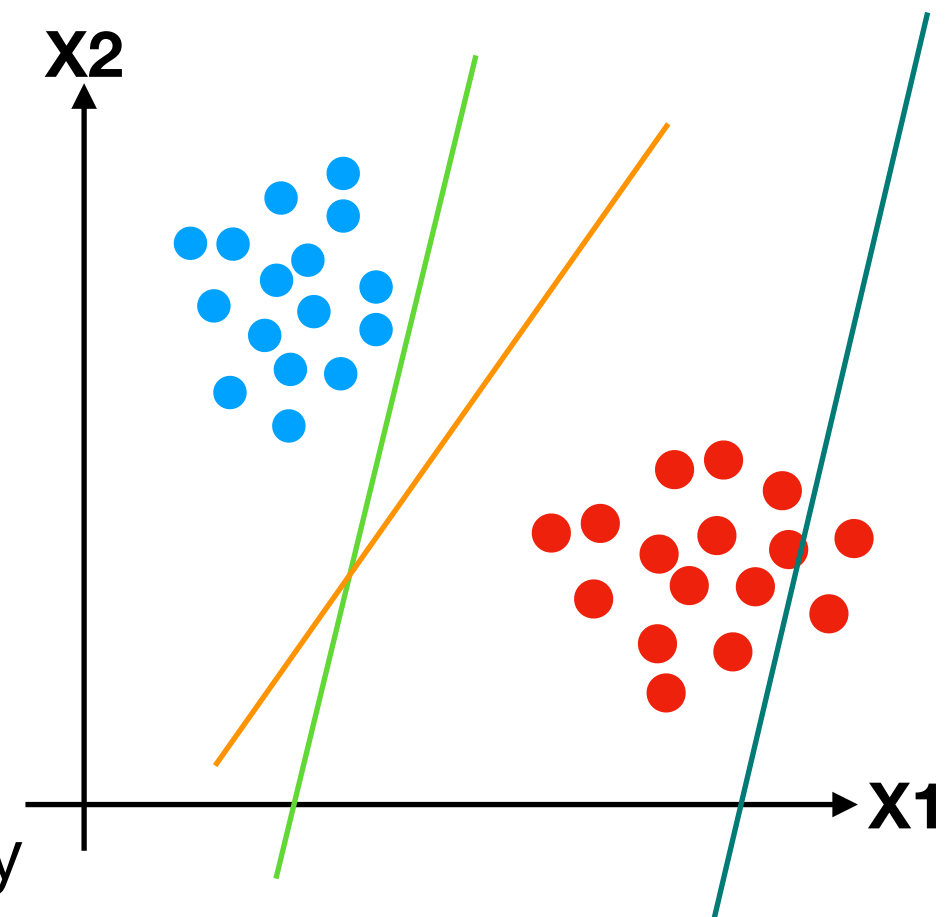
Generative model

*Try to model the **conditional probability distribution** of Y given X is the probability distribution of Y when X is known to be a particular value;*

e.g., Linear Discriminant Analysis (LDA)

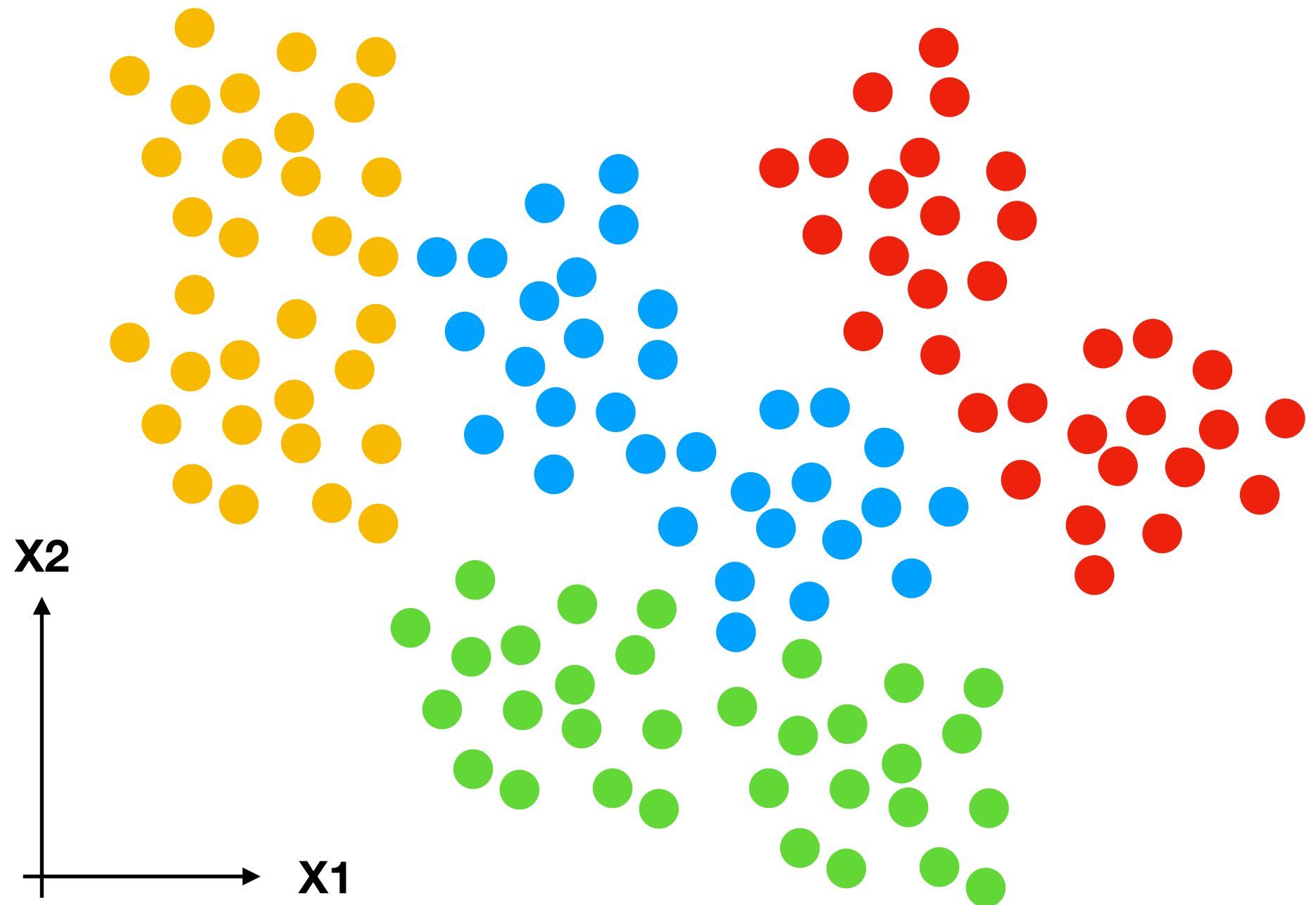
➡ Discriminative training often yields higher accuracy

➡ Conditional density models is more suited when handling missing data



K nearest neighbour

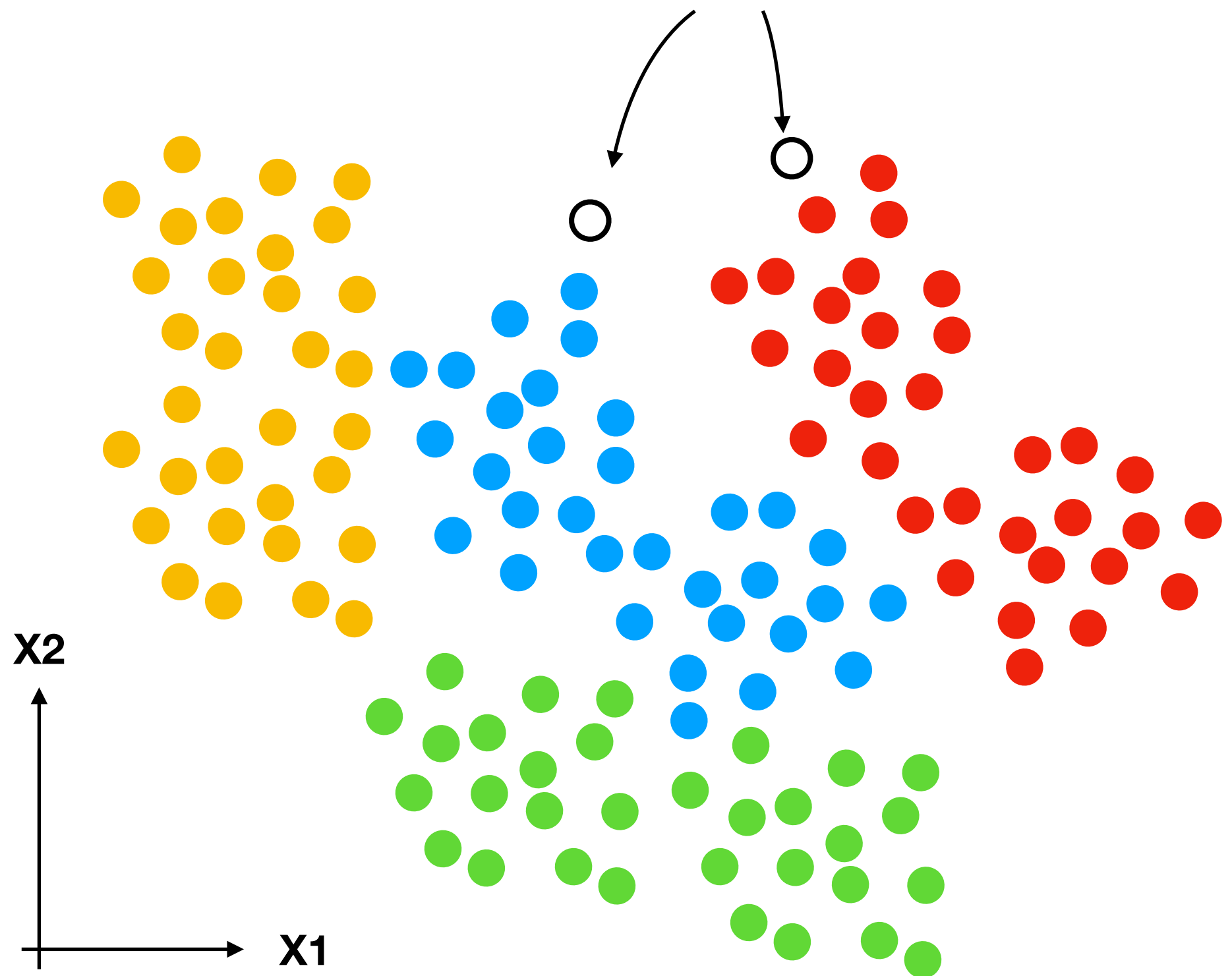
The simplest one



K nearest neighbour

New data to classify

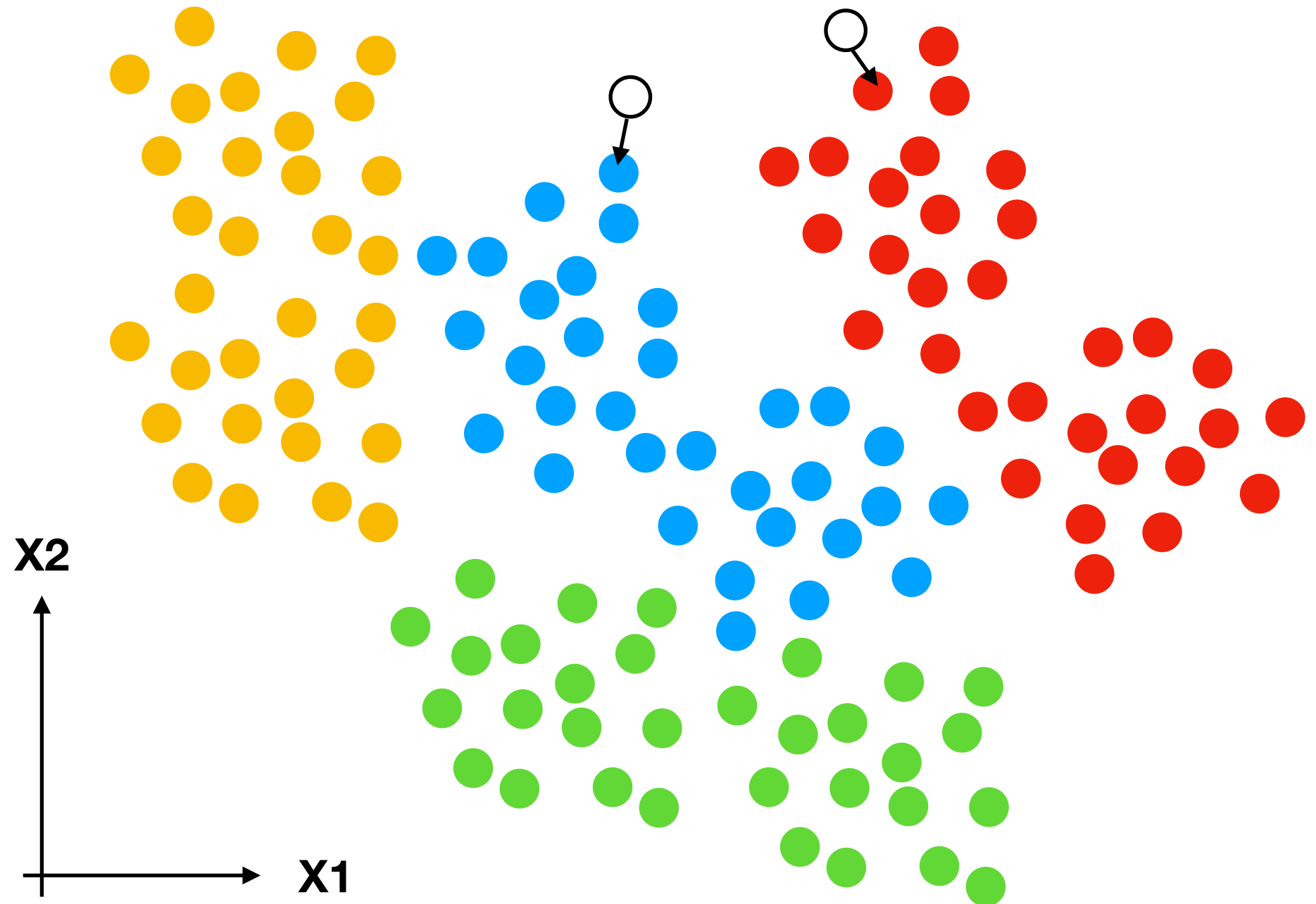
The simplest one



K nearest neighbour

We assign the class of the closest point in the X_n feature space by computing the euclidian distance

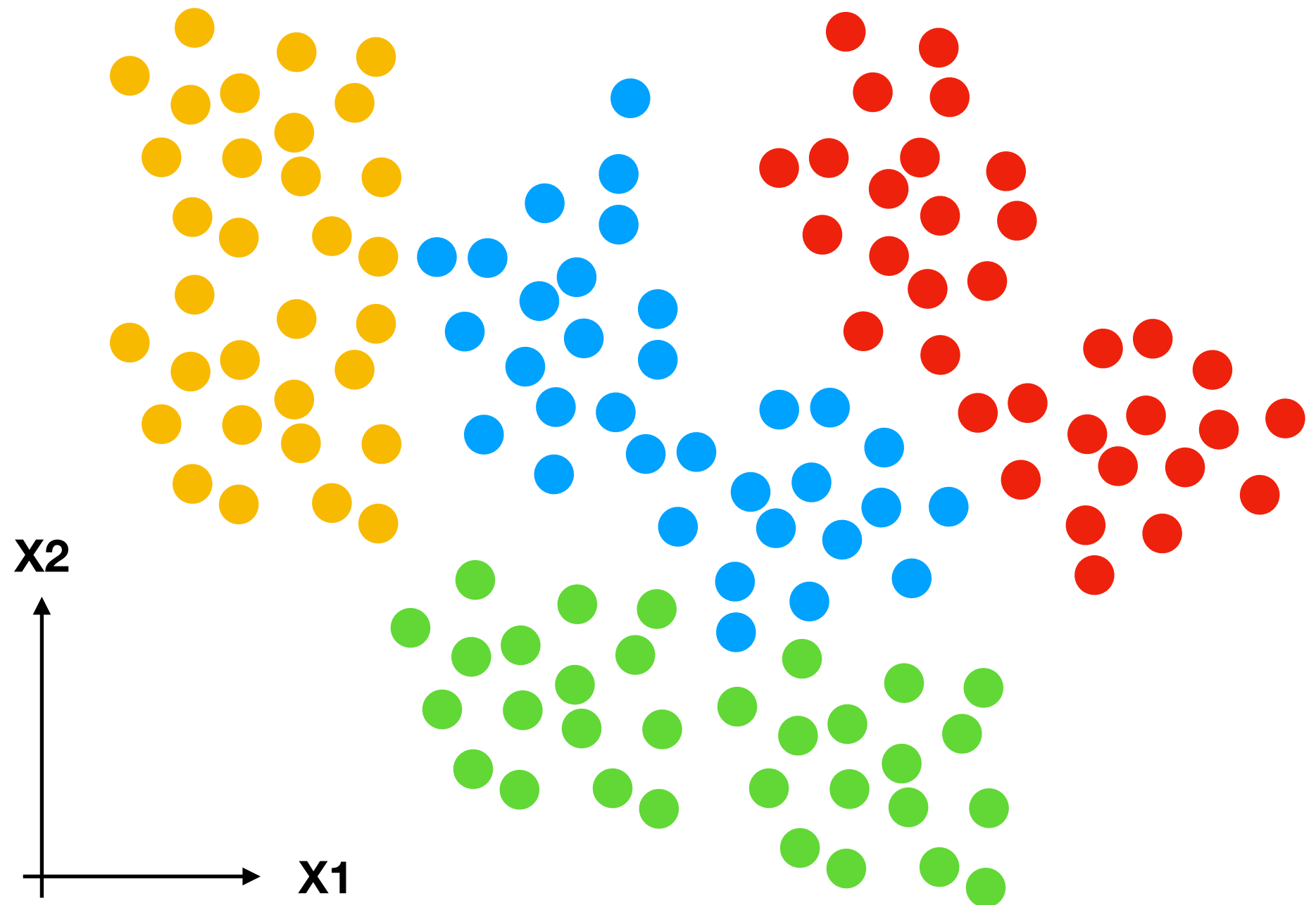
The simplest one



K nearest neighbour

We assign the class of the closest point in the X_n feature space by computing the euclidian distance

The simplest one



Logistic regression

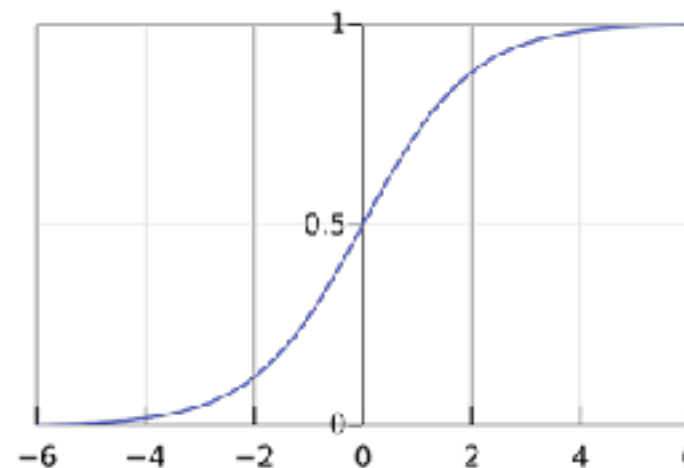
Special case of the linear regression

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

Where, y is dependent variable and $X_1, X_2 \dots$ and X_n are explanatory variables.

Sigmoid function is:

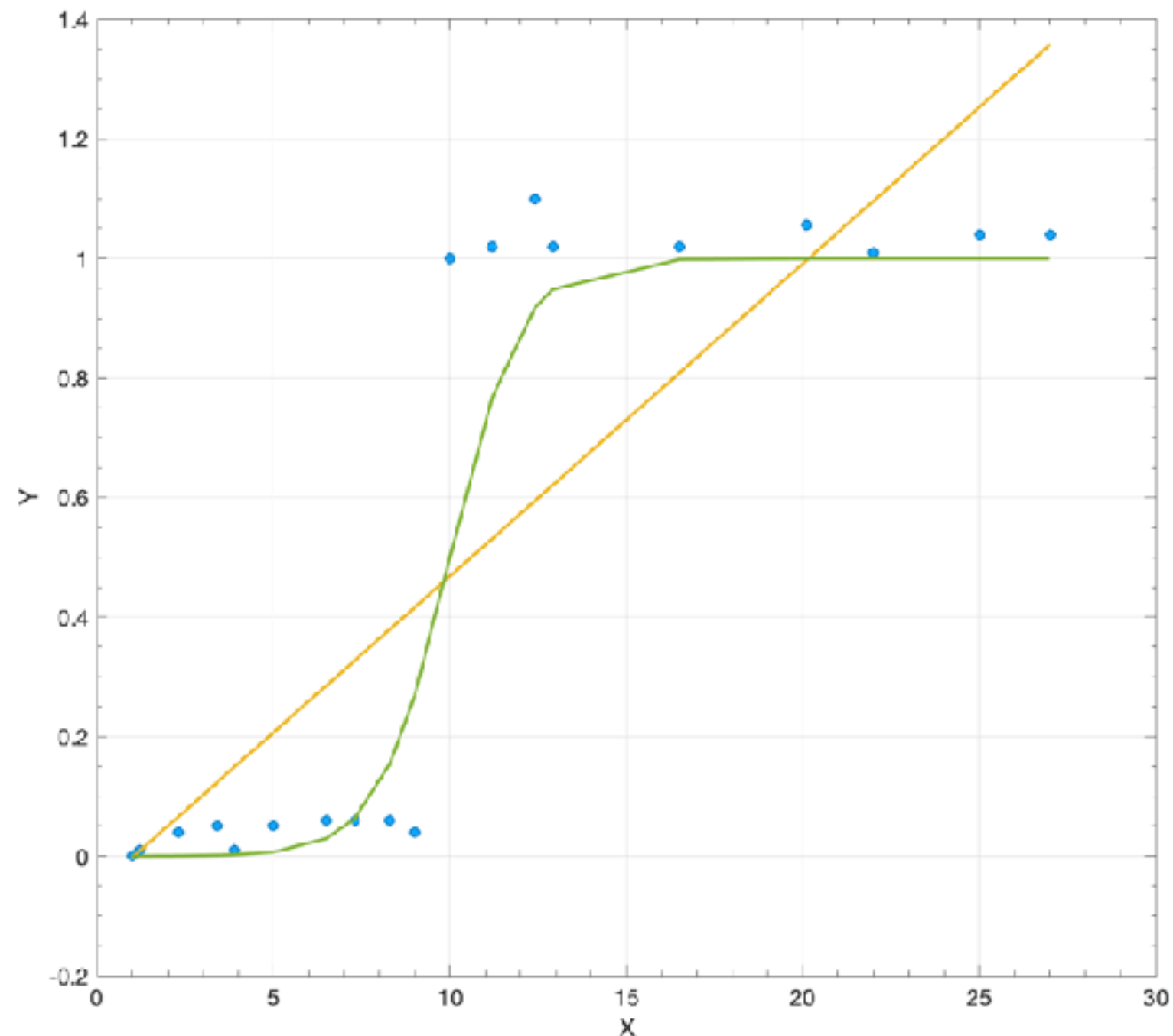
$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$



When applying sigmoid function on linear regression:

$$f(x) = \frac{1}{1 + e^{-y}}$$

$$f(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}}$$



Linear regression ► continuous output

Logistic regression ► discrete output

Linear regression is estimated using Ordinary Least Squares (OLS), **see slides from lecture 1** while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.

Note that when the errors are normally distributed, OLS is the maximum likelihood estimator.

Take away definition

Maximum Likelihood Estimation Vs. Least Square Method

The **MLE** is a "likelihood" maximization method, while OLS is a distance-minimizing approximation method. **Maximizing the likelihood function** determines the parameters that are most likely to produce the observed data. From a statistical point of view, MLE sets the mean and variance as parameters in determining the specific parametric values for a given model. This set of parameters can be used for predicting the data needed in a normal distribution.

Ordinary Least squares estimates are computed by fitting a regression line on given data points that has the **minimum sum of the squared deviations** (least square error).

Both are used to estimate the parameters of a linear regression model.

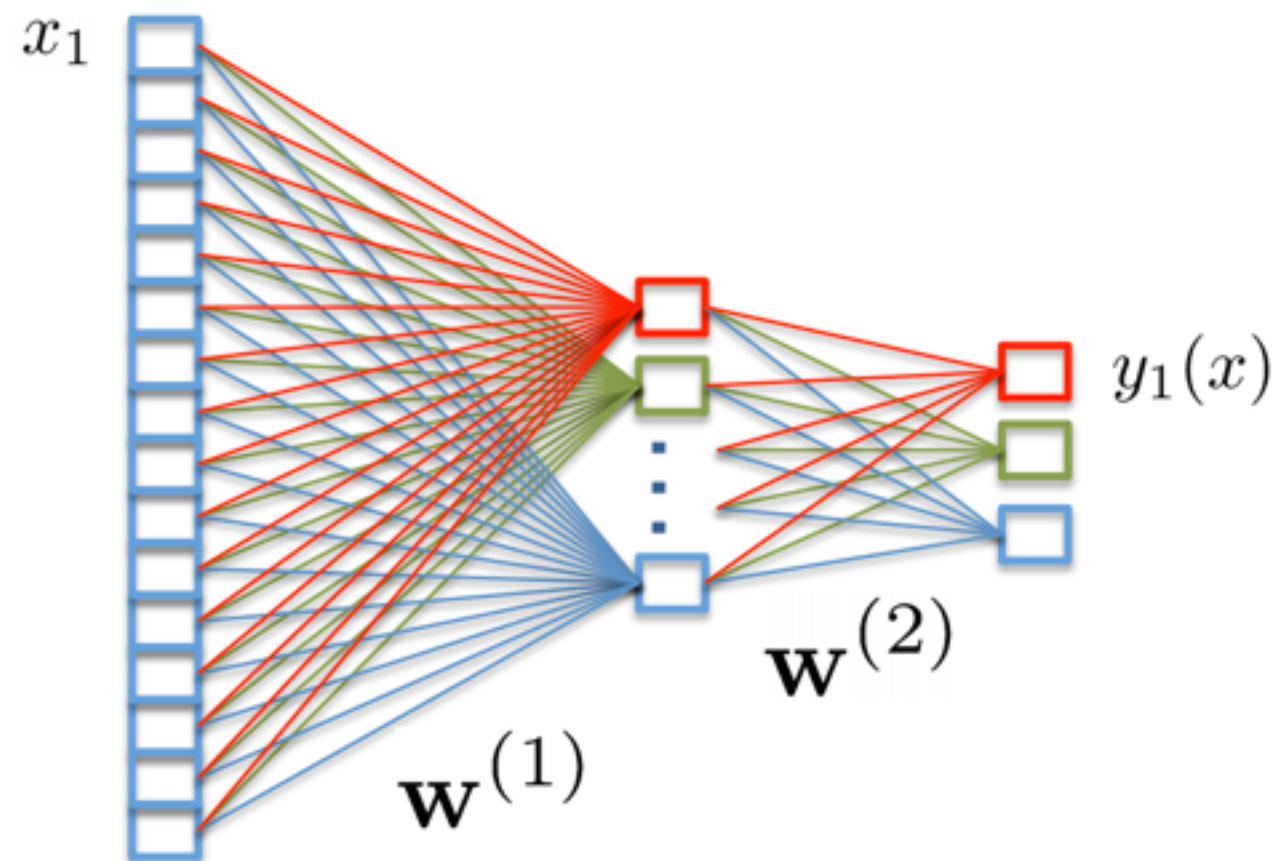
MLE assumes a joint probability mass function, while OLS doesn't require any stochastic assumptions for minimizing distance.

Take away message

Bimodal logistic regression is the entry to machine learning

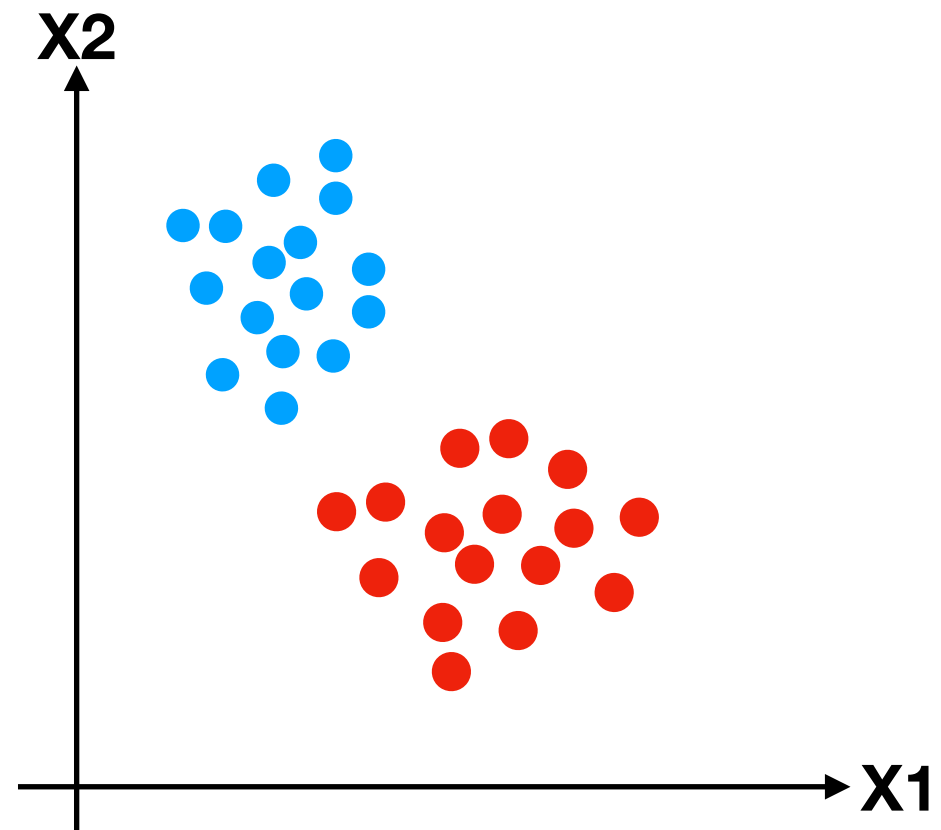
K-classe logistic regression is the entry to deep learning,

We will go back to this during the second part of this lecture with Greg.



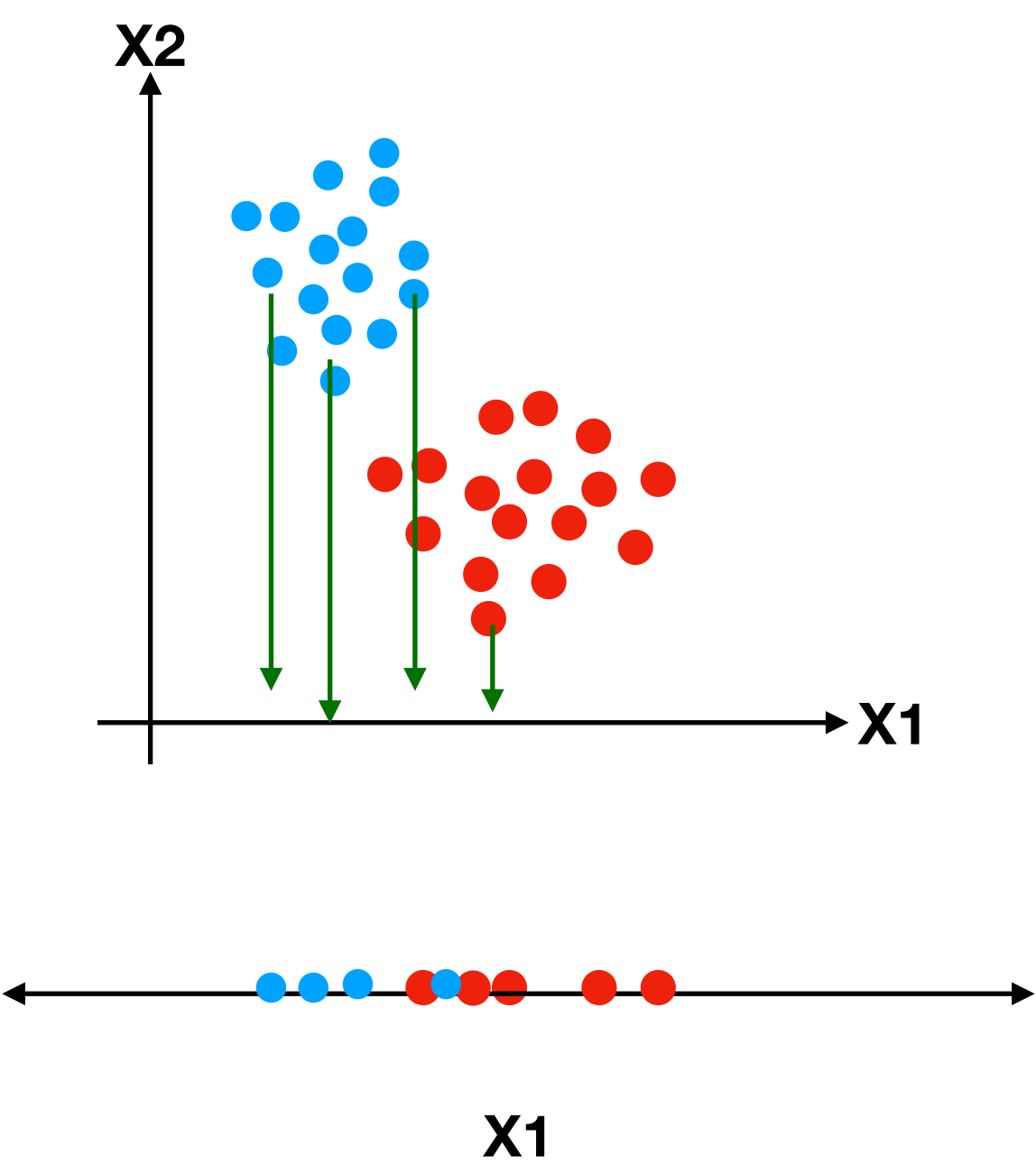
Generative model

Linear Discriminant Analysis (LDA)

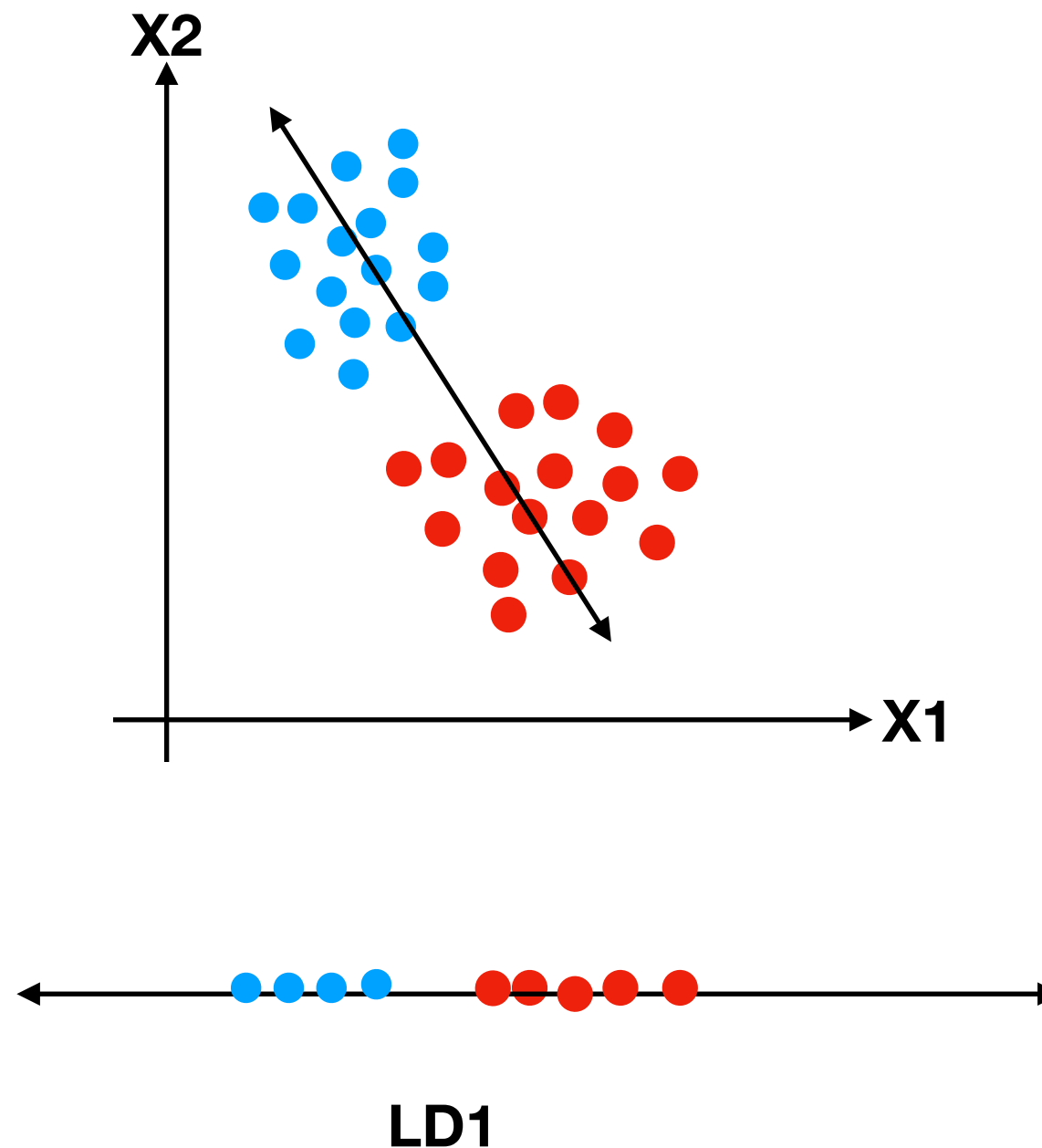


Generative model

Linear Discriminant Analysis (LDA)



We search for a function that minimises the variances and maximises the distance btw the means of the two classes



1. Compute the within class and between class scatter matrices
2. Compute the eigenvectors and eigenvalues for the scatter matrices
3. Sort the eigenvalues and select the top k
4. Create a new matrix containing eigenvectors that map to the k eigenvalues
5. Obtain the new features (i.e. LDA components) by taking the dot product of the data and the matrix from step 4

Scatter Matrix :

A scatter matrix is an estimation of covariance matrix when covariance cannot be calculated or costly to calculate. The scatter matrix is also used in lot of dimensionality reduction exercises. If there are k variables, scatter matrix will have k rows and k columns (*i.e* $k \times k$ matrix).

The scatter matrix is obtained from:

$$S = \sum_{k=1}^n (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T$$

with **\mathbf{m}** being the mean vector:

$$\mathbf{m} = \frac{1}{n} \sum_{k=1}^n \mathbf{x}_k$$

Generative model Linear Discriminant Analysis (LDA)

Within Class Scatter Matrix

We calculate the *within class scatter matrix* using the following formula.

$$S_W = \sum_{i=1}^c S_i$$

where c is the total number of distinct classes and

$$S_i = \sum_{\mathbf{x} \in D_i}^n (\mathbf{x} - \mathbf{m}_i) (\mathbf{x} - \mathbf{m}_i)^T$$

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i}^n \mathbf{x}_k$$

where \mathbf{x} is a sample (i.e. row) and n is the total number of samples with a given class.

Generative model Linear Discriminant Analysis (LDA)

In Python we do it this way (if/when using Panda)

```
class_feature_means = pd.DataFrame(columns=data.target_names)
for c, rows in df.groupby('class'):
    class_feature_means[c] = rows.mean()
class_feature_means
```

```
within_class_scatter_matrix = np.zeros((13,13))
for c, rows in df.groupby('class'):
    rows = rows.drop(['class'], axis=1)

    s = np.zeros((13,13))
    for index, row in rows.iterrows():
        x, mc = row.values.reshape(13,1), class_feature_means[c].values.reshape(13,1)
        s += (x - mc).dot((x - mc).T)

    within_class_scatter_matrix += s
```

Between Class Scatter Matrix

Next, we calculate the *between class scatter matrix* using the following formula.

$$S_B = \sum_{i=1}^c N_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

where

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}_k$$

$$\mathbf{m} = \frac{1}{n} \sum_i^n x_i$$

```
feature_means = df.mean()
between_class_scatter_matrix = np.zeros((13,13))
for c in class_feature_means:
    n = len(df.loc[df['class'] == c].index)
    mc, m = class_feature_means[c].values.reshape(13,1), feature_means.values.reshape(13,1)
    between_class_scatter_matrix += n * (mc - m).dot((mc - m).T)
```

Then, we solve the generalized eigenvalue problem for

$$S_W^{-1} S_B$$

S_W : within class **S_B : between classes**
to obtain the linear discriminants.

```
eigen_values, eigen_vectors =  
np.linalg.eig(np.linalg.inv(within_class_scatter_matrix).dot(between_class_scatter_matrix))
```

The eigenvectors with the highest eigenvalues carry the most information about the distribution of the data. Thus, we sort the eigenvalues from highest to lowest and select the first **k** eigenvectors. *In order to ensure that the eigenvalue maps to the same eigenvector after sorting, we place them in a temporary array.*

```
pairs = [(np.abs(eigen_values[i]), eigen_vectors[:,i]) for i in range(len(eigen_values))]  
pairs = sorted(pairs, key=lambda x: x[0], reverse=True)  
for pair in pairs:  
    print(pair[0])
```

```
eigen_value_sums = sum(eigen_values)  
print('Explained Variance')  
for i, pair in enumerate(pairs):  
    print('Eigenvector {}: {}'.format(i, (pair[0]/eigen_value_sums).real))
```


So, bottom line is....

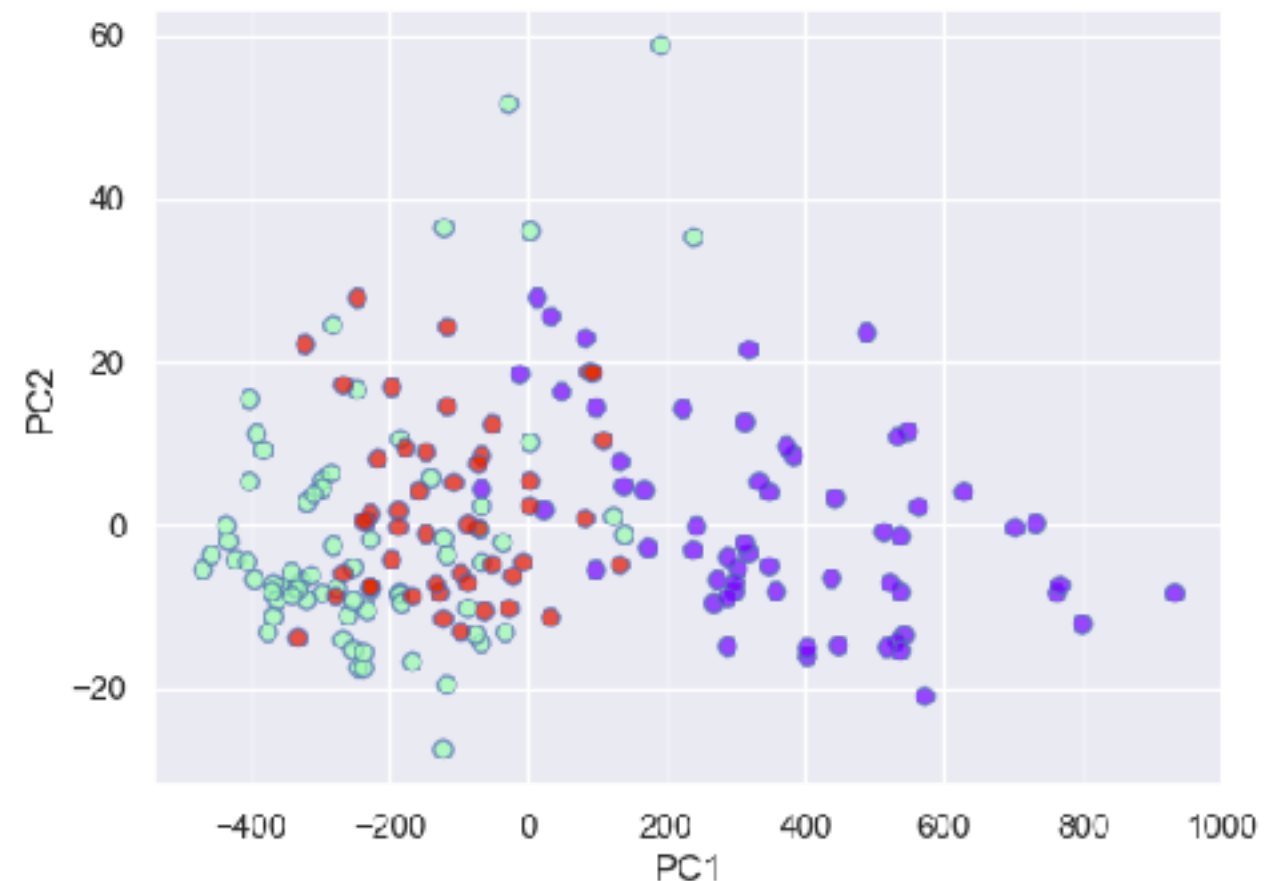
From lecture #1:

PCA selects the components which would result in the highest spread (retain the most information) and not necessarily the ones which maximize the separation between classes.

LDA



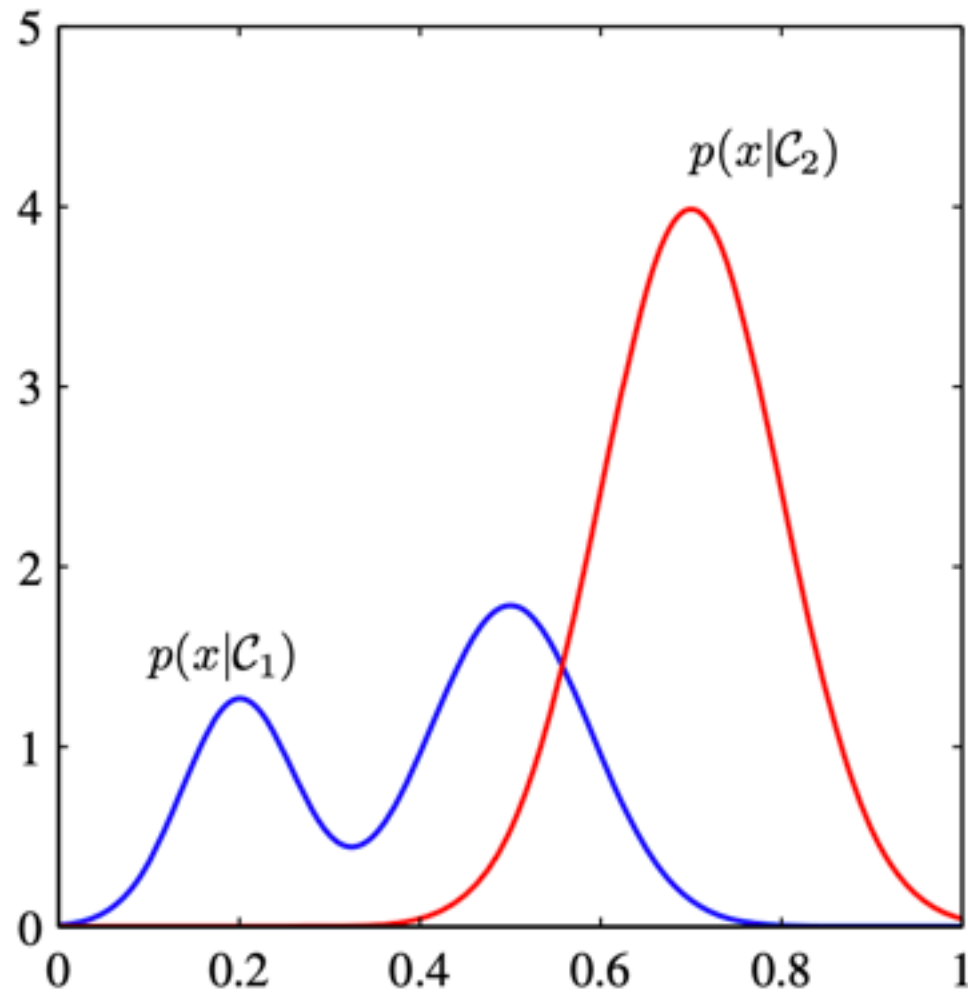
PCA



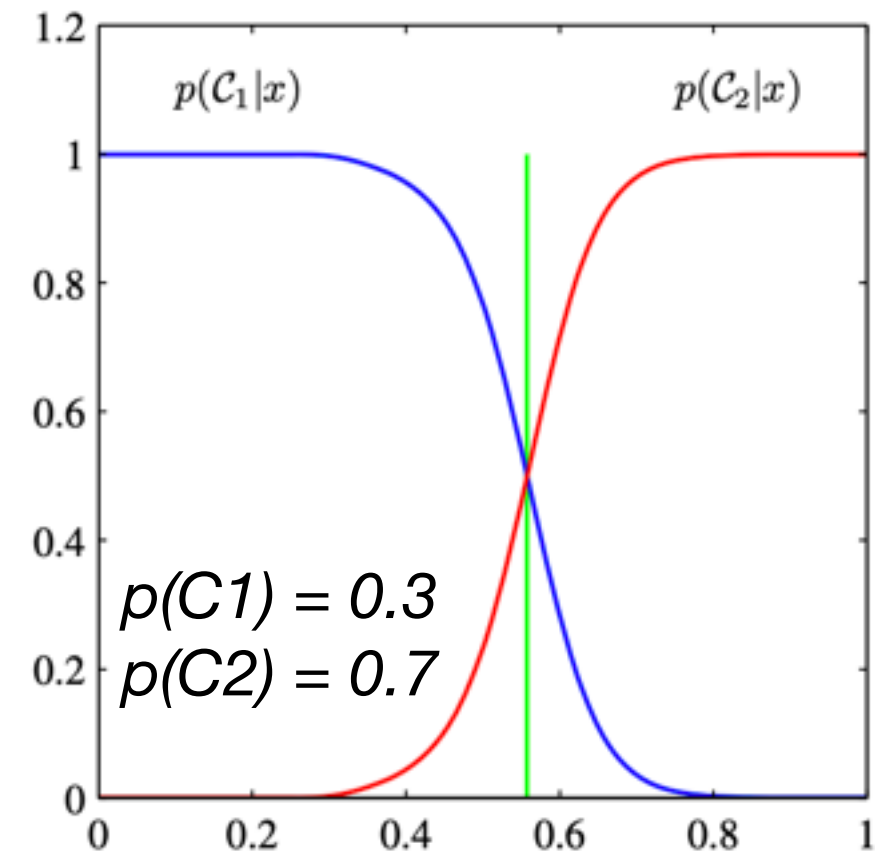
Generative model

A word on probabilistic classifier (Bayes)

Suppose a problem of classification into two classes C_1 and C_2 and from 1D data modelled by a random variable x (Observation)



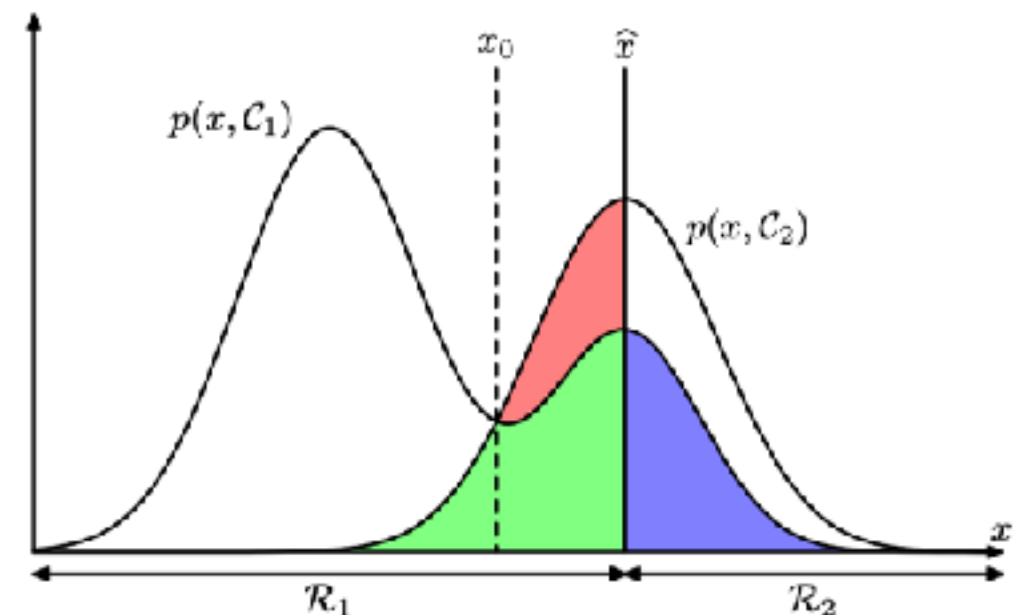
[C. Bishop, Pattern recognition and Machine learning, 2006]



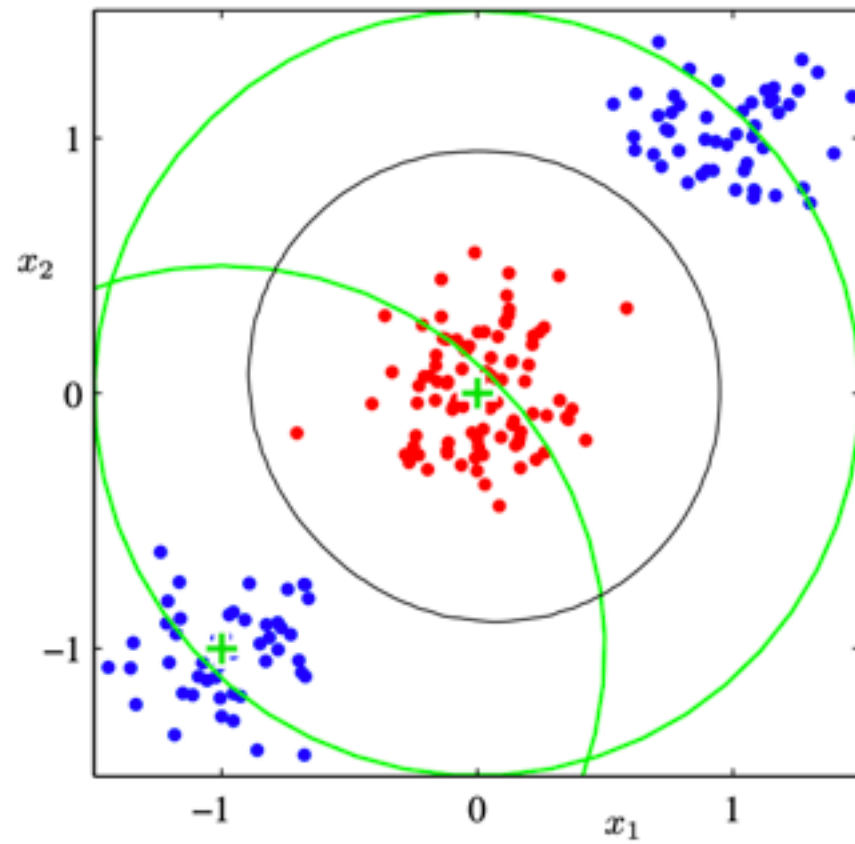
Likelihood

A priori

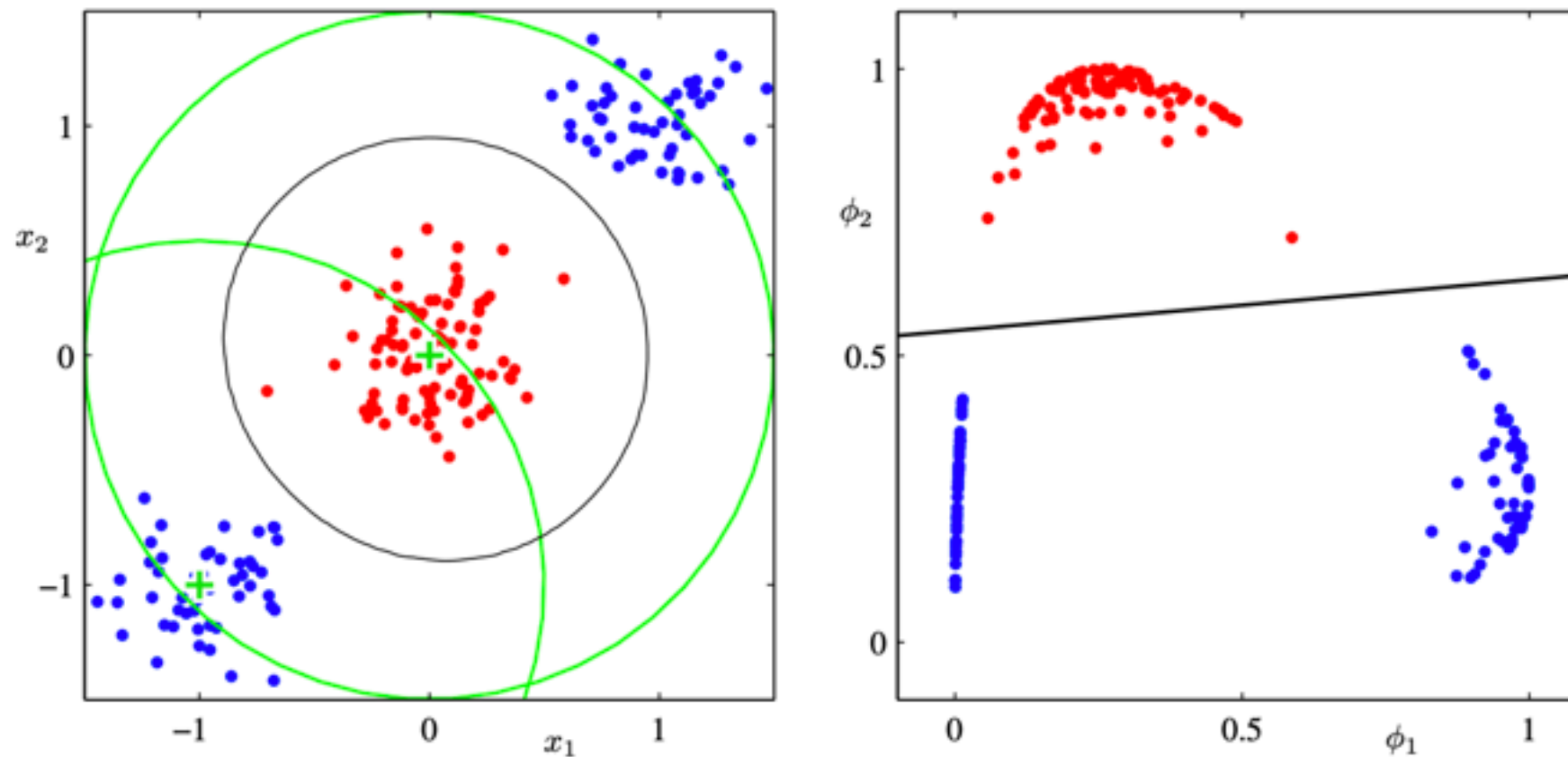
$$p(C_K | x) = \frac{p(x | C_k)p(C_k)}{p(x)}$$



Non-linear cases

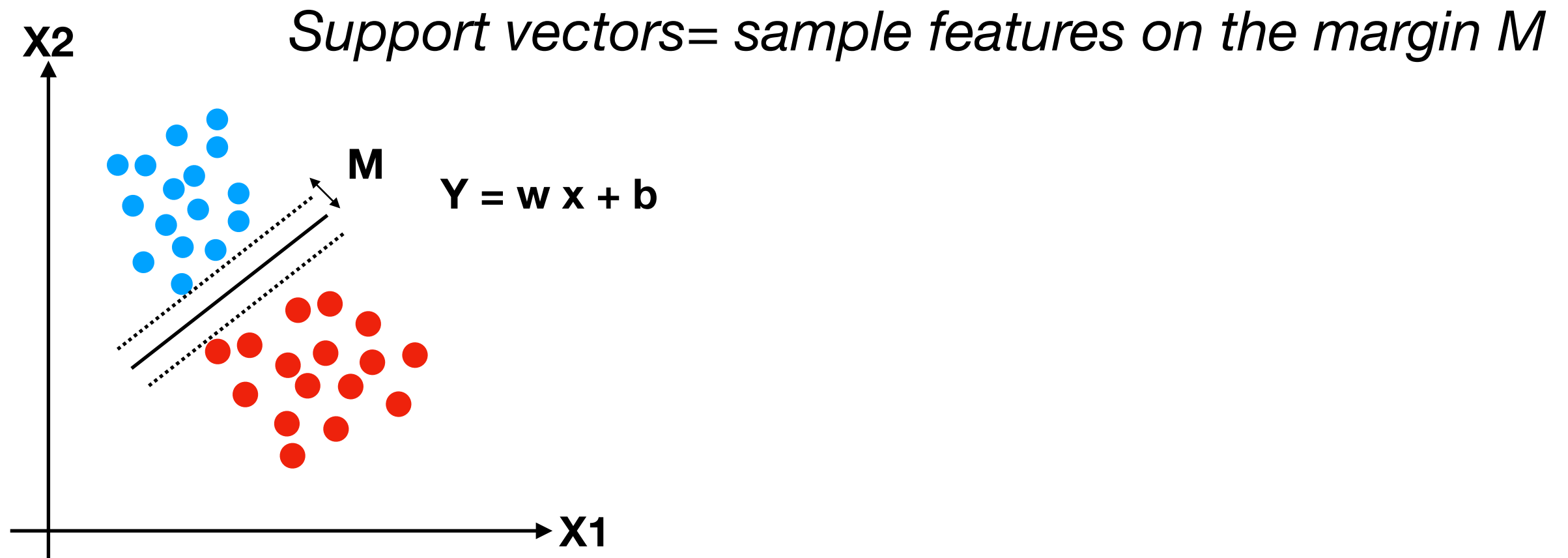


Non-linear cases



Remember, like the non-linear regression (see lecture 1)

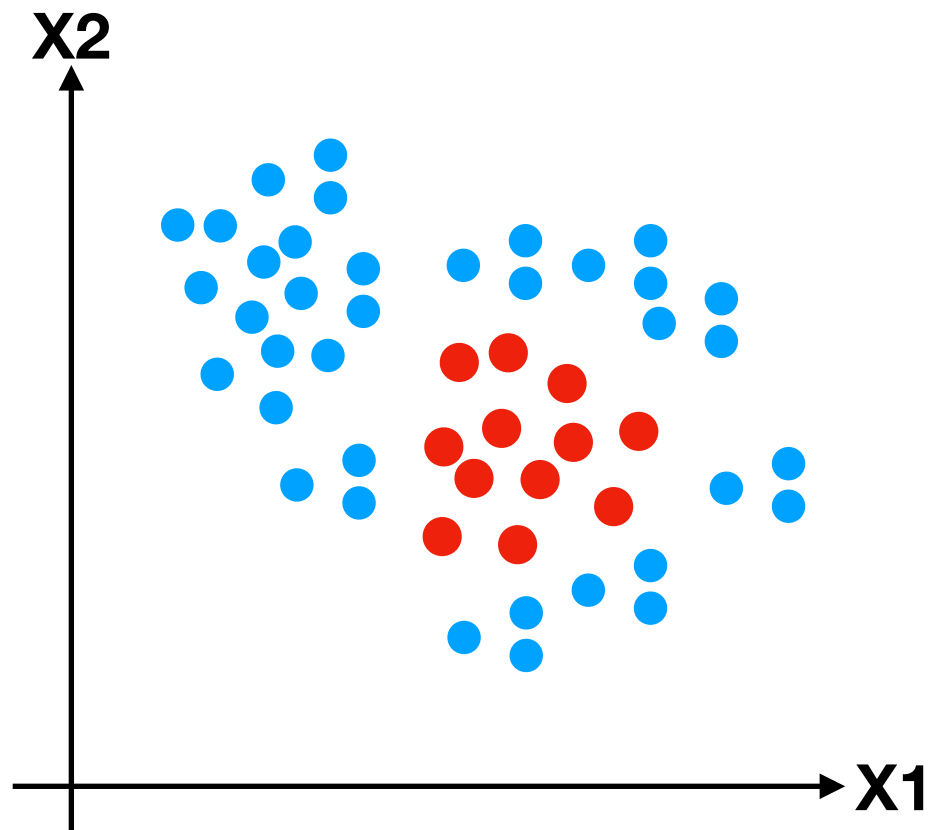
Support Vector Classifier



Margin M = Minimal distance between hyperplane and observations
Optimal Hyperplane = with the maximal margin, the thicker layer that can be slipped

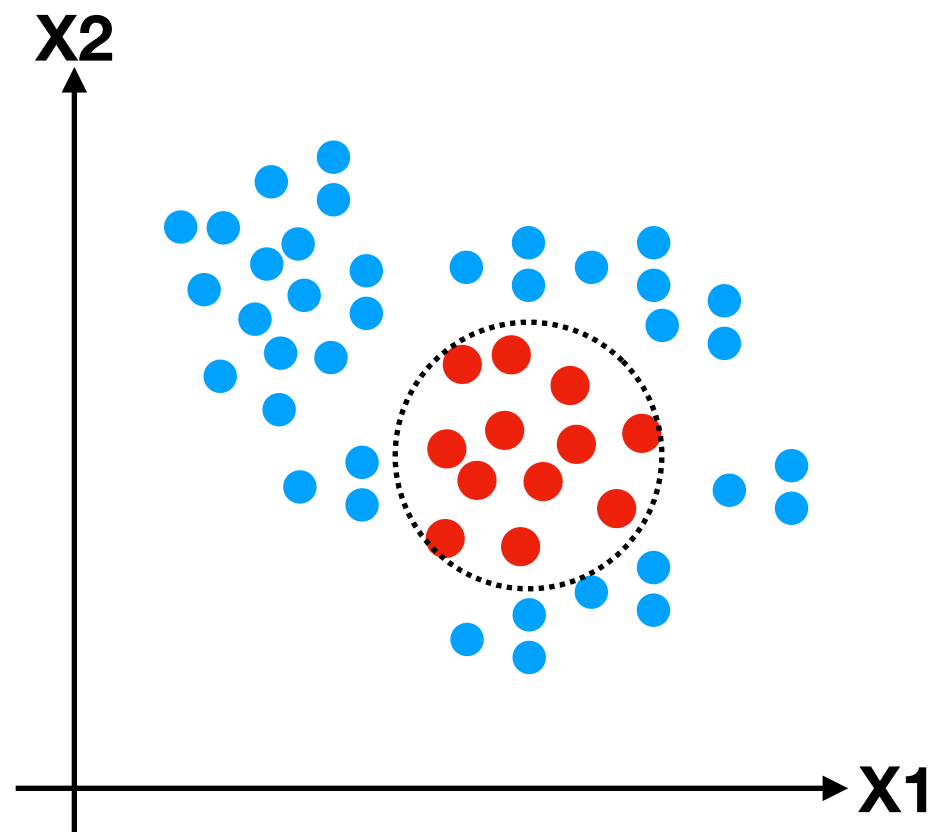
-> This can be applied to non-linear function as well

Support Vector Classifier



What if the problem is not linear?

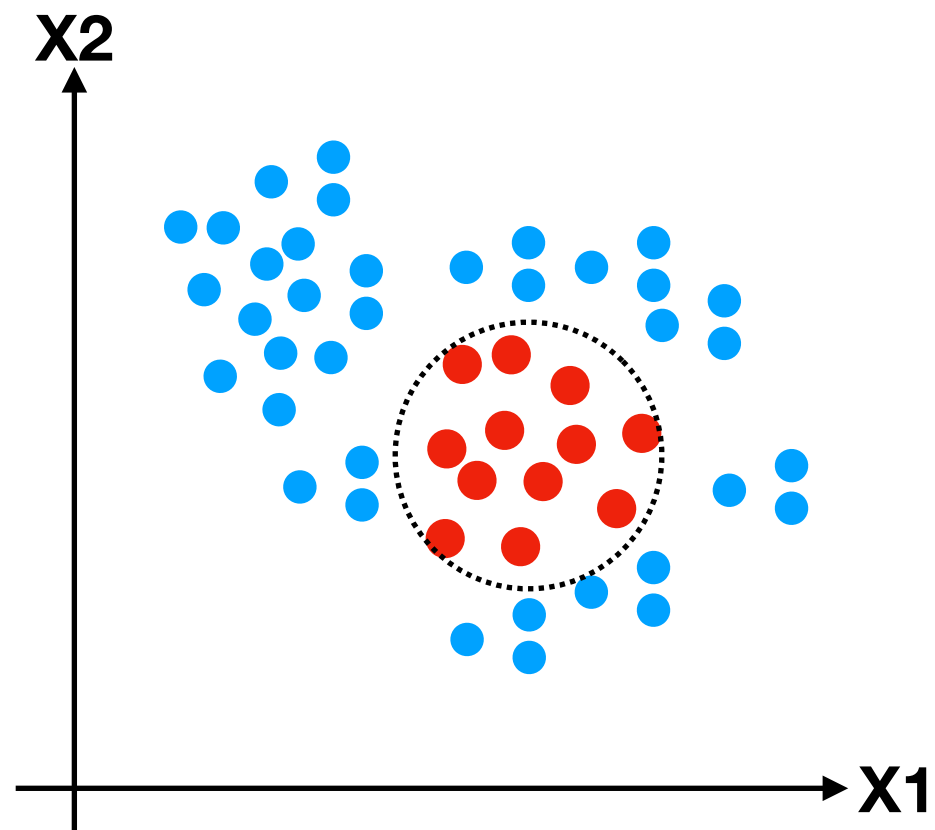
Support Vector Classifier



What if the problem is not linear?

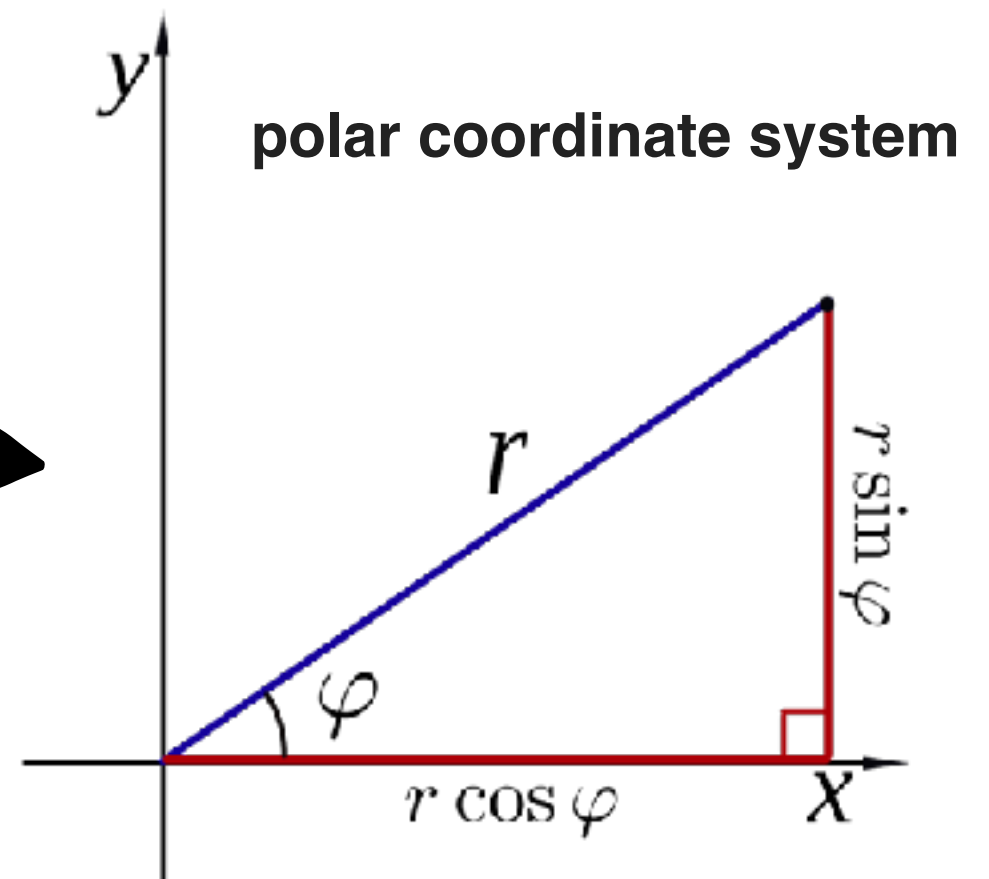
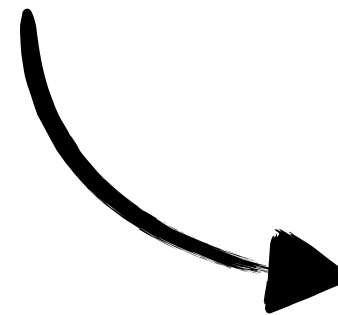
$$X^2 + Y^2 = 1$$

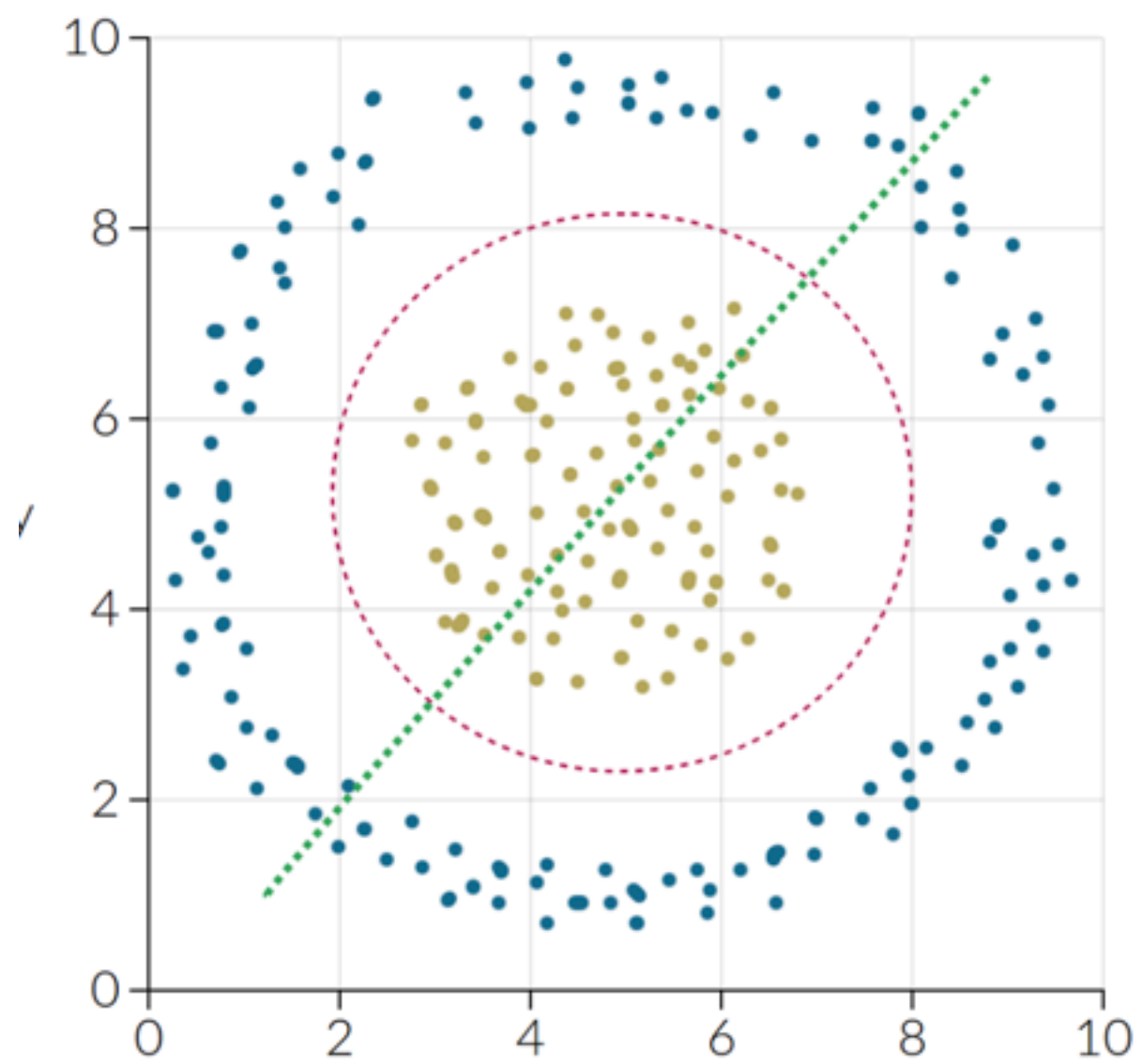
Support Vector Classifier

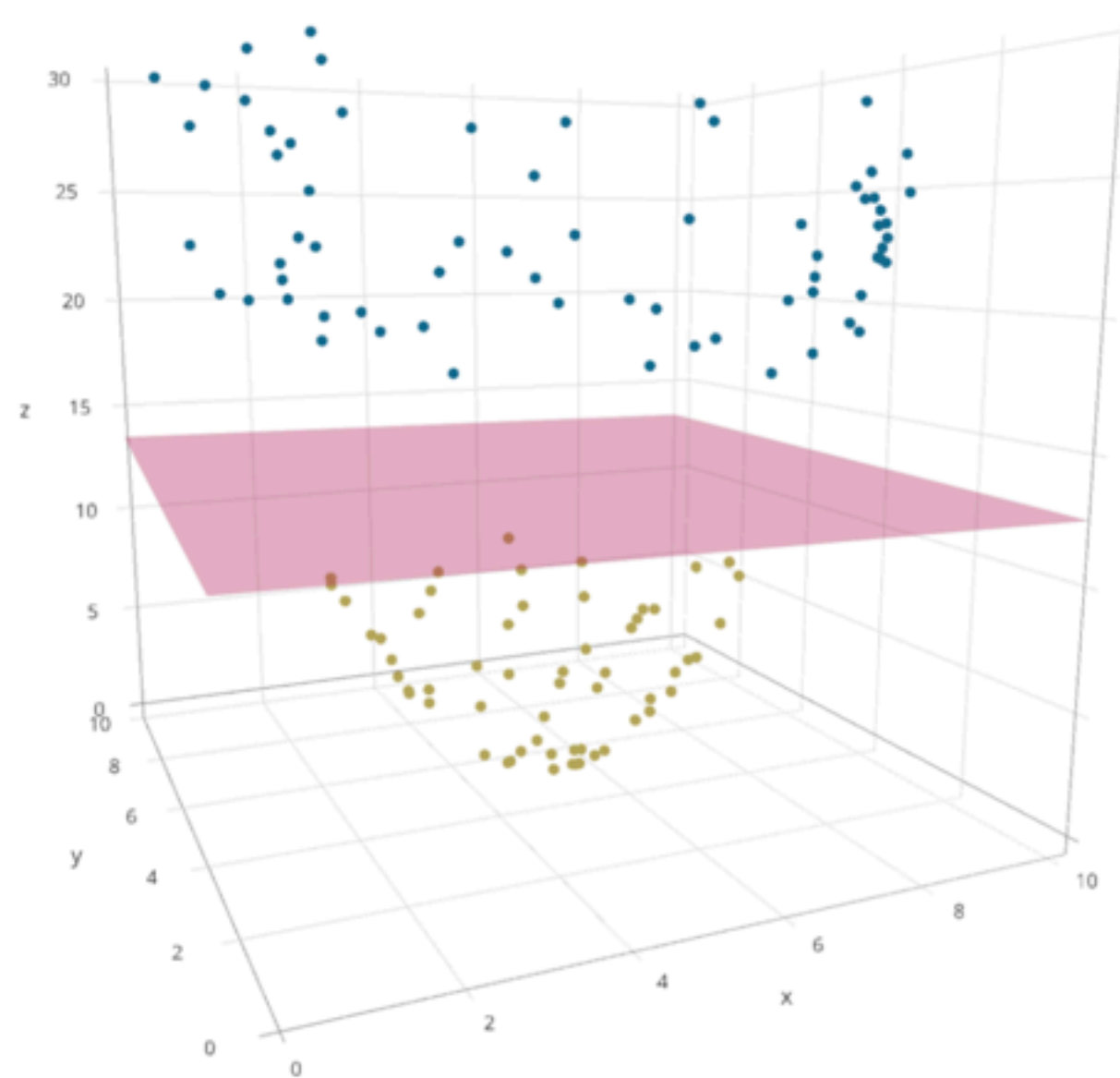
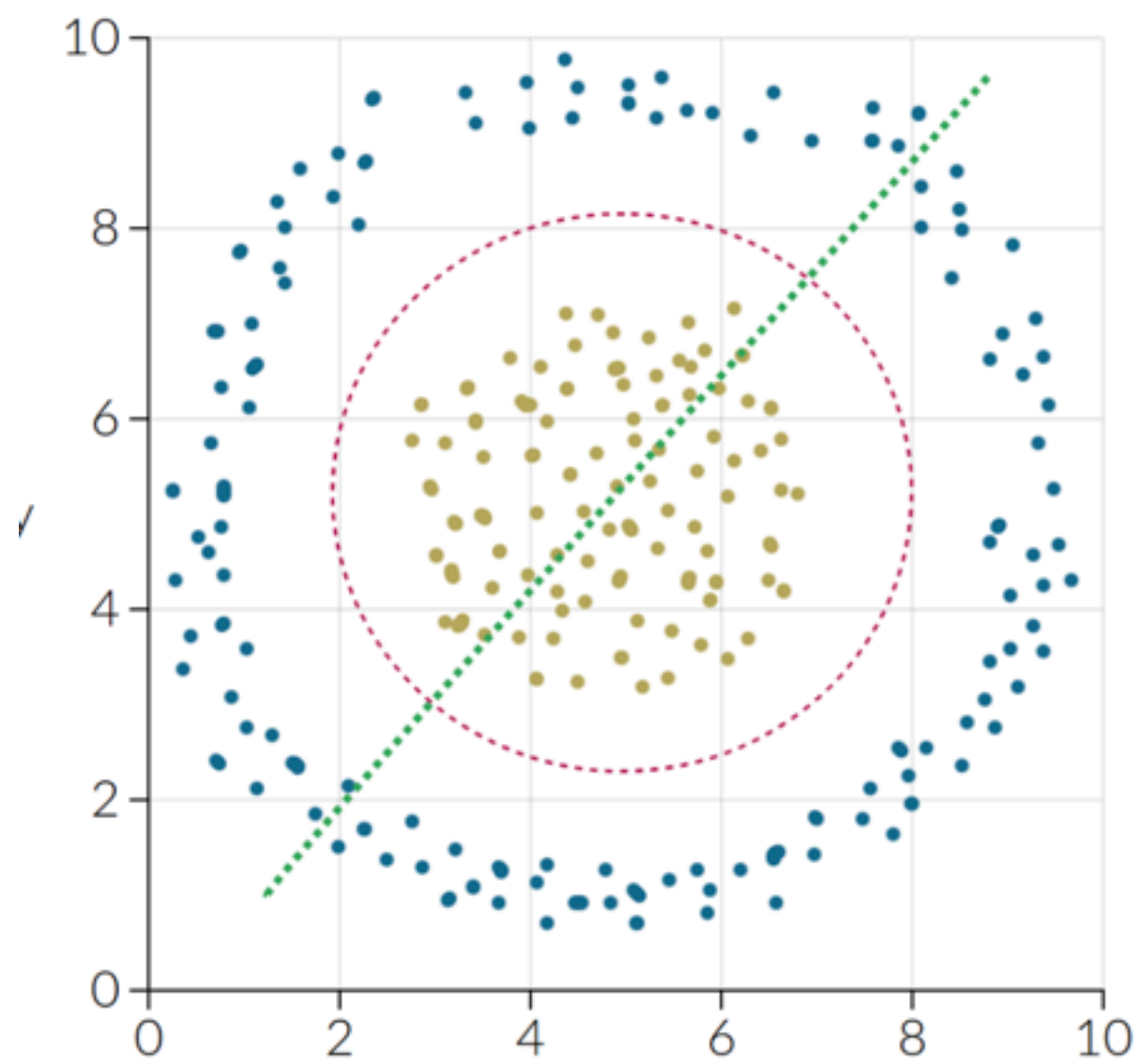


What if the problem is not linear?

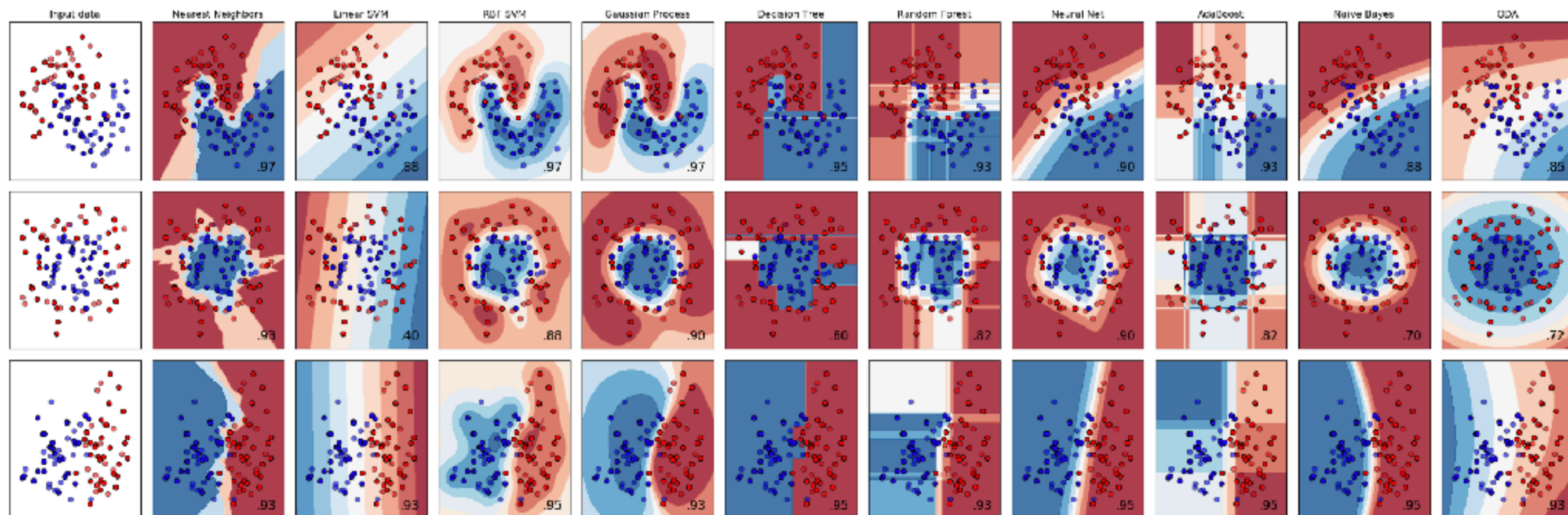
$$X^2 + Y^2 = 1$$

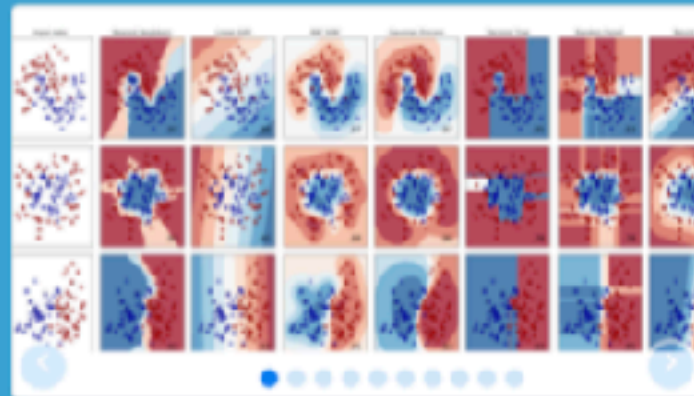






Classifier Zoo





scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

[Examples](#)

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

[Examples](#)

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

[Examples](#)

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency

Algorithms: PCA, feature selection, non-negative matrix factorization, ...

[Examples](#)

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Modules: grid search, cross validation, metrics, ...

[Examples](#)

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction, ...

[Examples](#)

News

On-going development: [What's new](#) (Changelog)

Community

About us See [authors](#) and [contributing](#)

Who uses scikit-learn?



Fitting problem

e.g., N degree polynomial regression

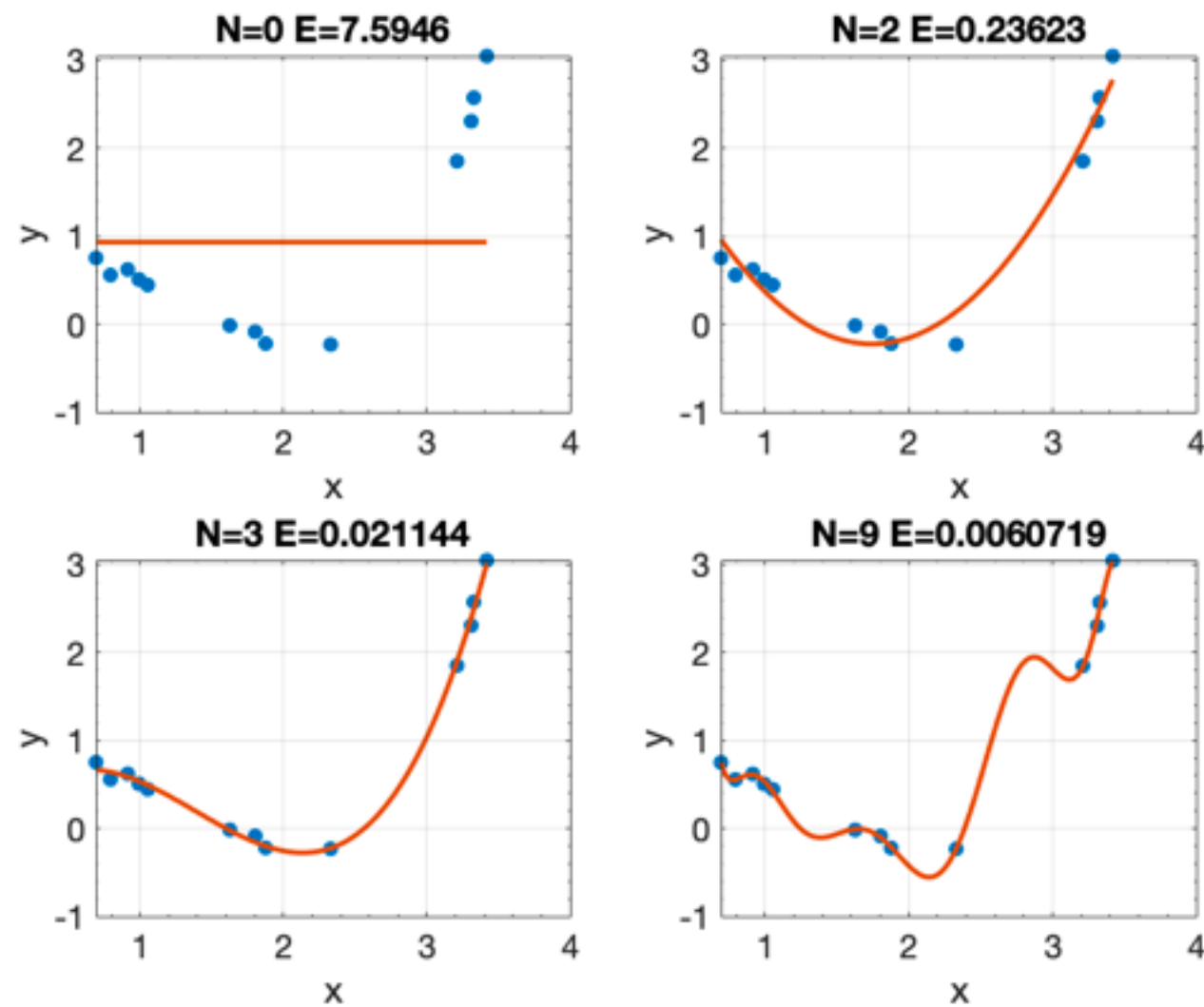
$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Ns^N = \sum_{j=0}^N w_jx^j$$

Least squares of errors

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^I y(x_n, \mathbf{w}) - y_i^2$$

Fitting problem

Evaluation of model prediction over the data

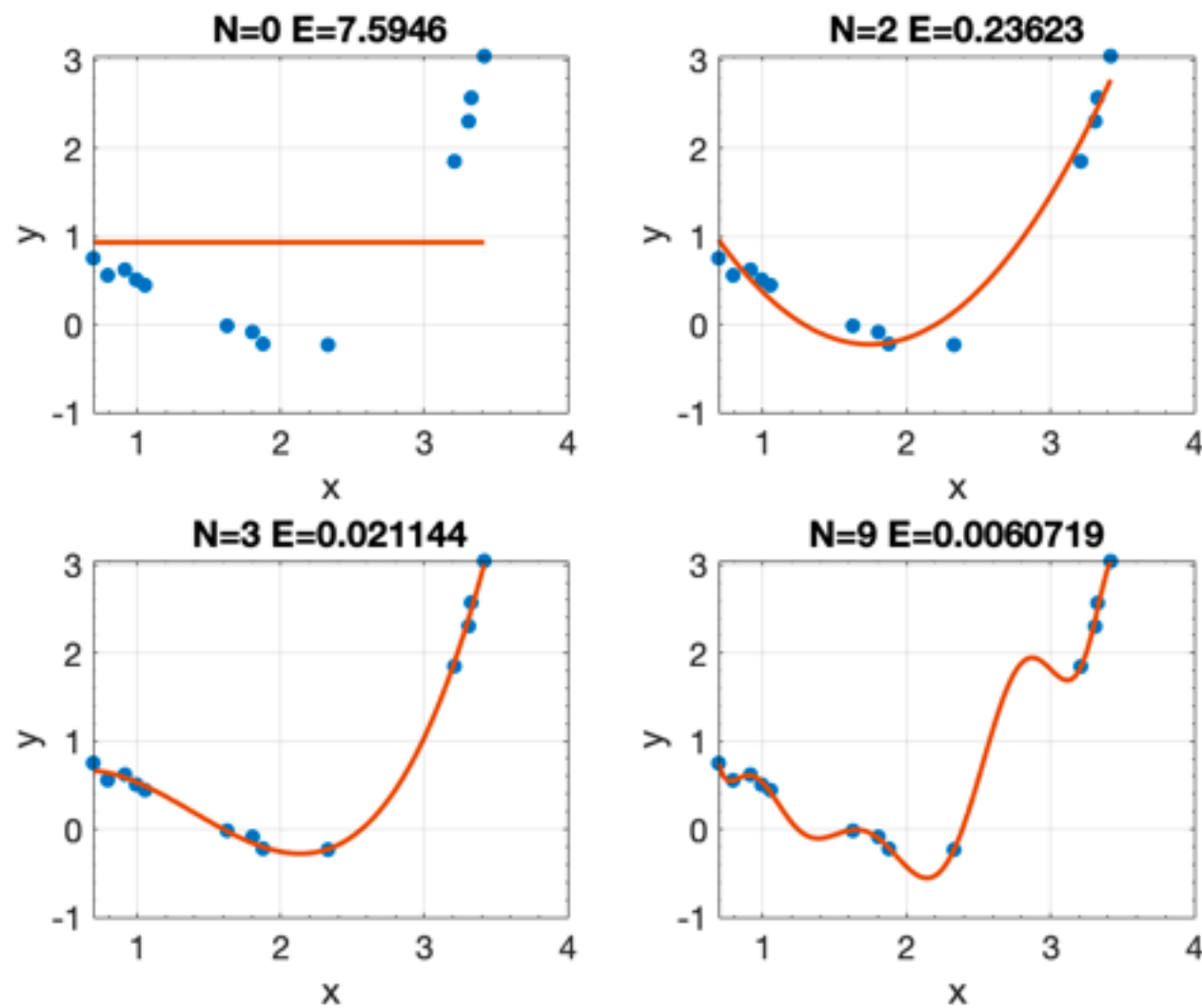


$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Nx^N = \sum_{j=0}^N w_jx^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^I y(x_i, \mathbf{w}) - y_i^2$$

Fitting problem

Which one is the best?

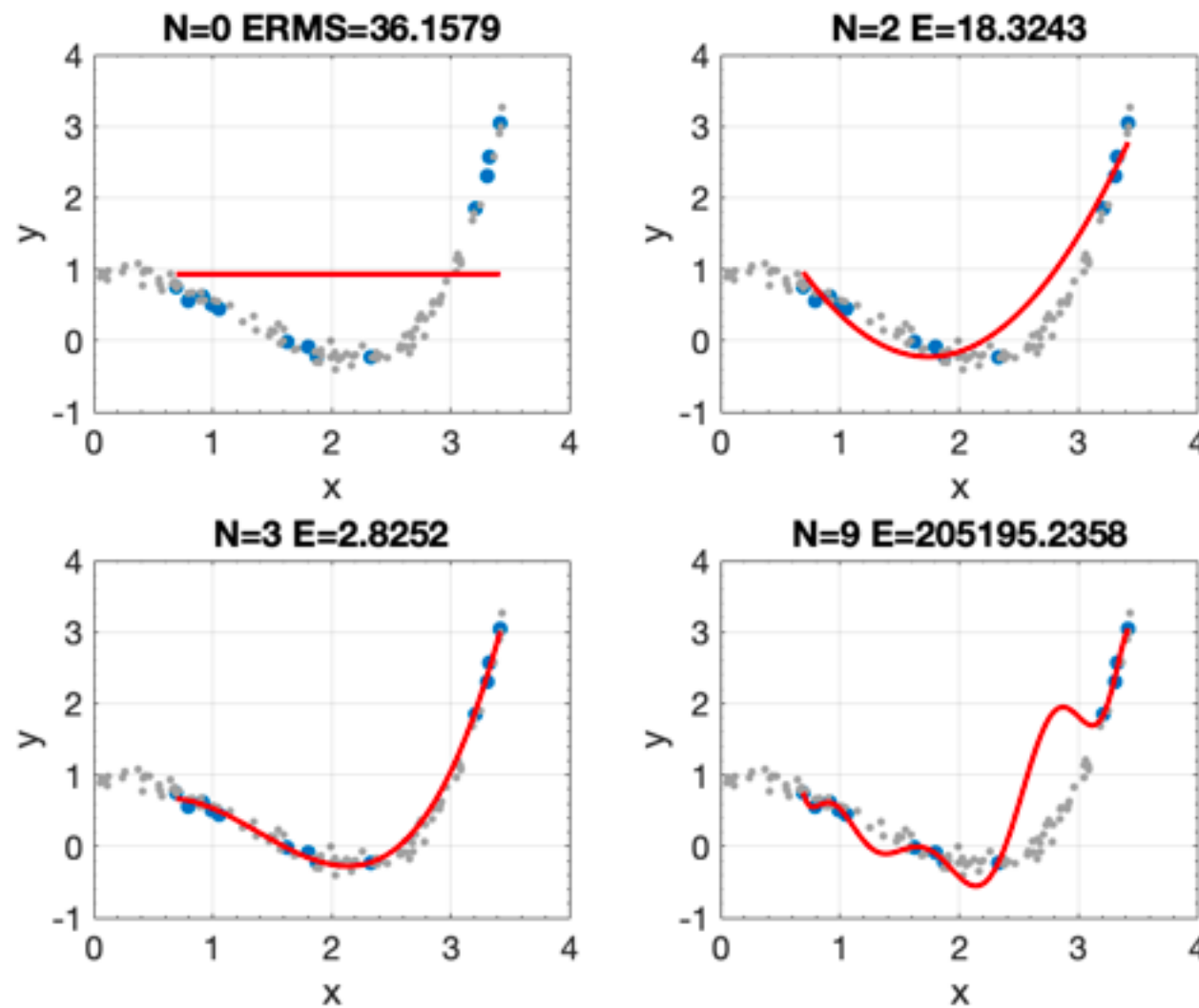


$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Nx^N = \sum_{j=0}^N w_jx^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^I y(x_i, \mathbf{w}) - y_i^2$$

Fitting problem

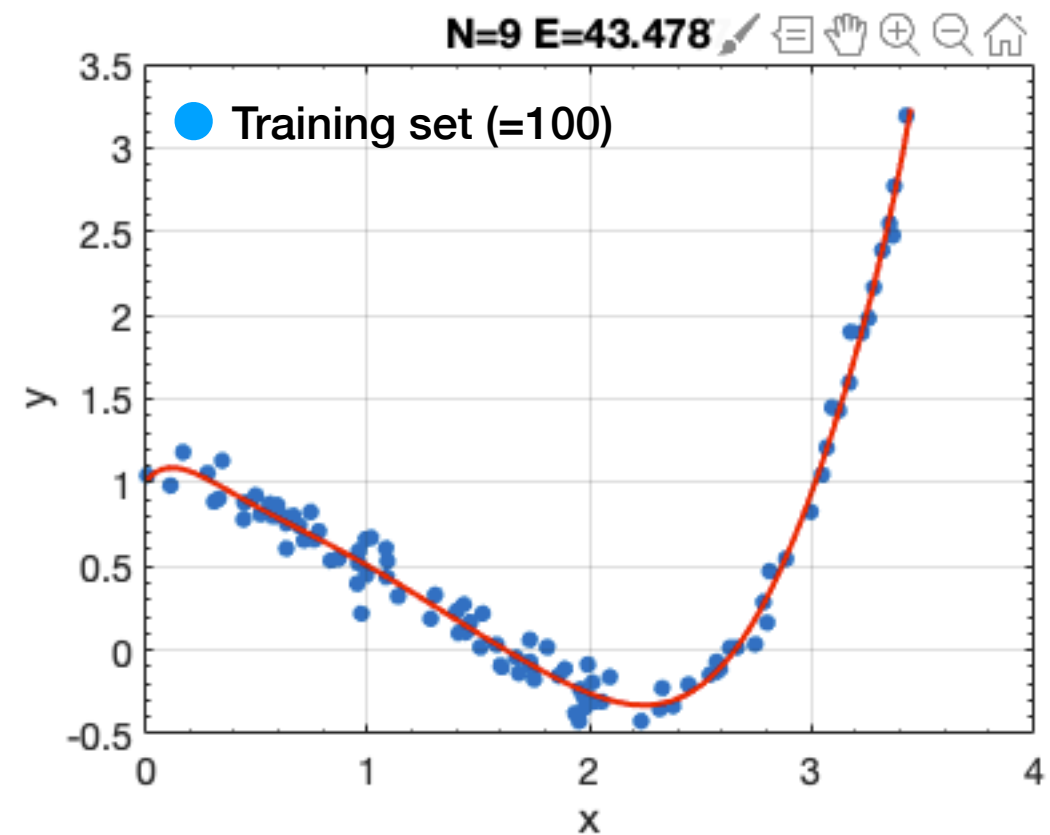
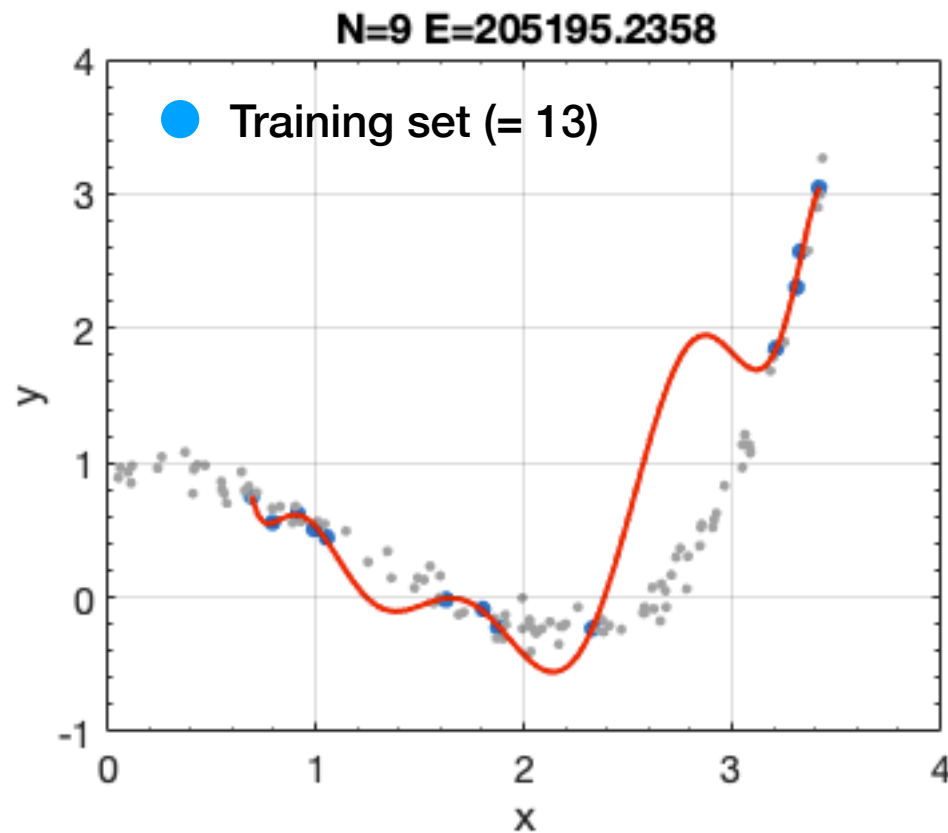
The least square errors is not a good criteria for evaluating the model prediction ability



Fitting problem

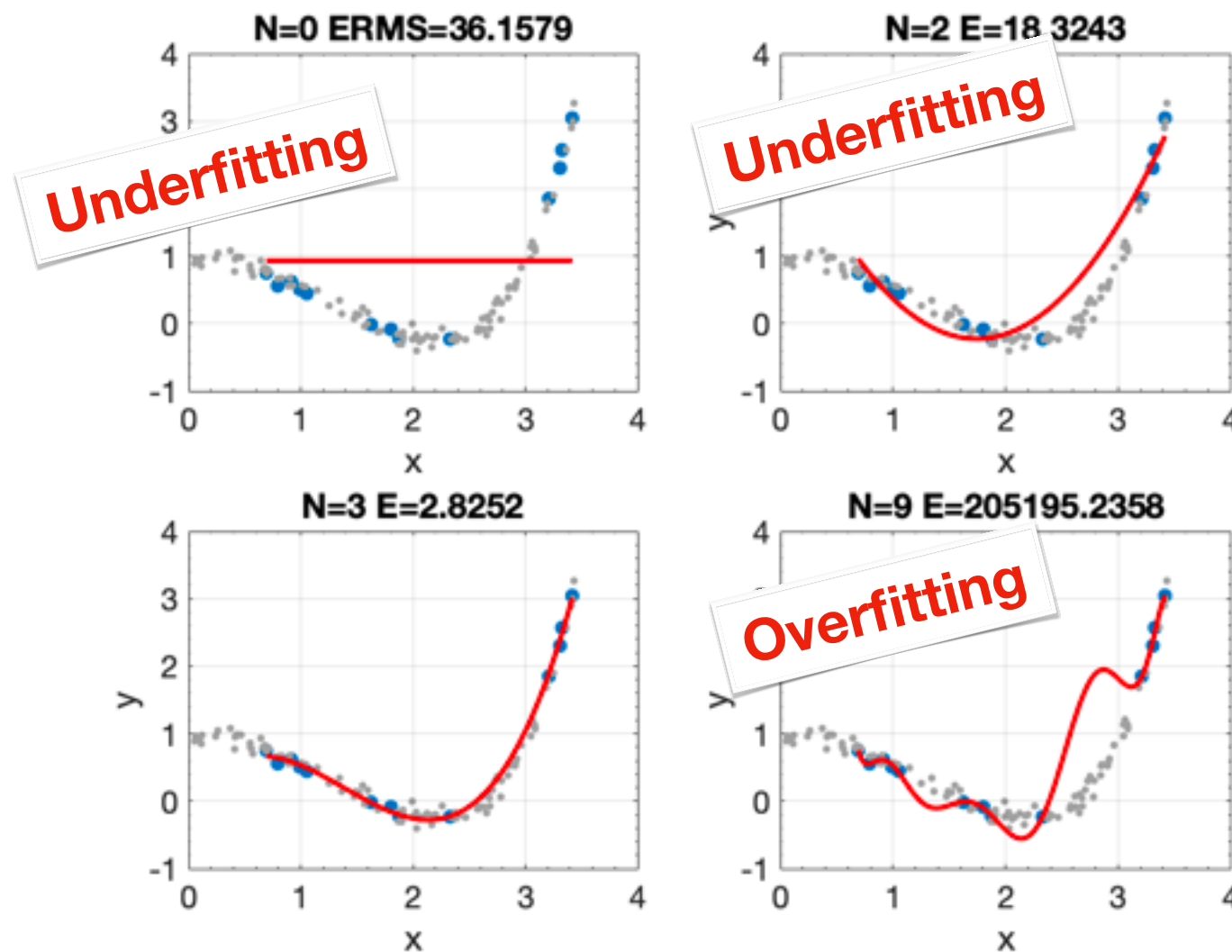
The overfitting problem can be overcome by adding data into the training set
=> Big Data!!!!

Or we provide a priori knowledge

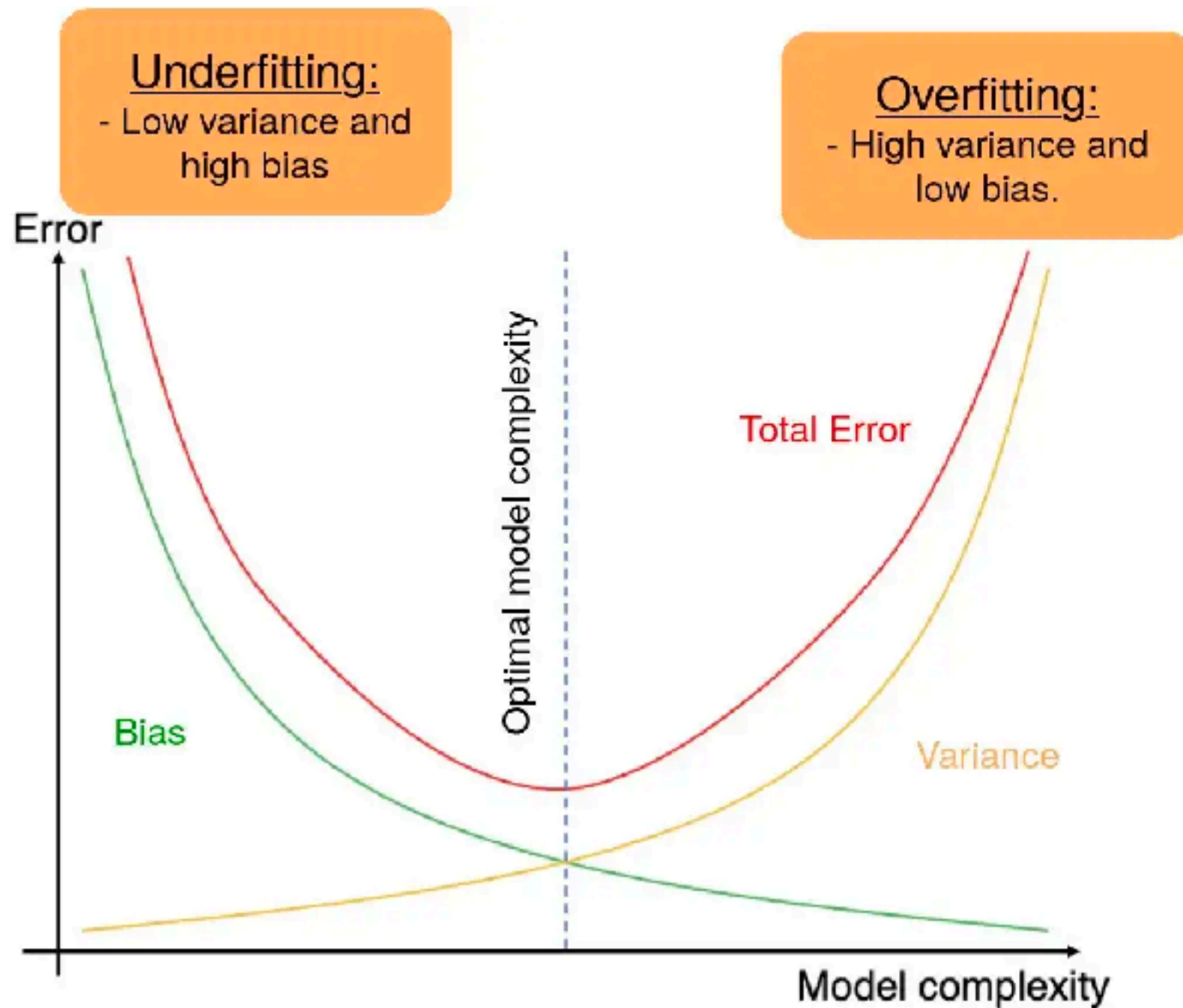


Fitting problem

The least square errors is not a good criteria for evaluating the model prediction ability



Fitting problem



An ideal model is the one with small variance and small bias.

Fitting problem

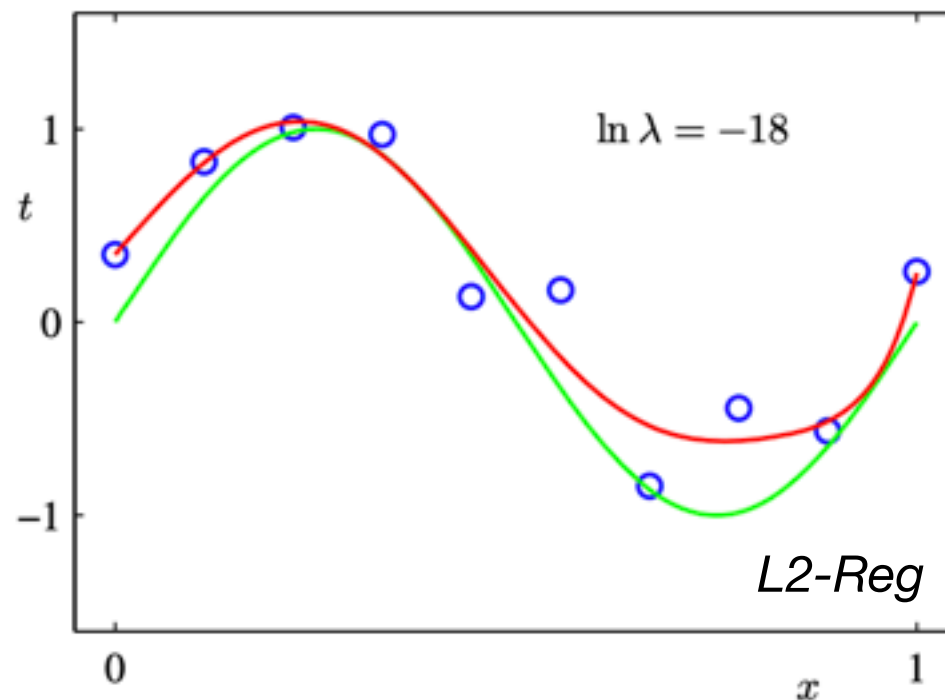
Regularization := adding penalty term

In order to create less complex (parsimonious) model when you have a large number of features in your dataset, some of the Regularization techniques used to address over-fitting and feature selection are

L1 Regularization

Lasso Regression

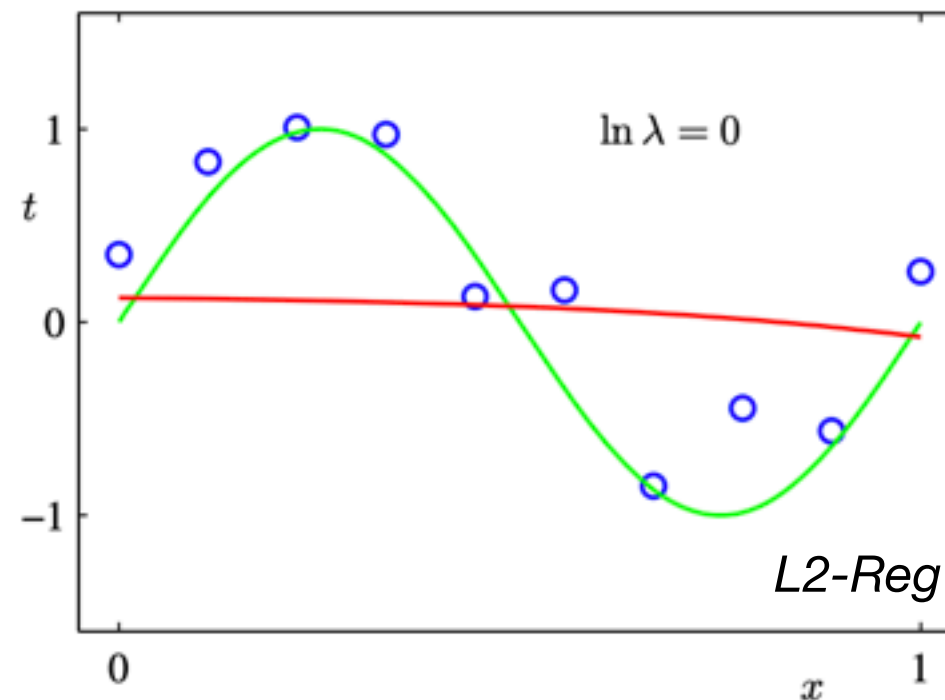
$$\sum_{i=1}^n (y_i - \sum_{j=1}^p x_{i,j} w_j)^2 + \lambda \sum_{j=1}^p w_j^2$$



L2 Regularization

Ridge Regression

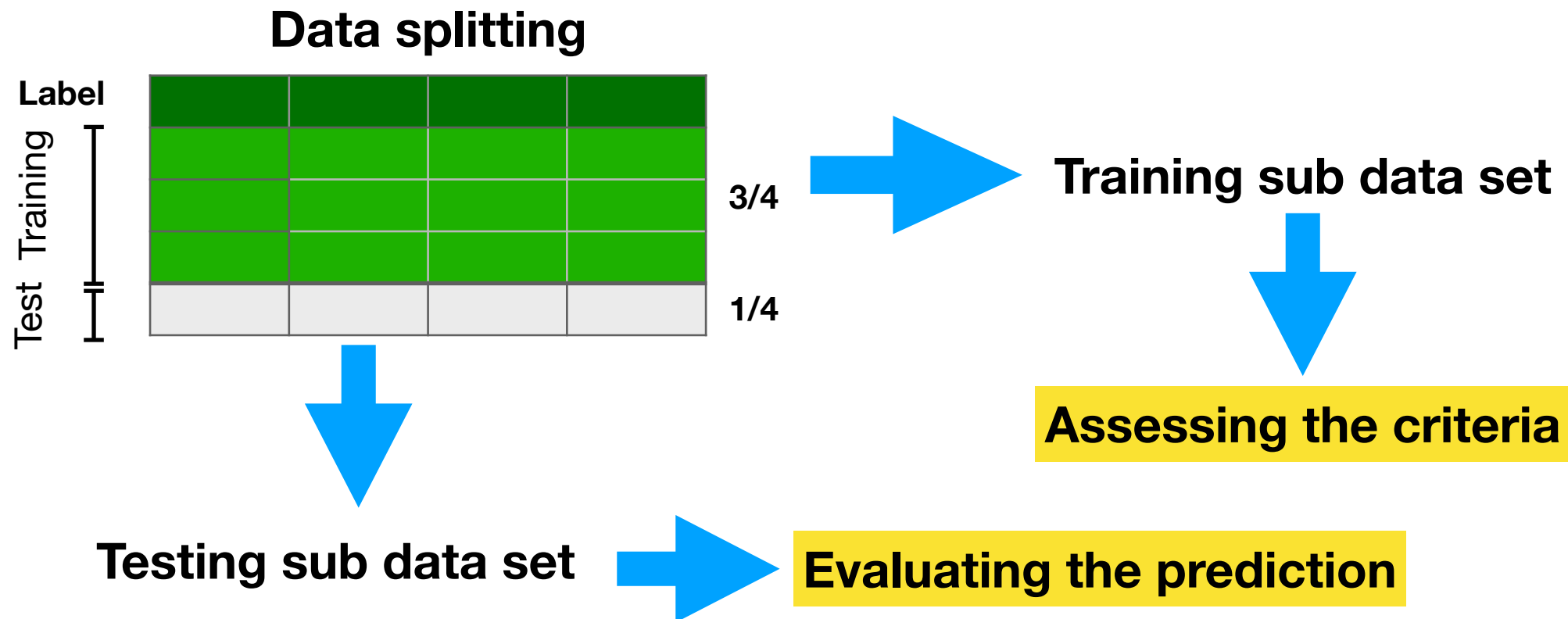
$$\sum_{i=1}^n (y_i - \sum_{j=1}^p x_{i,j} w_j)^2 + \lambda \sum_{j=1}^p \|w_j\|$$



The **key difference** between L1 and L2 techniques is that L1 shrinks the less important feature's coefficient towards zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

Fitting problem

Model selection

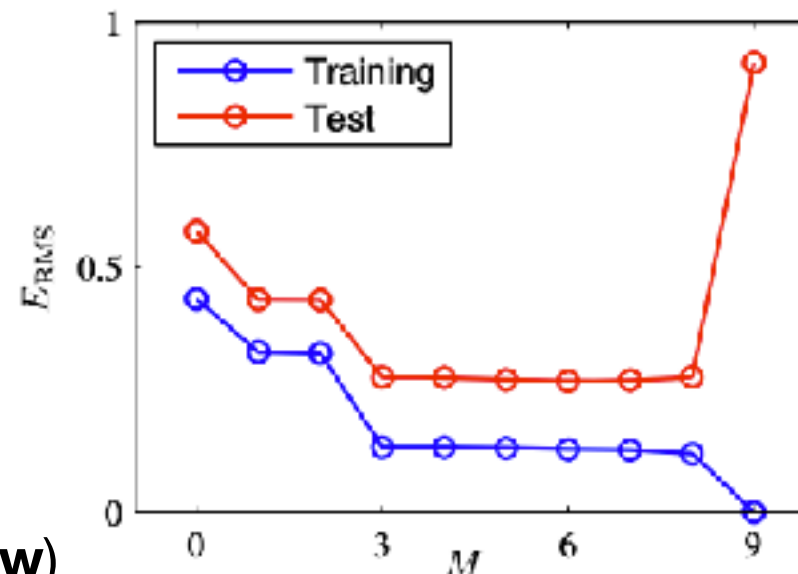


Root Mean Square of Error (RMS)

$$E(\mathbf{w}) = \sum_{i=1}^I y(x_n, \mathbf{w}) - y_i^2$$

$$E_{RMS} = \sqrt{E(\mathbf{w}) / (N - p)}$$

p being the number of parameters (i.e., size of \mathbf{w})

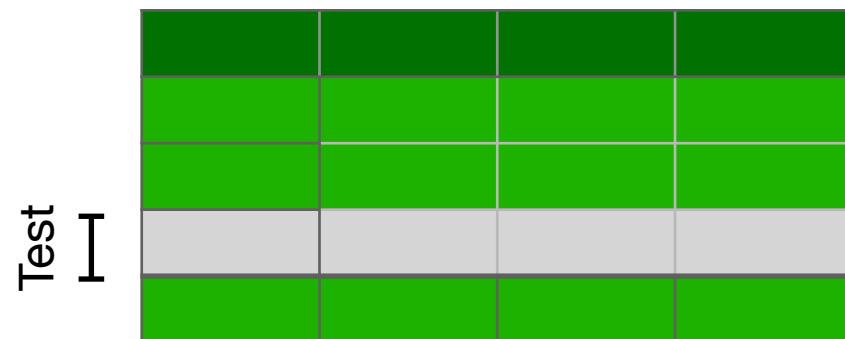


Fitting problem

Model selection

Crossing validation

Data splitting

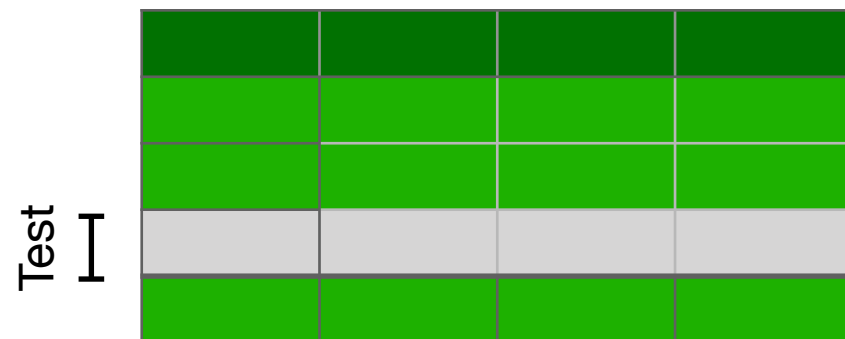


Fitting problem

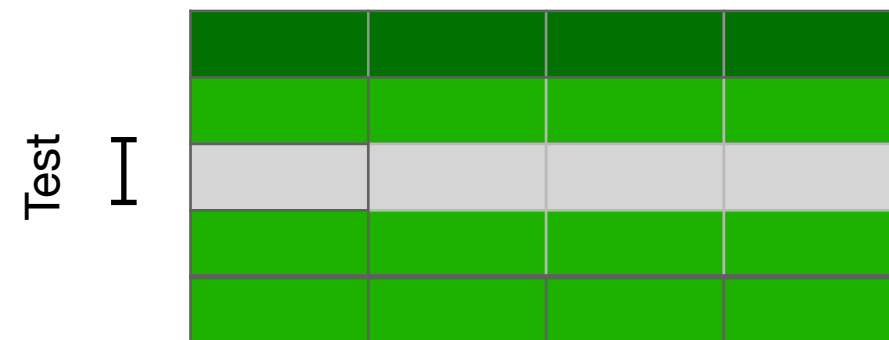
Model selection

Crossing validation

Data splitting



Data splitting

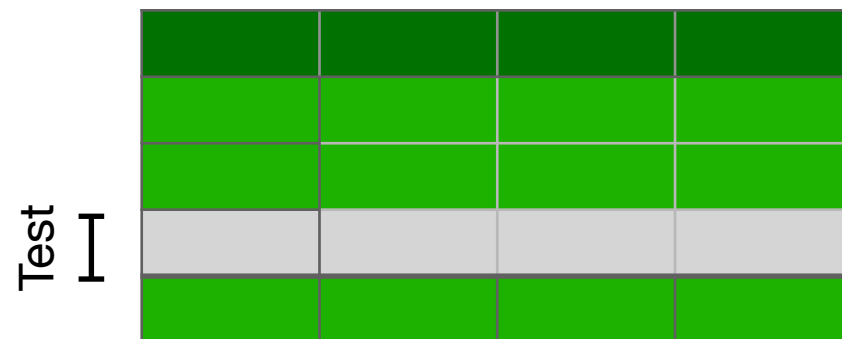


Fitting problem

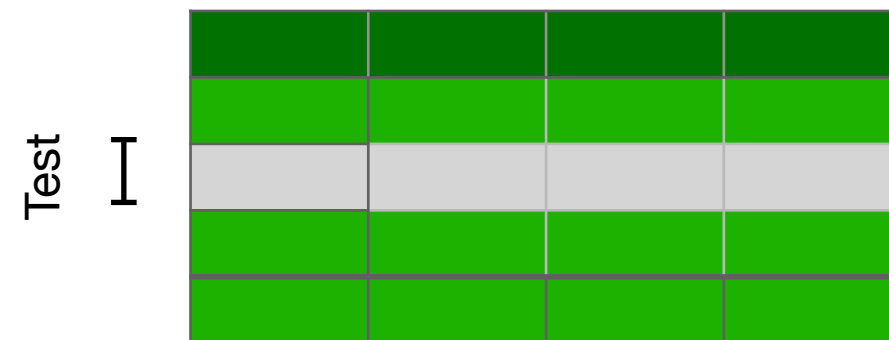
Model selection

Crossing validation

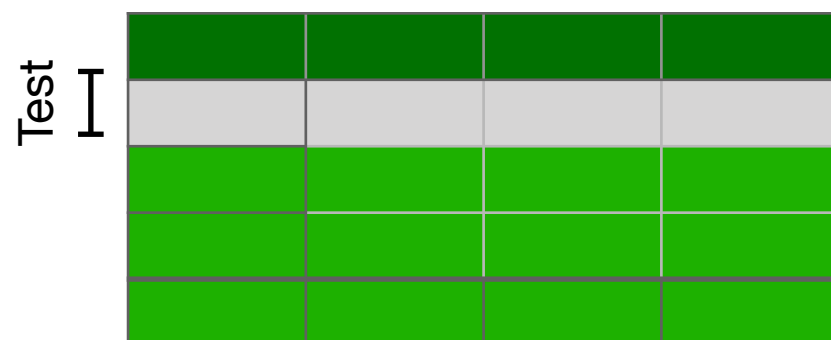
Data splitting



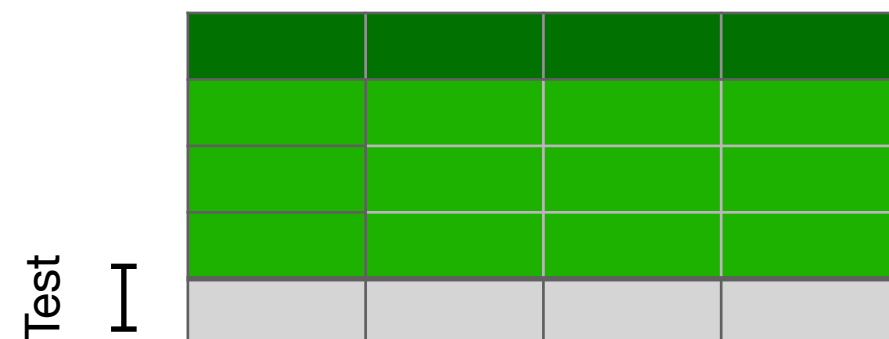
Data splitting



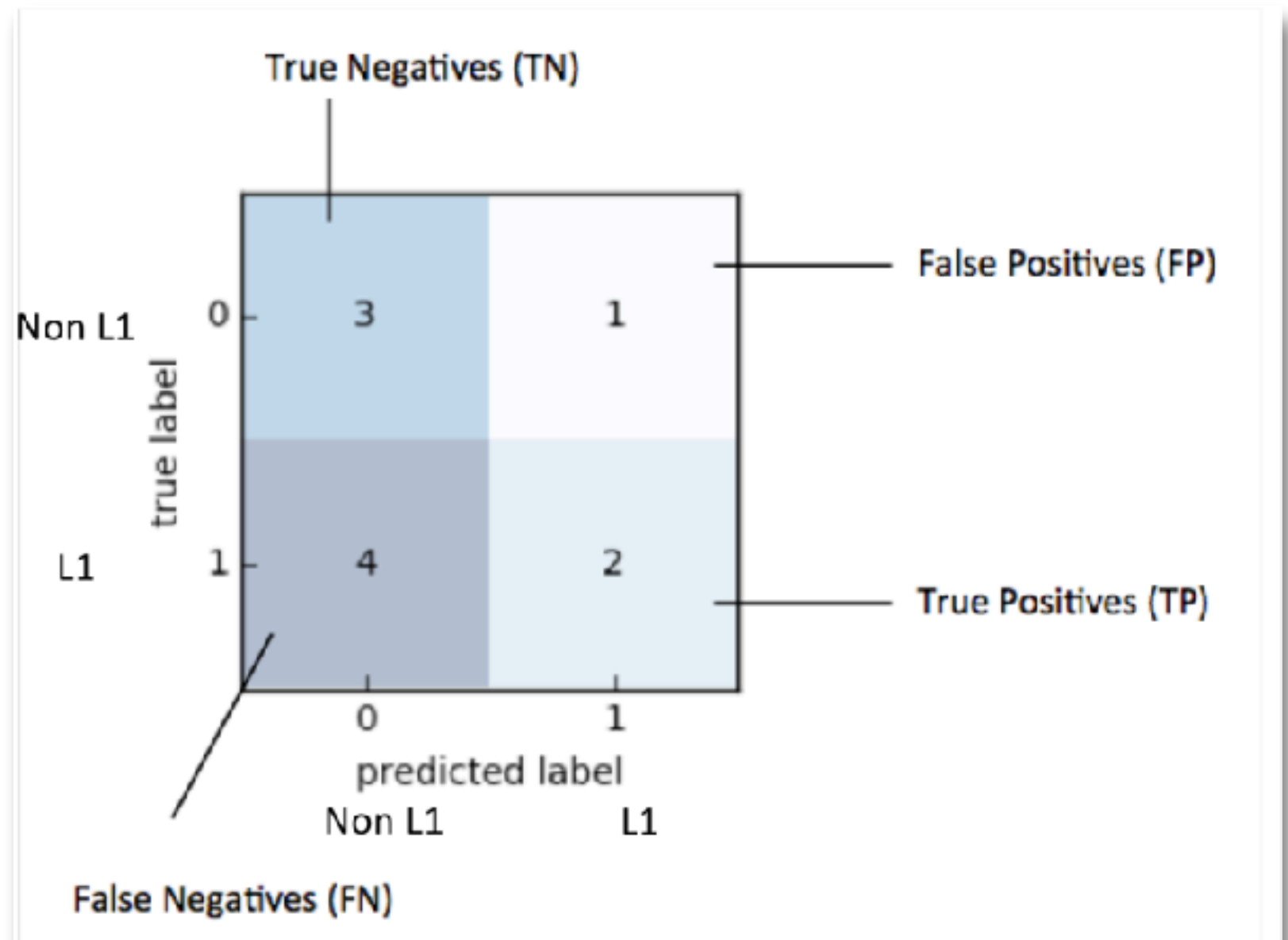
Data splitting



Data splitting



Performance and score



Performance and score

Sensitivity, recall, true positive rate

$$\frac{TP}{TP + FN}$$

Specificity, true negative rate

$$\frac{TN}{TN + FP}$$

Precision

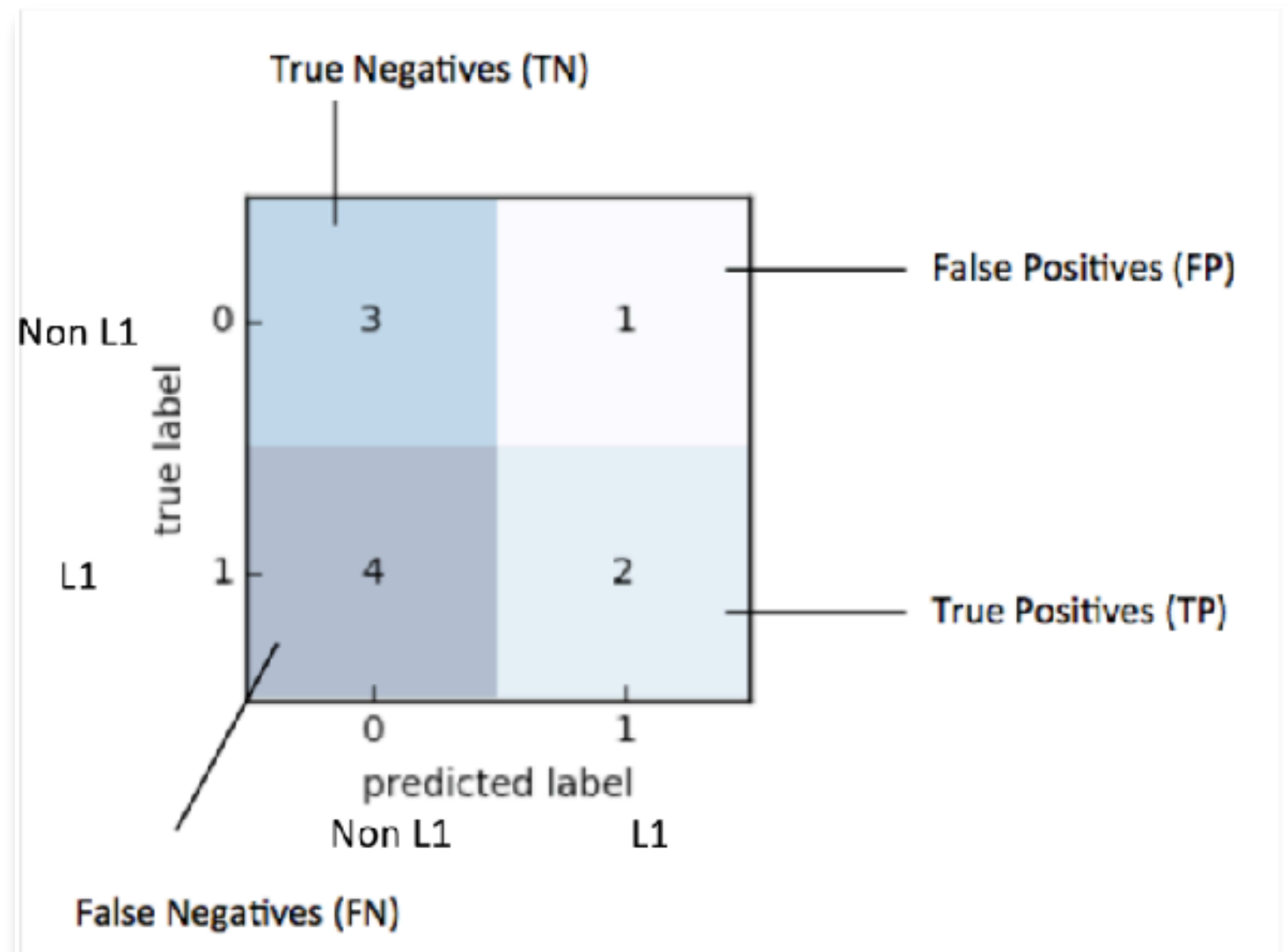
$$\frac{TP}{TP + FP}$$

Critical Success Index

$$\frac{TP}{TP + FN + FP}$$

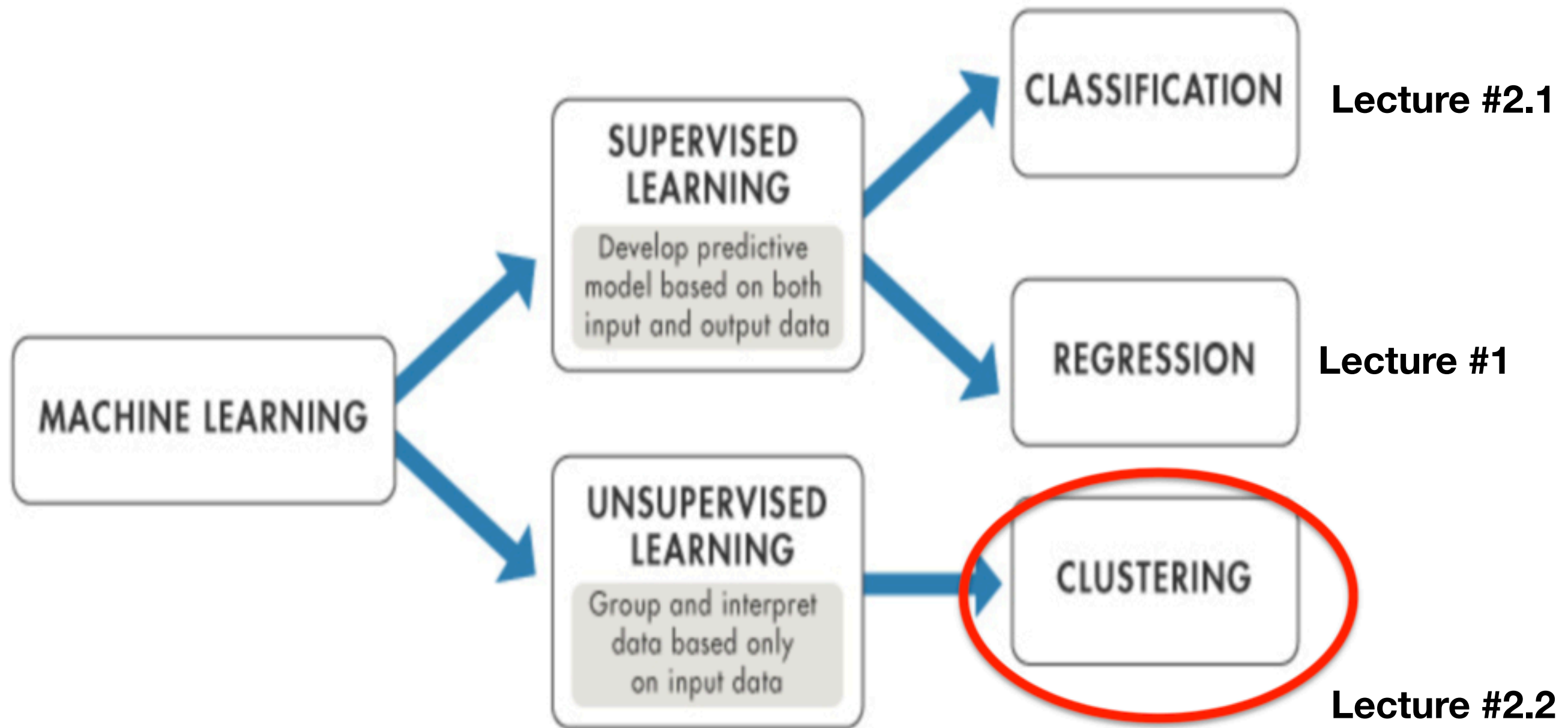
Accuracy, classification rate

$$\frac{TP + TN}{TP + TN + FP + FN}$$



How to proceed ?

- Fit on learning zone
- predict on test zone and estimate performance.



Clustering and the like....

Grouping : building objects collection

- sharing similar properties (in the considered feature space)
- non-sharing similar properties when not belonging to the same group

Clustering is an unsupervised classification (no predefined classes)

Similarity (distance in the feature space)

e.g., Minkowski distance

$$d(i, j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ et $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional objects
with q being a positive integer

Criteria of clustering

- Partitioning: Dividing the population, and evaluate in regard to criteria
- Hierarchical: Decomposition over criteria
- Density: Connectivity in the space feature
- Grid: multi-level of granularity
- Model-based: a priori rules driven from a given model

Partitioning clustering

Create partition of k clusters following criteria

K-mean (MacQueen' 67) : each cluster is defined from its centroid

K-medoids (Kaufman & Rousseeuw 87) : each cluster is defined from one object

K-means algorithm

1. Define k (number of clusters)
2. Assigning each object O to cluster C_i with M_i centroid as we minimise $\text{dist}(O, M_i)$
3. Compute barycentre of M_i for each cluster
4. Goto #2 in case of a new assignment

Strength

Efficient towards $\mathcal{O}(tkn)$

n : object #

k : cluster #

t : iteration #

as $k, t \ll n$

Weakness

Not applicable if features are not interval/ratio (see lecture #1)

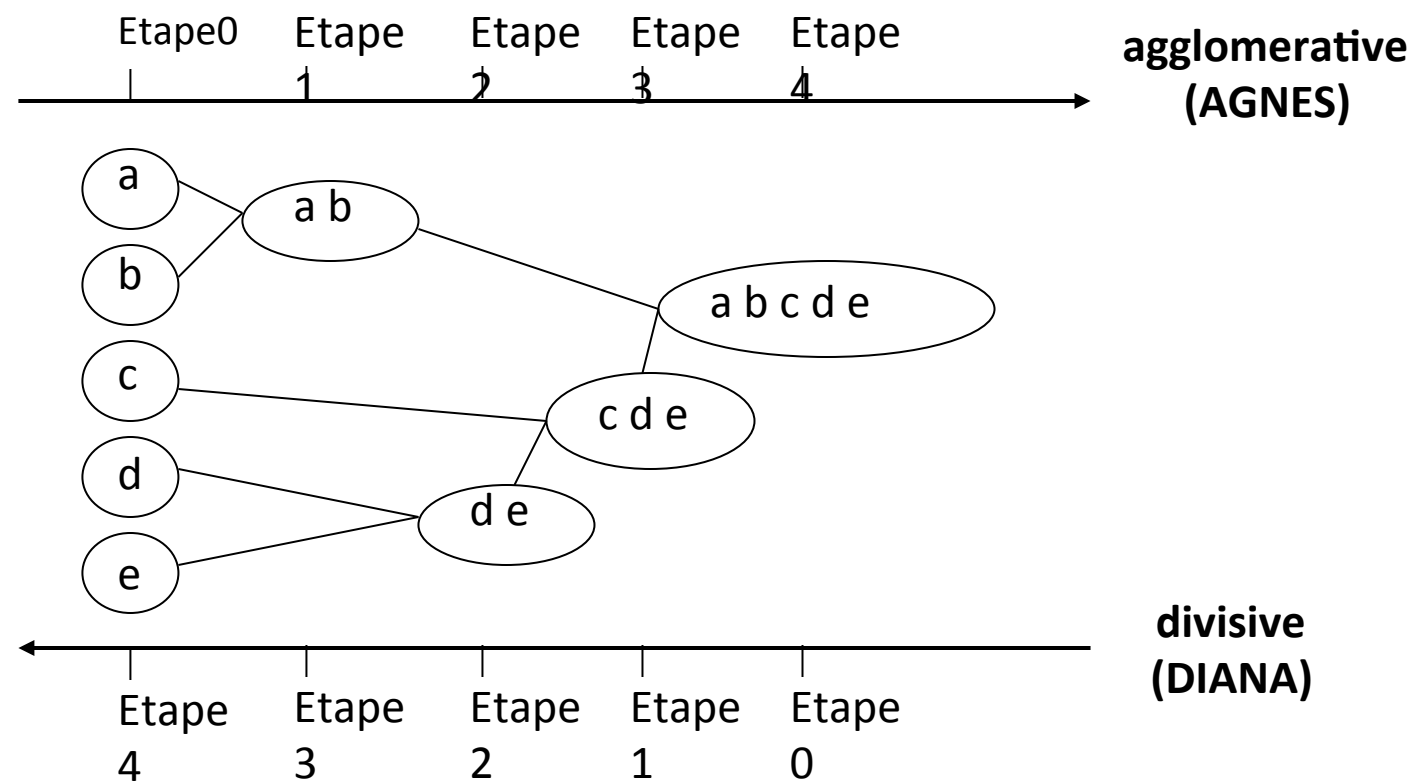
k -cluster needs to be defined

Not suitable for non-convex hull

Hierarchical clustering

We compute the distance matrix sa clustering criteria.

K is not defined but we need a stop condition.



Density clustering

Dense cloud vs parse areas (i.e., noise)

Two parameters

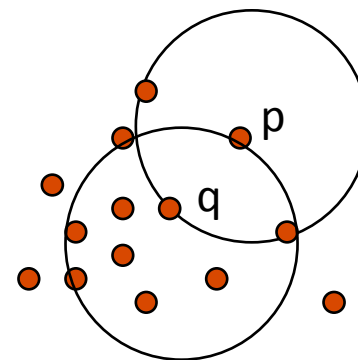
Eps: neighbours maximum radius

MinPts: Minimal number of points in the **Eps** neighbours radius

$$V_{Eps}(p): \{q \in D \mid dist(p,q) \leq Eps\}$$

p Belongs to neighbours V_{esp} if

- 1) $p \in V_{Eps}(q)$
- 2) $|V_{Eps}(q)| \geq MinPts$



MinPts = 5

Eps = 1 cm

Strength

Work with non-convex hull

Supervised

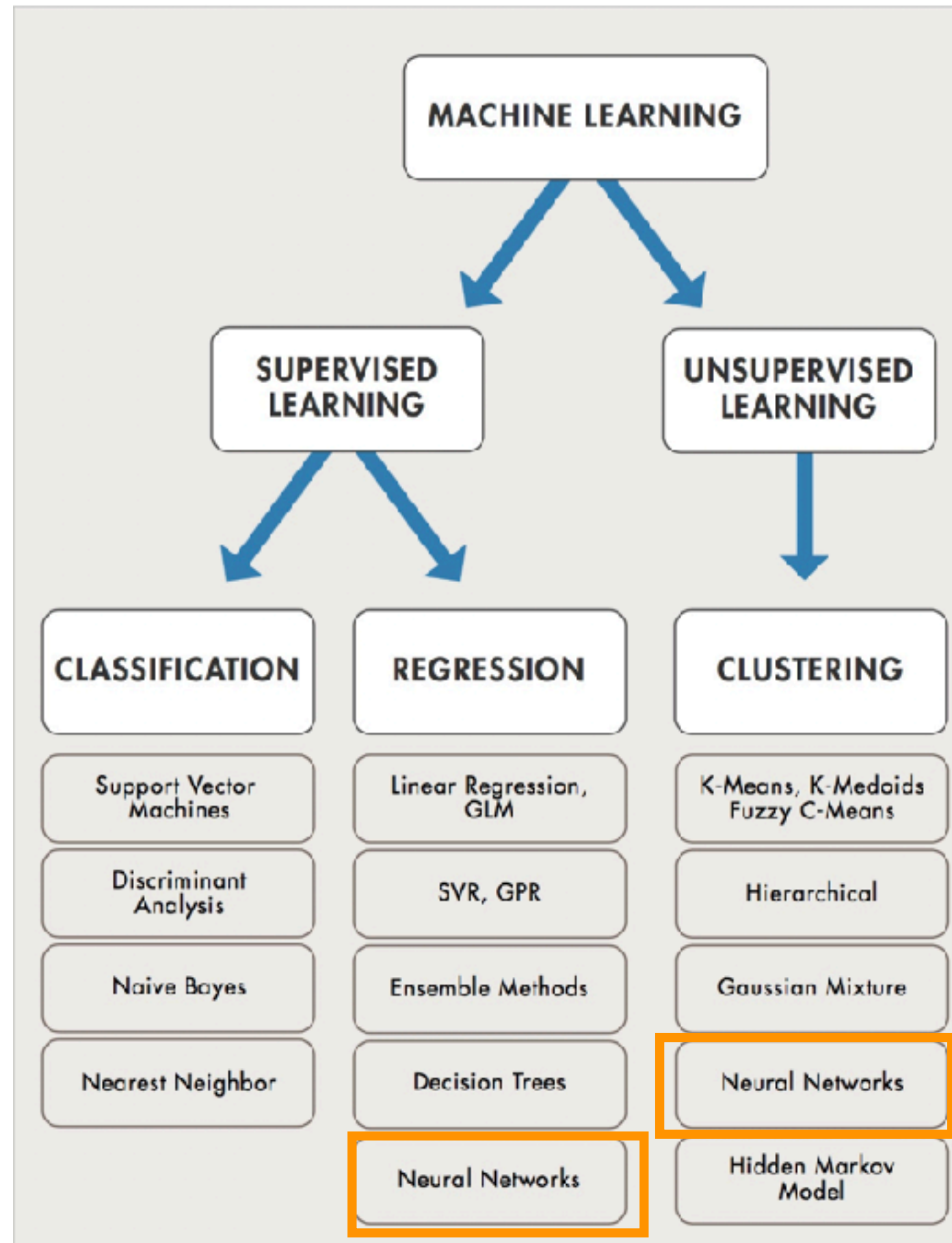
Training a model on input and output data known exits so that he can predict the exit on future entries. An a priori the data set of learning

Unsupervised

Finding forms, an intrinsic structure in the data without *a priori*

Lecture #2.3

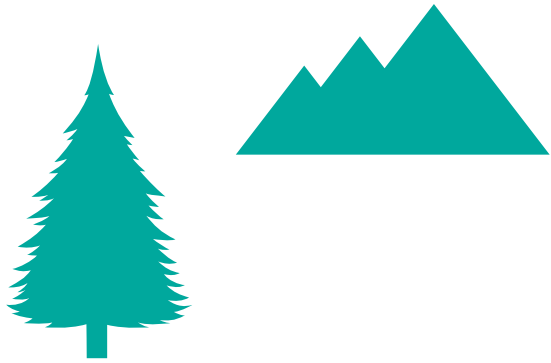
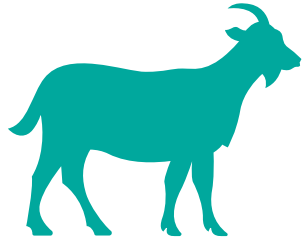
with Greg in a few



Deep learning is a subfield of Machine learning aiming at data representation.

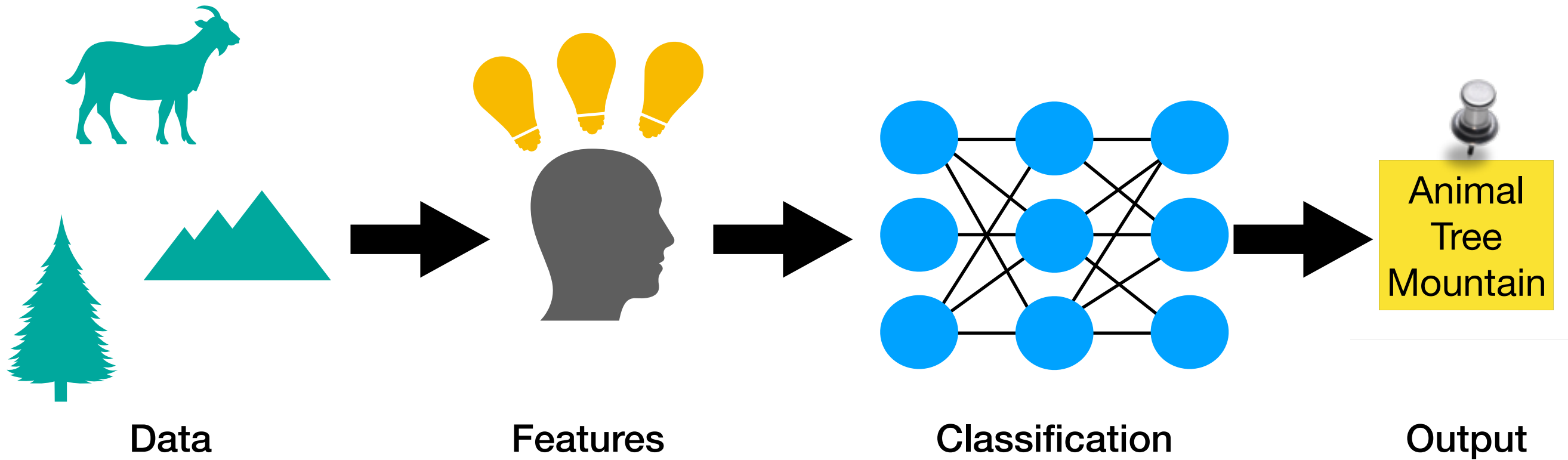
DL algorithms attempt to learn a representation by using a hierarchy of multiple layers (hidden layers) based on a neural network

Machine learning is:

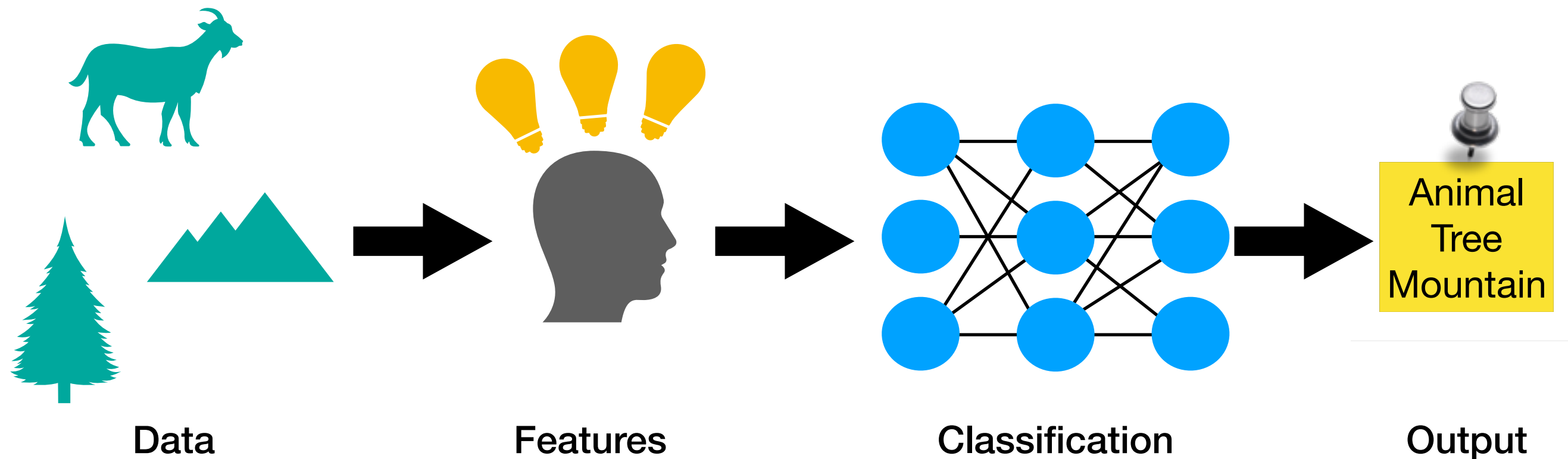


Data

Machine learning is:



Machine learning is:



Deep learning is:

