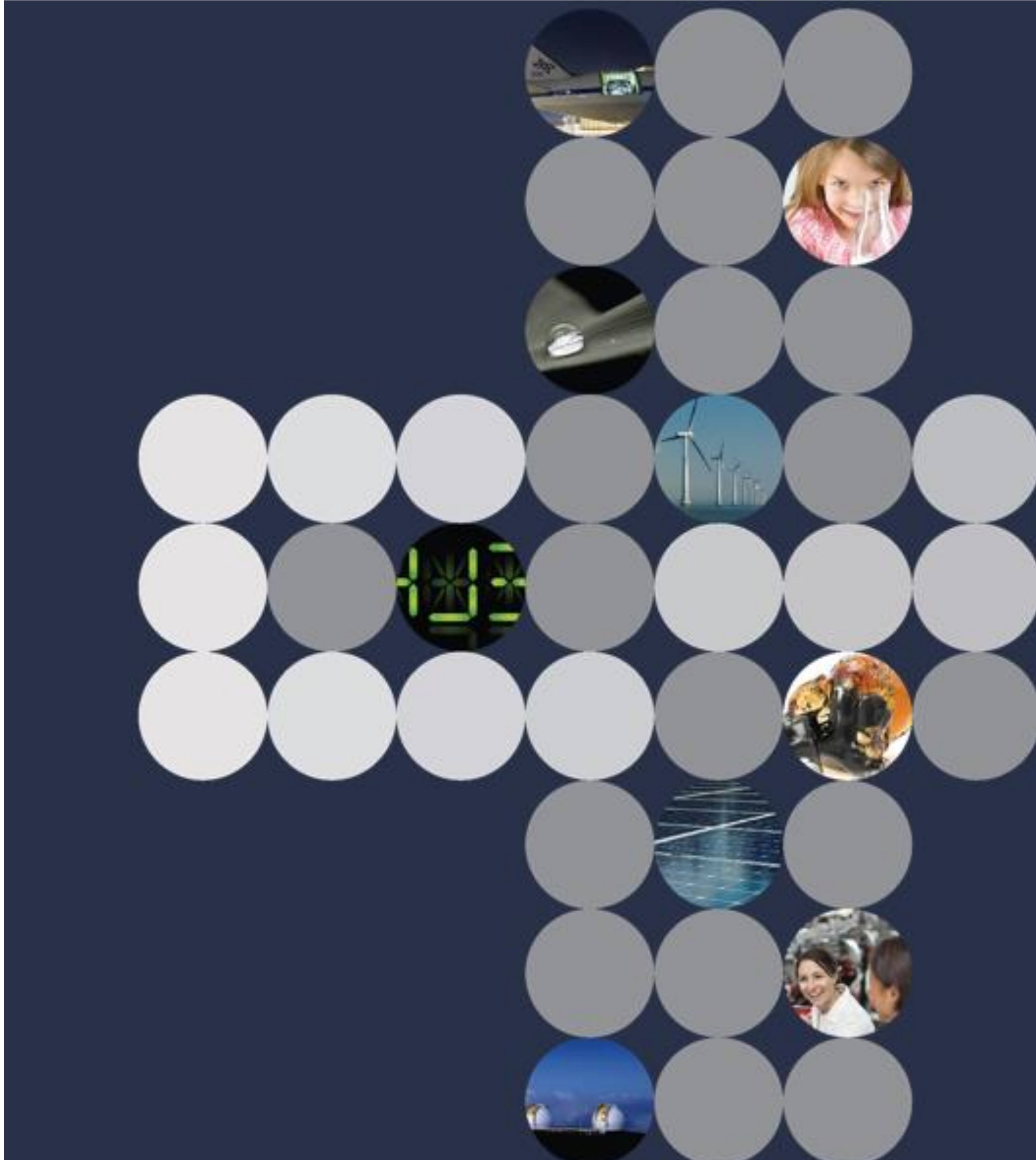
# EE512 – Applied Biomedical Signal Processing

## Regression and Classification

Ramin Soltani(asi@csem.ch)
CSEM Signal Processing Group



# Outline

- Introduction to Machine Learning (ML)
- Regression
- Classification
  - Supervised
  - Unsupervised
- Regularization
- Feature selection
- Validation strategies
- Performance evaluation

# Introduction to Machine Learning

- **Goal:** "teach" a machine to interpret data
- **How:** Training a model with a data subset
- **Usage:** The trained model can then interpret new data



**Data collection**
- PPG
- ECG
- Accelerometer

**Feature engineering**
- Filtering
- Time
- Frequency
- Statistics

**Feature selection**
- Univariate analysis
- PCA
- LASSO

**Model training**
- Decision tree
- Support vector machine
- Bayes

**Model deployment**
- Embed in a device

# Introduction to Machine Learning

## Supervised (known labels)

- Support vector machine
- Linear regression
- Logistic regression
- Naïve Bayes
- Decision trees
- Neural networks

## Unsupervised (unknown labels)

- K- means
- Gaussian Mixture Model
- DBScan

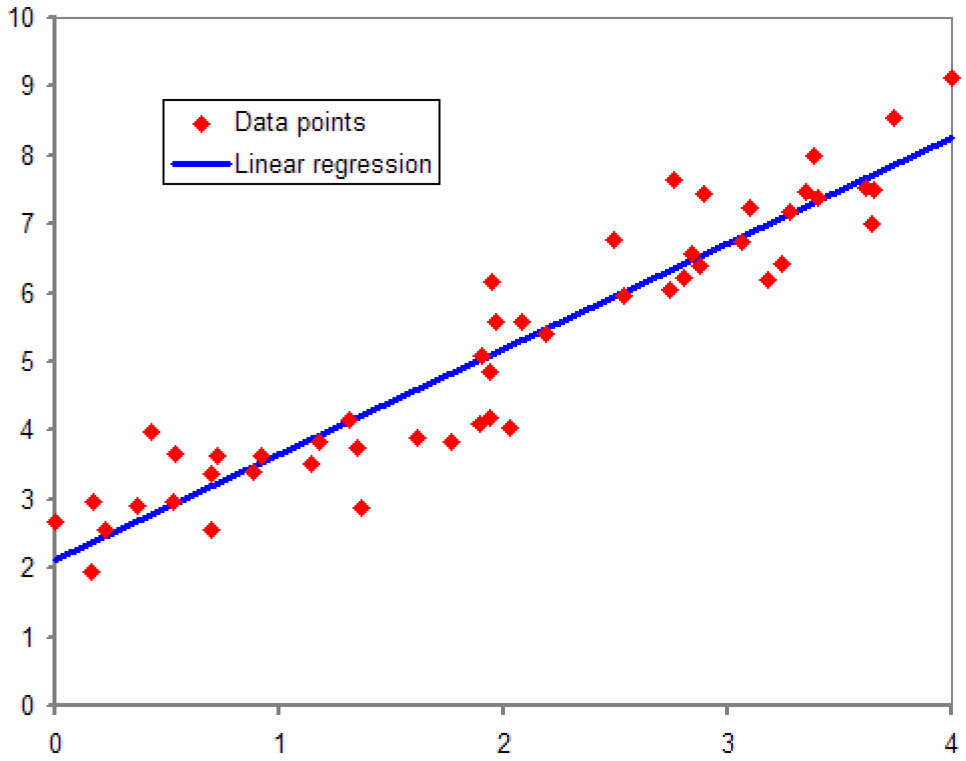
## Reinforcement learning

- Q-Learning
- Markov decision process
- Monte Carlo

# Regression vs Classification

- **Regression**
  - Goal: estimate an output **value**
  - Usage: Mainly for **curve fitting**
- **Classification**
  - Goal: predict an output **class**

# Regression



# Regression

- Estimate the dependence between a dependent variable (the output, Y) and 1 or more independent variables (the inputs, X).

$$Y_i = f(X_i, \beta) + e_i$$

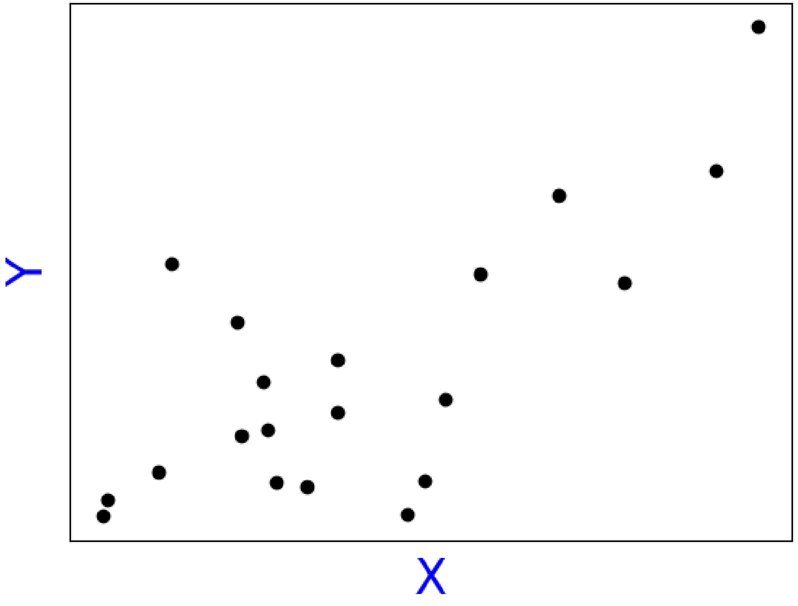
  $\beta$: model parameters, $e$: error term

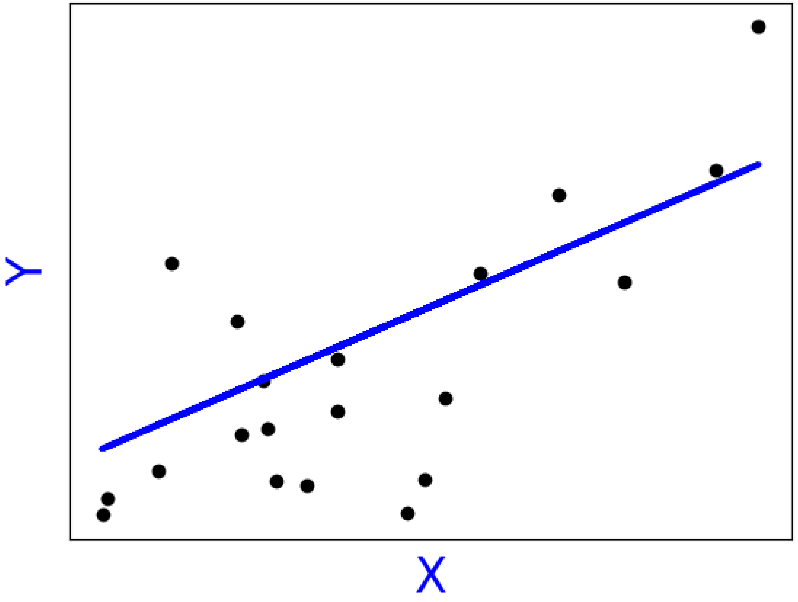
- Linear regression assumes a linear relationship between inputs and output

- Nonlinear regression does not assume a linear relationship
  - E.g., exponential, logarithmic, Gaussian

# Linear regression

- $Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_n X_{in} + e_i$
- Least squares to find the best fit:
- $r_i = Y_i - f(X_i, \beta)$
- Objective: minimize $S = \sum_{i=0}^{n} r_i^2$
- Polynomial regression is also linear regression!



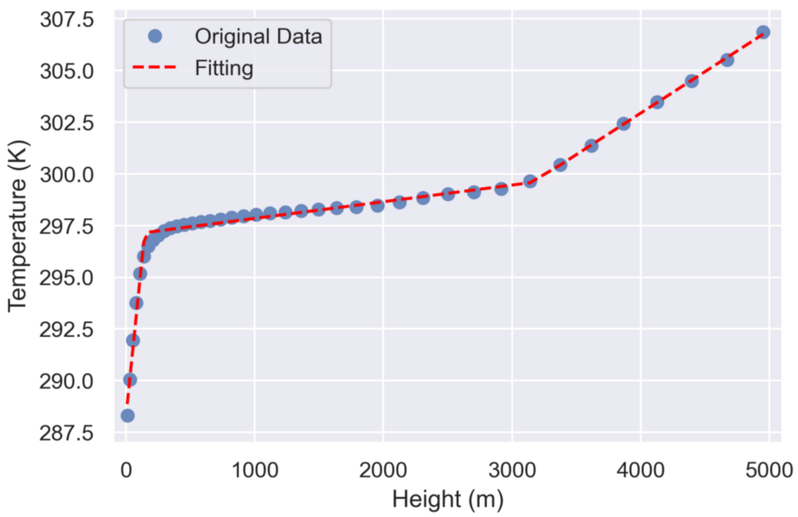





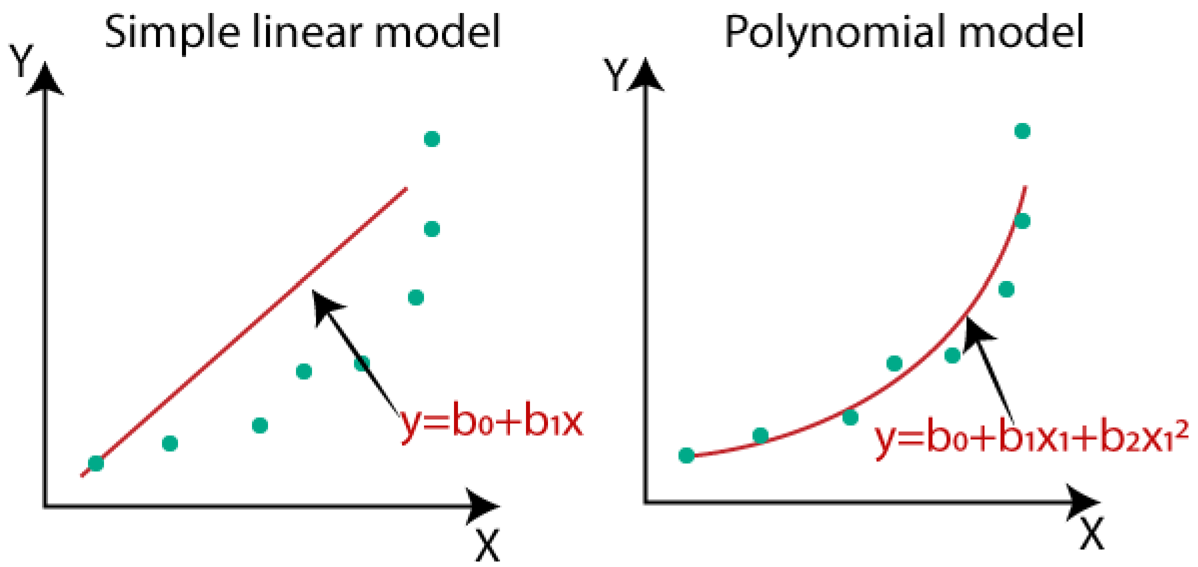

EPFL csem

# Linear regression – additional examples

- Piecewise linear regression




- Cubinc polynomial regression





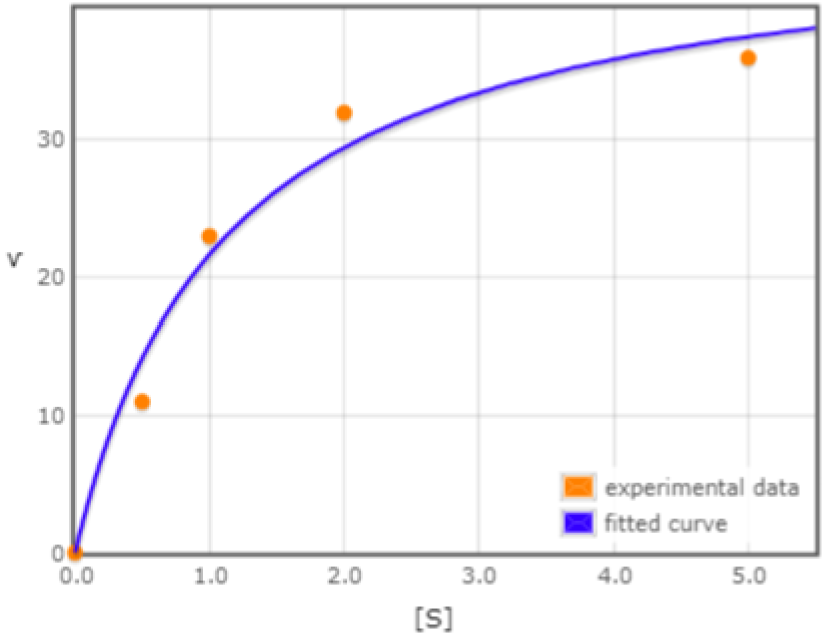

EPFL csem

# Nonlinear regression

- Includes logarithmic, exponential, Gaussian, …

- Example: Equation for enzyme kinetics:

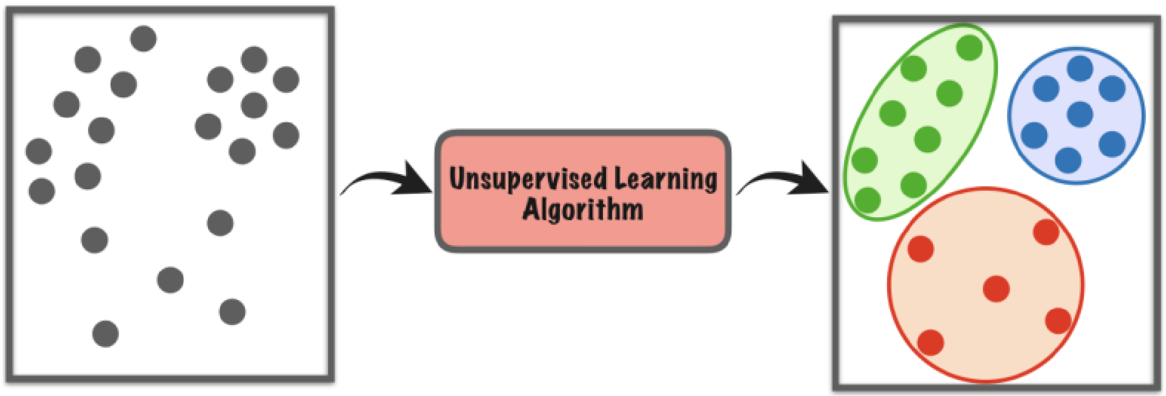
- $f(x, \beta) = \frac{\beta_1 * x}{\beta_2 + x}$



# Practical examples in digital health

- Speed estimation from cadence
  - $Speed = \beta_0 + \beta_1 cadence + \beta_2 slope + \beta_3 height + \beta_4 energy + \ \ldots$

# Unsupervised learning / Clustering



# Centroid based method – k means

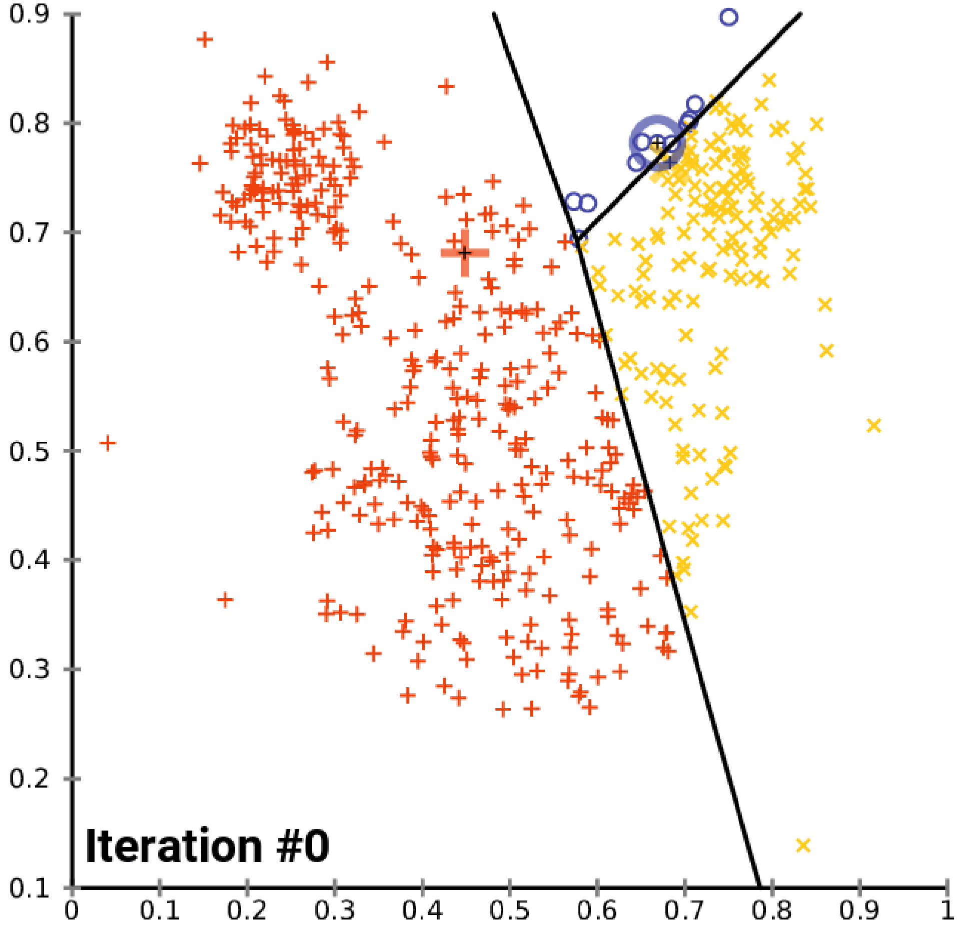
- Initialize a set of means $\{m_1, m_2, \ldots m_k\}$
- Assign each observation to a cluster with mean based on least square distance

$$S_i^{(t)} = \left\{x_p: \left\|x_p - m_i^{(t)}\right\|^2 \le \left\|x_p - m_j^{(t)}\right\|^2 \forall j, 1 \le j \le k\right\}$$

- Update means (centroids)

$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$

- Converge when no more updates possible



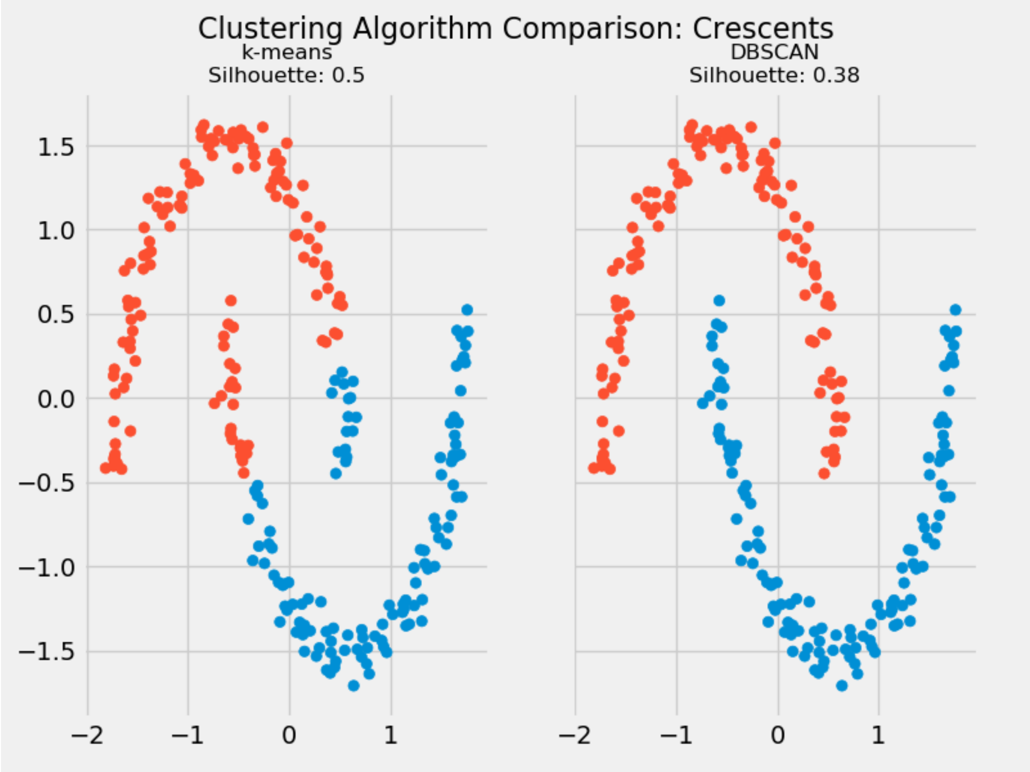
# Clustering – k means

- Advantages: easy method when labels are unknown

- Limitations: may not converge optimally (wrong clusters!)

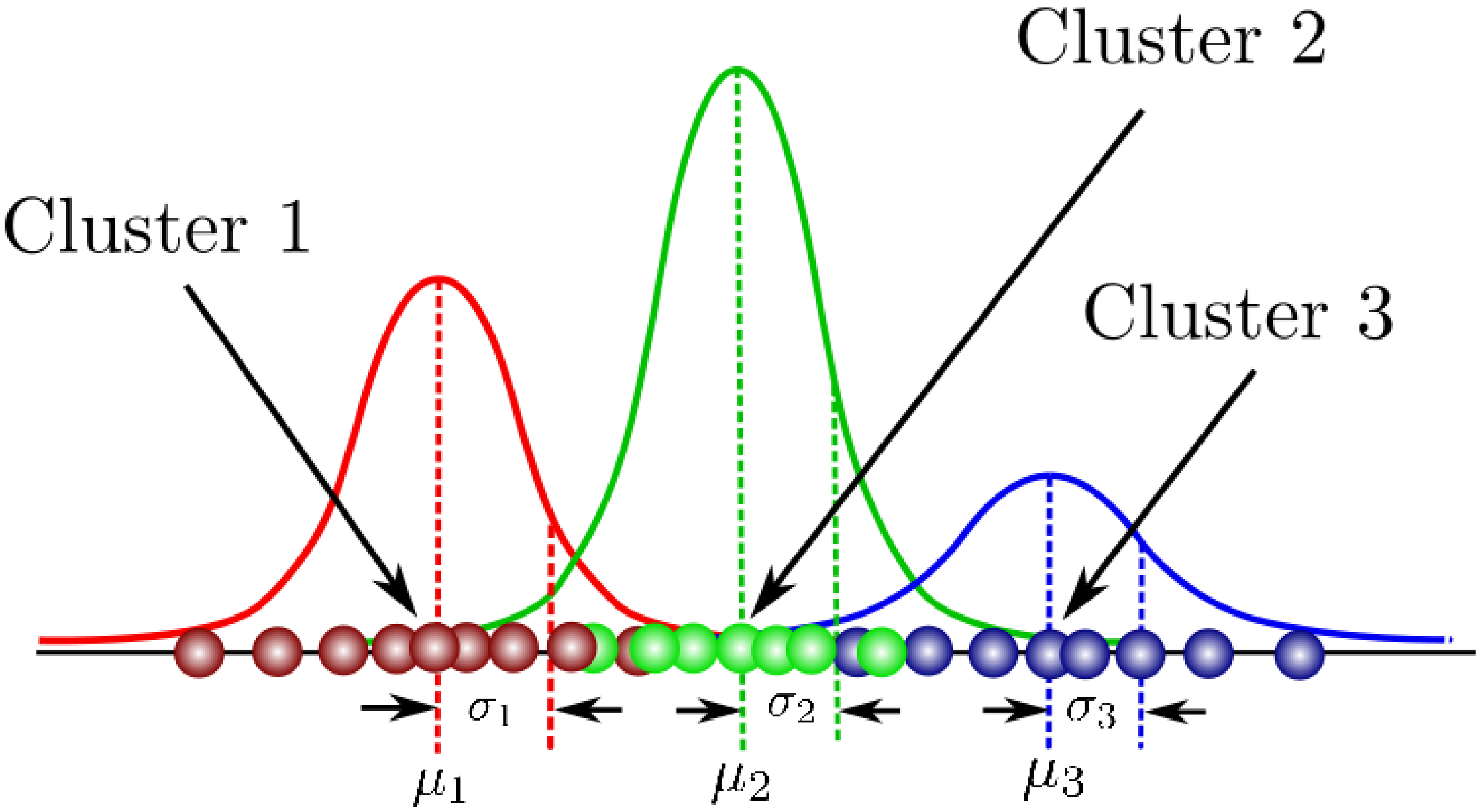
- Initialization of clusters is not always evident, needs a predefined number of clusters

- Insensitive to data shape (e.g. non-linearly separable clusters)

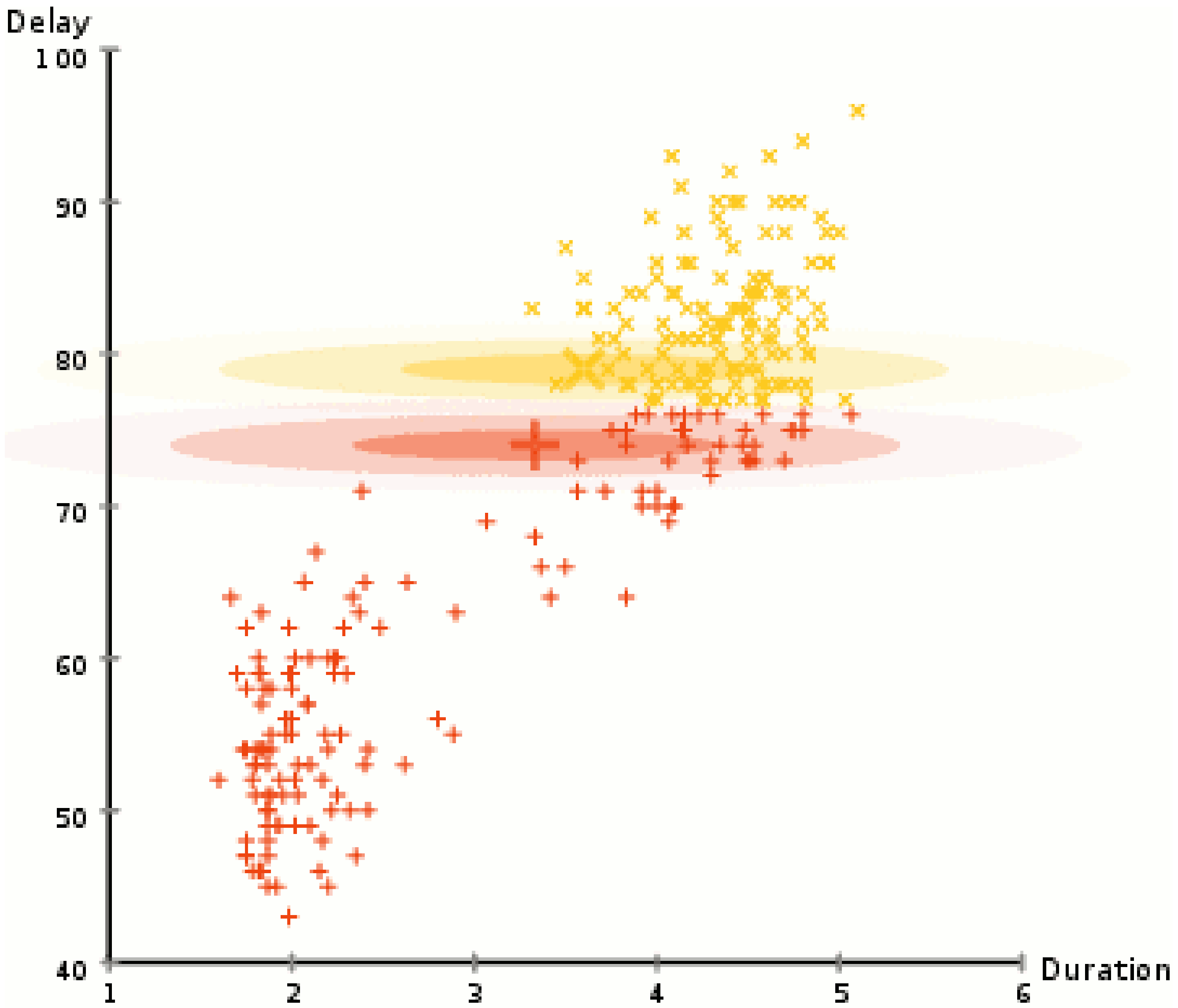


# Distribution based method – Gaussian Mixture Model

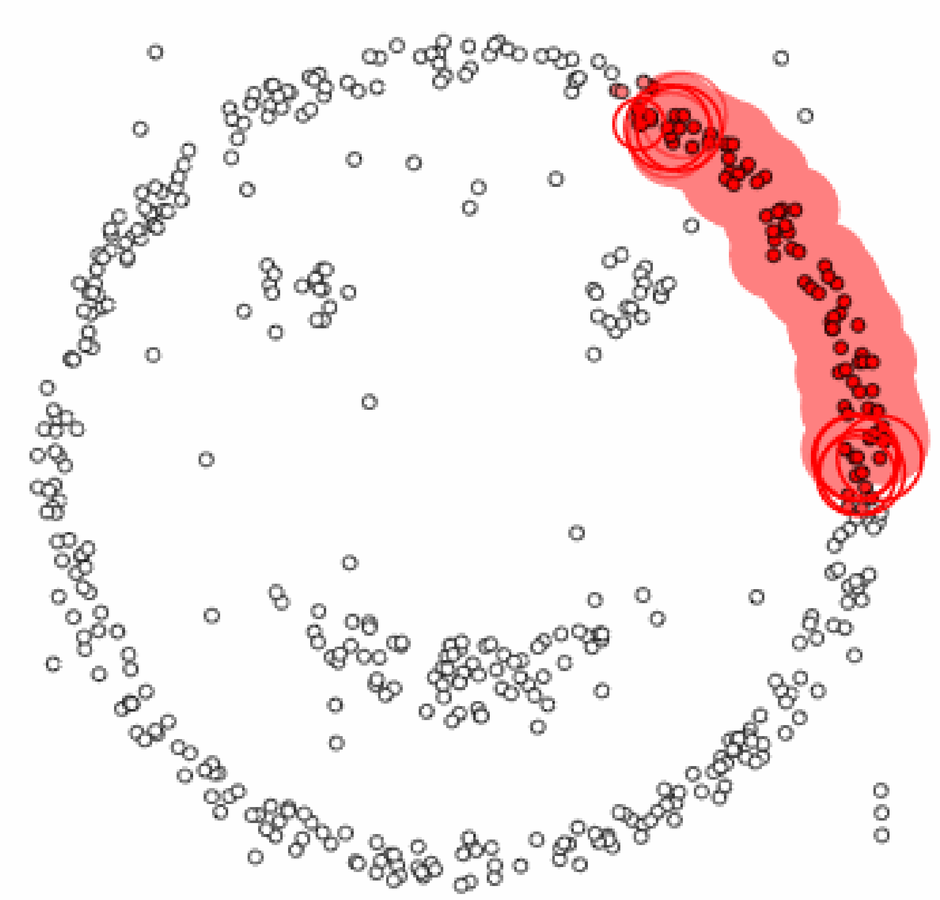
- Relies on a mixture of Gaussian densities
- Expectation – Maximization algorithm



# Distribution based method – Gaussian Mixture Model



# Density based method – DBSCAN



- Requires two parameters:
  - ε, the radius of a neighborhood
  - minPTs, minimum number of points to form a dense region
- Start with a single point, and add new points to the cluster until there are no more points within ε
- Then, a new point belongs to a new cluster
- Repeat until there are no more points

# Clustering – DBSCAN

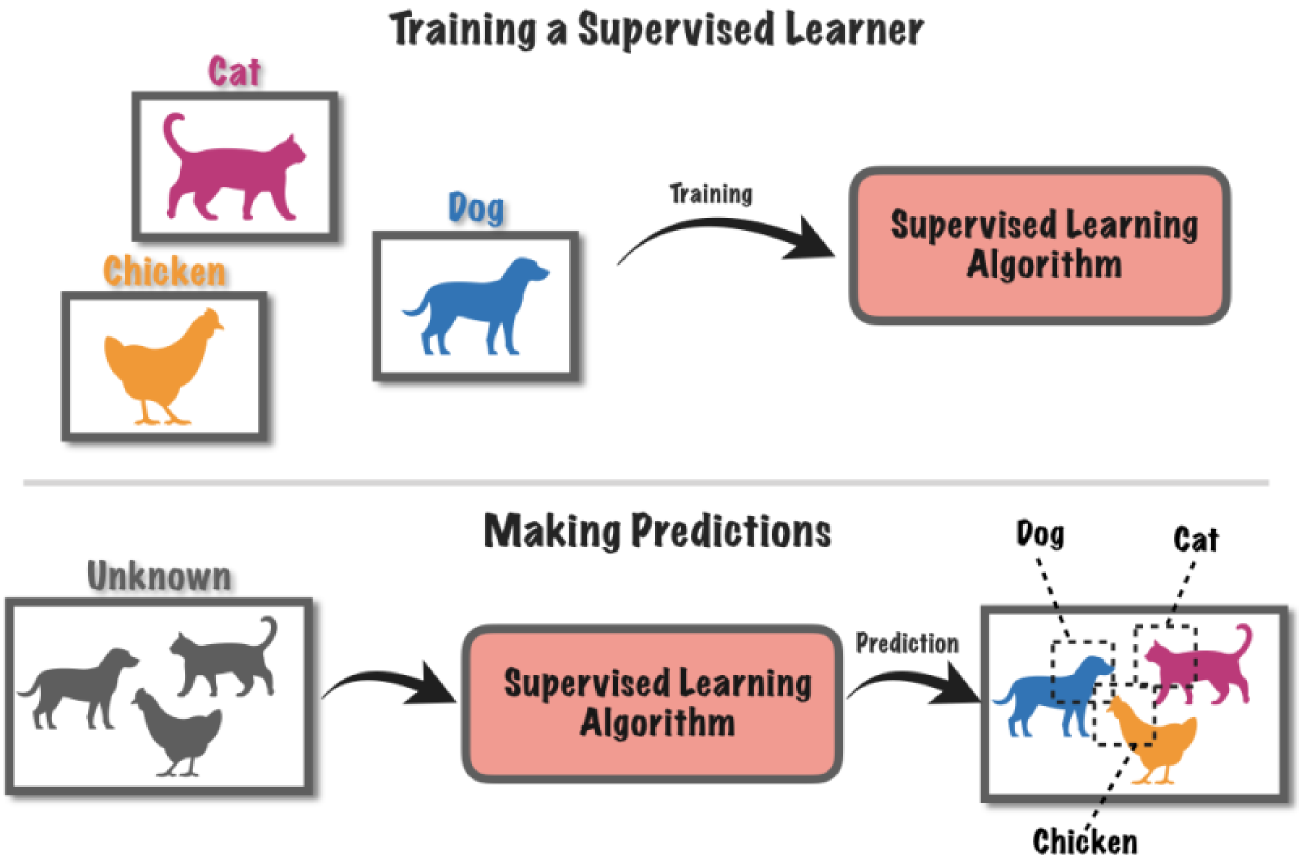
- **Advantages**
  - No prior #clusters knowledge
  - Sensitive to data shape
  - Robust to outliers
  - Only two parameters needed

- **Drawbacks**
  - Depends on order of point selection
  - Heavily dependent on distance metric
  - Choosing appropriate distance metric may be difficult

# Clustering – Conclusion

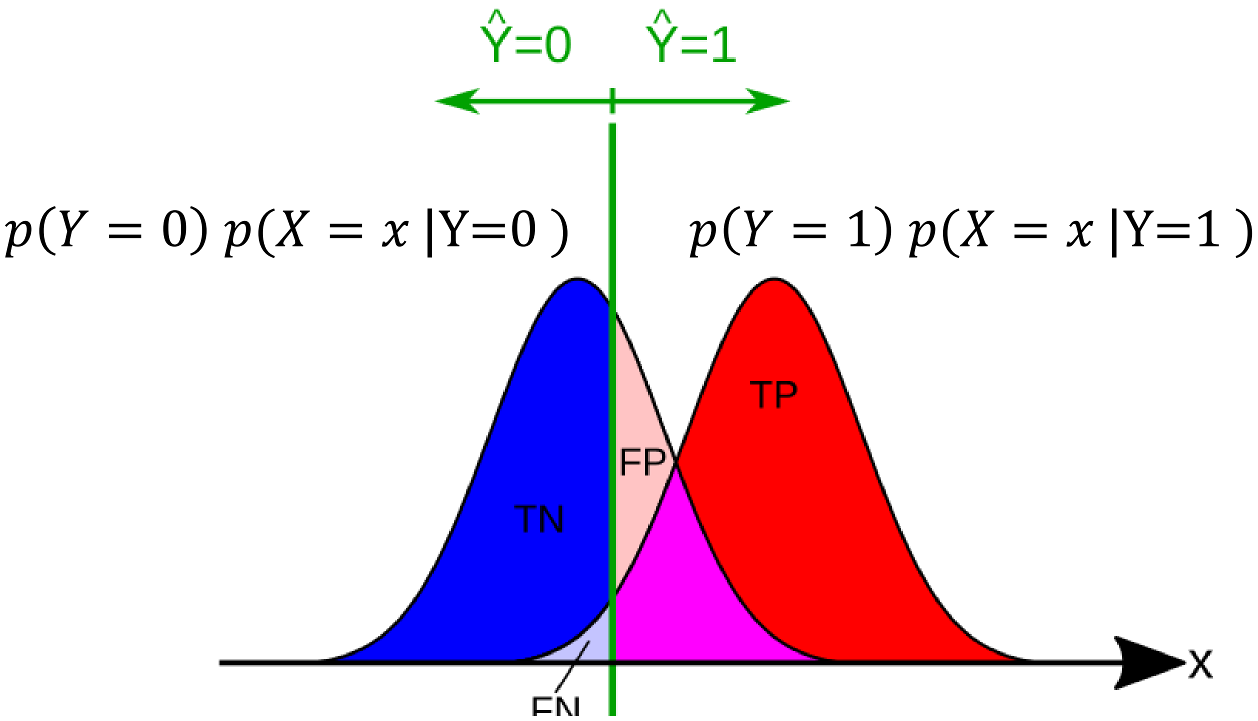
- Clustering is useful when attempting to make sense of unlabeled data
- Several methods exist with varying advantages/drawbacks
- Model selection may benefit from prior knowledge about the data distribution/shape/domain expertise

# Supervised learning



# Naïve Bayes (probabilistic classifier)

- Assign a new observation $\hat{y}$ to a class C:
  - $\hat{y} = \underset{k\in\{1,\ldots,K\}}{argmax}\, p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$
- Common assumption is that probability is Gaussian:
  - $p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$
  - Where v is an observation



# Logistic Regression

- Based on the logistic function, for data that has a “sigmoid” distribution

$$p(x) = \frac{1}{1+e^{-(x-\mu)/s}}$$

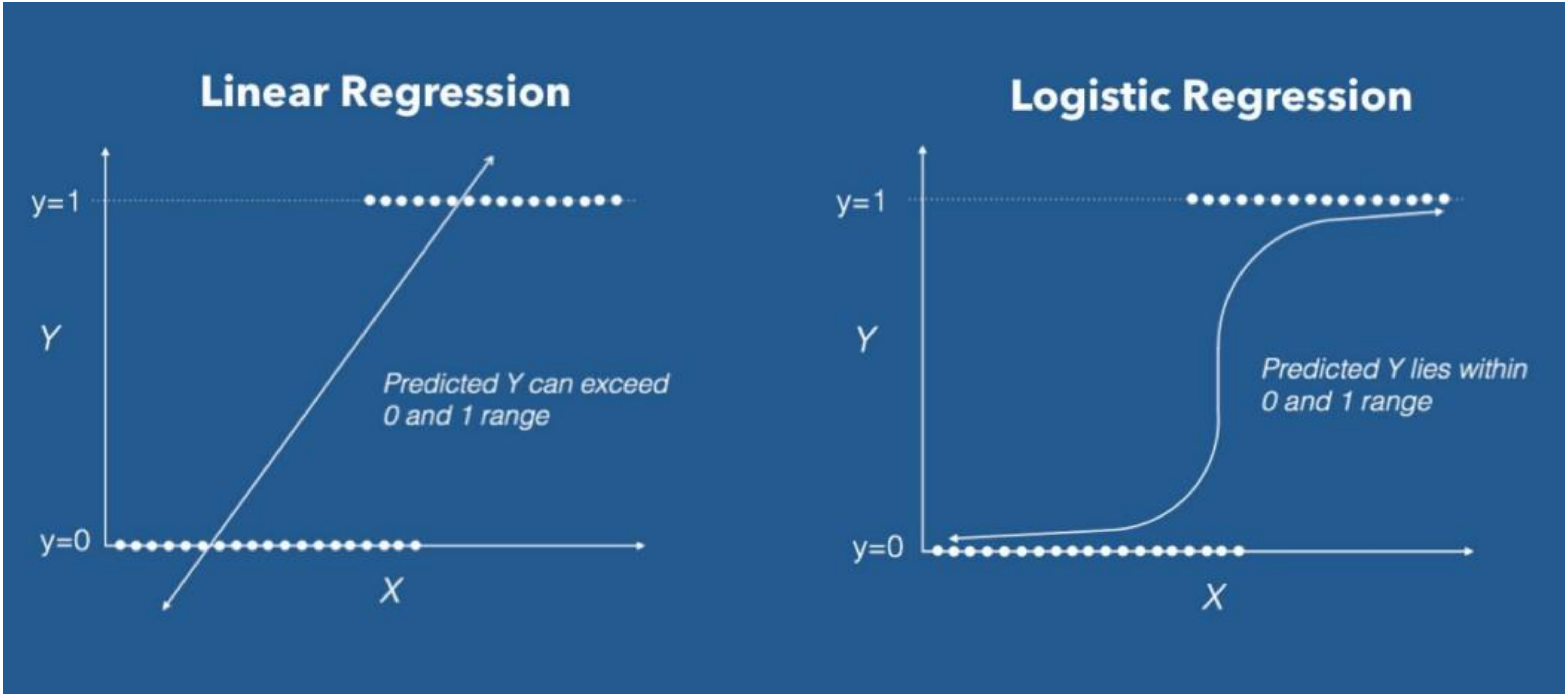
with μ a location parameter (p(μ) = 0.5) and s a scale parameter

- Goal is to minimize negative log-likelihood (alternatively, maximize the positive log-likelihood):

$$cost = \begin{Bmatrix} -\log(p(x)\ if\ y = 1 \\ -\log(1 - p(x)\ if\ y = 0 \end{Bmatrix}$$

- Gradient descent: minimize the derivative of the cost function

# Logistic Regression



# Decision trees

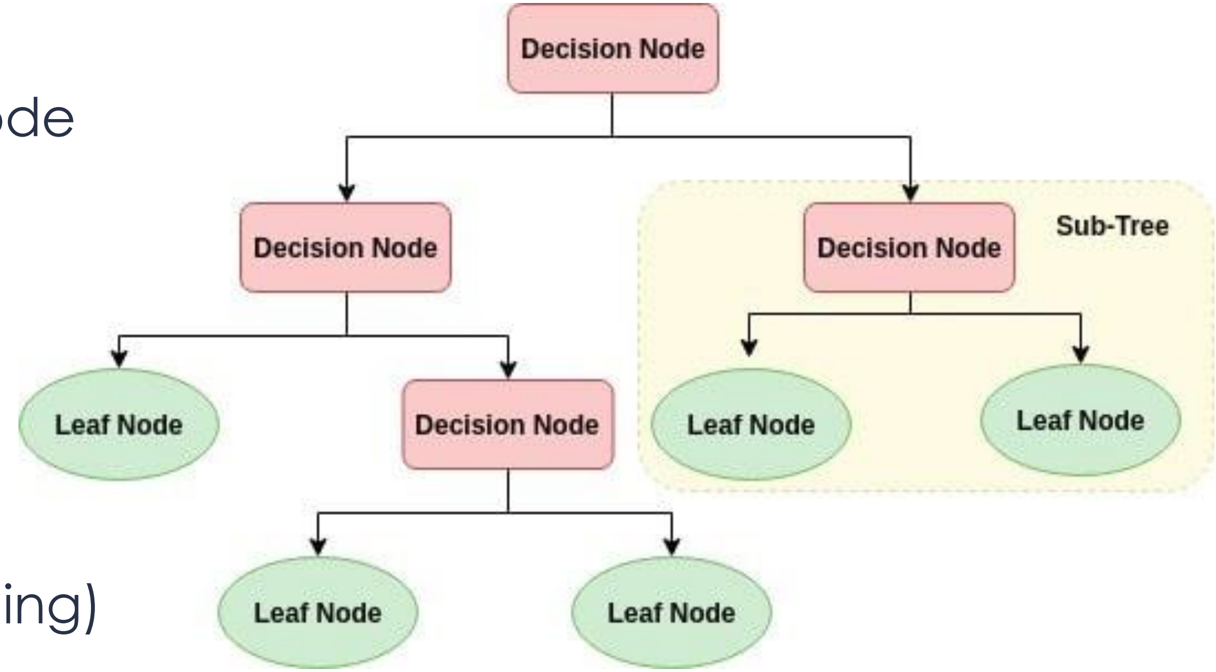
- Classification based on splitting data through decision nodes
- Attribute selection measure to optimize nodes:
- E.g., information gain / Gain ratio (used in C4.5 tree)

  Entropy: $H(T) = -\sum_{i=1}^{J} p_i log_2(p_i)$

  - Information gain between parent node and sum of children nodes:

    $$IG(T,a) = H(T) - H(T|a)$$

- Tree depth and maximum nodes may be selected and optimized
- Tree pruning (e.g., reduced error pruning)

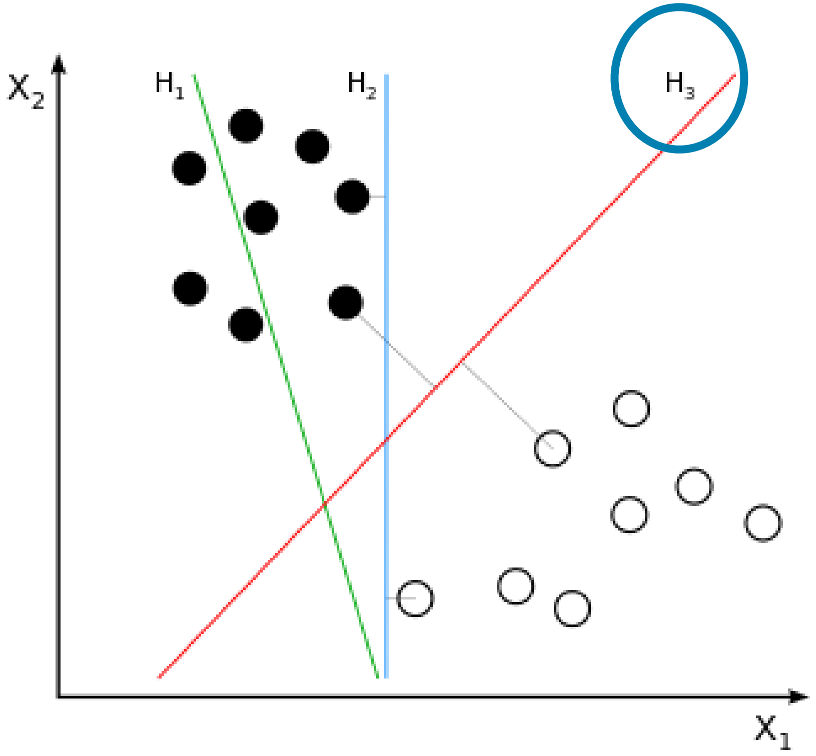


# Support Vector Machines

- Motivation: classify data with highest margin between classes
- Linear classifier:

  $w^T x \; - b = 1$

  $w^T x \; - b = -1$, $w$ a vector normal to the hyperplane




- Non-linear classifier (kernel)
  - E.g., Gaussian radial basis function
  - $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$, $x, x'$ are feature vectors
- Hard and soft margins

# Regularization

# Regularization

- Add additional constrain to the cost function of the models
- Regularized linear regression
  - **Ridge Regression**

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

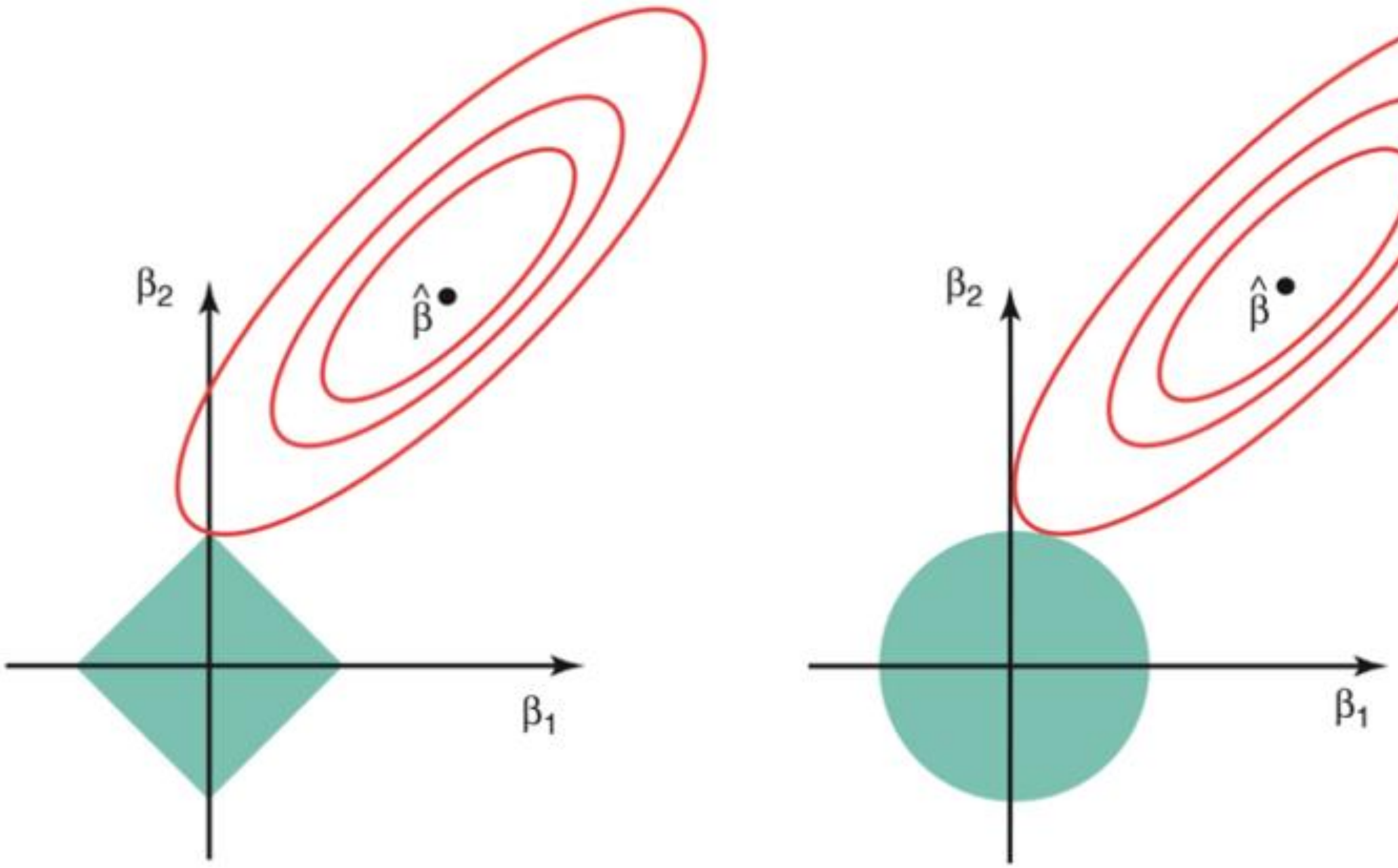


  - **LASSO** (least absolute shrinkage and selection operator)

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

  - **Elastic net: Ridge + LASSO**

# Regularization



# Regularization

- Advantages:
  - Avoids overfitting
  - Manage multicollinearity among features (avoid singularity)
  - Dimensionality reduction (i.e., Simplicity and Computational Efficiency)D



- Disadvantages
  - Deviation from the original goal (learning error reduction)

# Feature selection

# Feature selection

- What are features? **Special characteristics of a class**
- E.g.:
  - Time domain: peak detection, zero-crossings, amplitude, peak-to-peak
  - Frequency domain: spectrogram transformation (e.g. Fourier)
  - Statistical: mean, median, variance, standard deviation, …
  - Expert based / heuristic: domain or application specific!

# Feature selection

- Goal: select subset of features to be used in classification

- Why not use all available features?
  - Simplify models, reduce time/computational complexity
  - Avoid "curse of dimensionality": more data is needed to train with more features!
  - Avoid overfitting

# Feature selection - Algorithm

- Filter
  - Rank features based on information metrics
- Wrapper
  - Use the performance of the model with the selected features to select the best subset
- Embedded
  - Feature selection is embedded in model (learns features and model simultaneously)

# Feature selection methods

## Filter

- Univariate analysis
- Information gain
- Pearson correlation
- T-test

## Embedded

- LASSO

## Wrapper

- Sequential (forward/backward)
- Genetic algorithm
- Recursive elimination

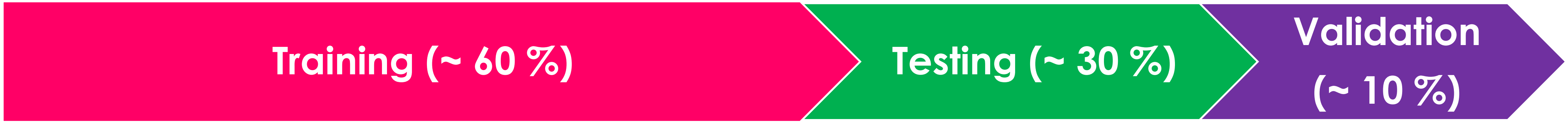
# Feature selection

- **Filter**
  - Fast, classifier independent, reduces risk of overfitting, BUT
  - Does not look at feature or model dependencies
- **Wrapper / Embedded**
  - Models interactions/dependencies, may perform better than filter, BUT
  - Slower algorithms, overfitting prone, classifier-dependent

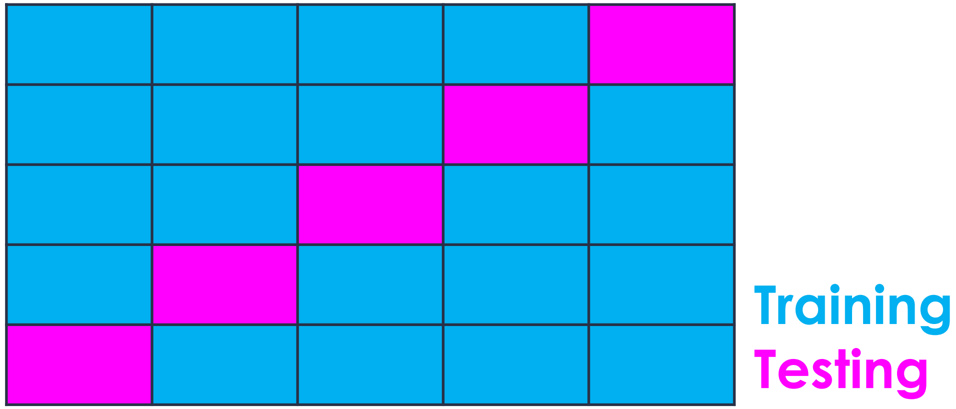
# Validation strategies

# Training / testing / validation

- Why train / test / validate?
- Model may perform well if trained with entire dataset, but may not **generalize** to classify unseen data → Avoid **overfitting**
- **Data splitting**




- **Cross validation (e.g. leave one out)**
  - Split the data into n folds
  - Train on n-1 folds and test on remaining fold
  - Repeat n times



# Performance evaluation

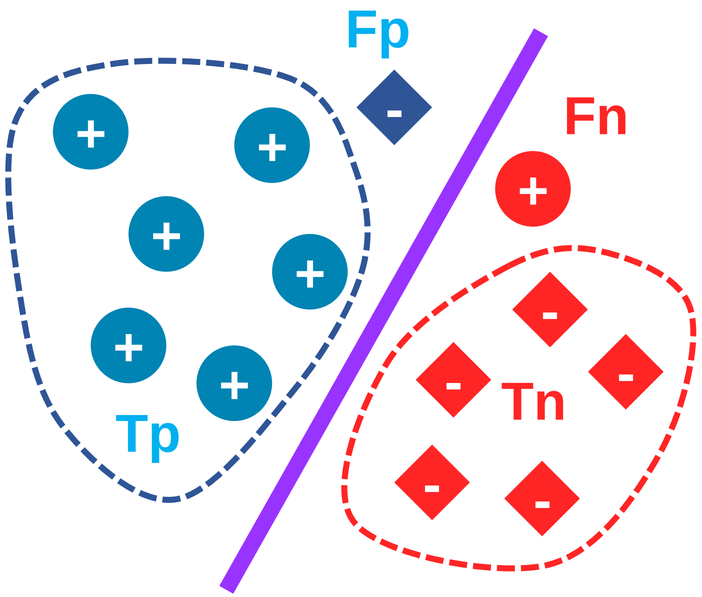
# Confusion matrix

- Table representation of classification outputs
- Used for performance metrics calculation
- Activity classification example:

| | | Predicted class | | |
|---|---|---|---|---|
| | | Rest | Walk | Run |
| Actual class | Rest | 95 | 5 | 0 |
| | Walk | 0 | 92 | 8 |
| | Run | 0 | 2 | 98 |

# Confusion matrix performance metrics

- **TP = True Positive**: Class correctly detected
- **TN = True Negative**: Class correctly rejected
- **FP = False Positive**: Class incorrectly detected
- **FN = False Negative**: Class incorrectly rejected




$$Sensitivity = \frac{Tp}{Tp + Fn}$$

$$Specificity = \frac{Tn}{Tn + Fp}$$

$$Accuracy = \frac{Tp + Tn}{Tp + Fn + Tn + Fp}$$

$$Precision = \frac{Tp}{Tp + Fp}$$

# Confusion matrix performance metrics

- Activity classification example:

| | | Predicted class | | |
|---|---|---|---|---|
| | | Rest | Walk | Run |
| Actual class | Rest | 95 | 5 | 0 |
| | Walk | 0 | 92 | 8 |
| | Run | 0 | 2 | 98 |

- Walk sensitivity = 92 / (92 + 8) = 92%
- Walk precision = 92 / (92 + 5 + 2) = 93%
- Walk specificity = (95 + 98) / ( 95 + 98 + 5 + 2) = 97%

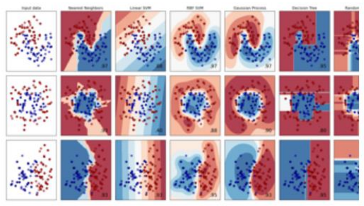
# Coding tools for regression/machine learning

- scikit-learn (python toolbox)

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.
**Algorithms:** SVM, nearest neighbors, random forest, and more...

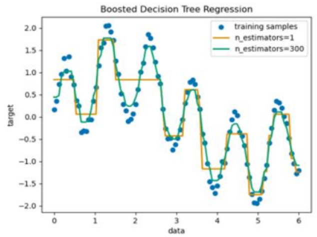


Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.
**Algorithms:** SVR, nearest neighbors, random forest, and more...

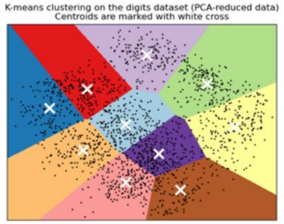



Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes
**Algorithms:** k-Means, spectral clustering, mean-shift, and more...

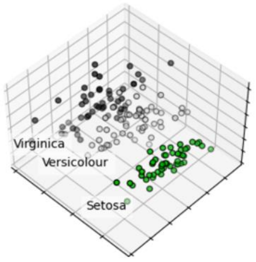



Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency
**Algorithms:** PCA, feature selection, non-negative matrix factorization, and more...

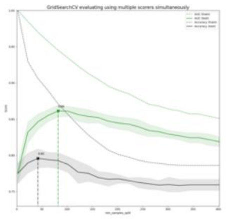



Examples

## Model selection

Comparing, validating and choosing parameters and models.

**Applications:** Improved accuracy via parameter tuning
**Algorithms:** grid search, cross validation, metrics, and more...

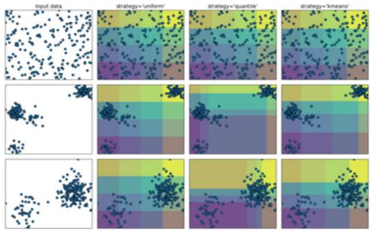


Examples

## Preprocessing

Feature extraction and normalization.

**Applications:** Transforming input data such as text for use with machine learning algorithms.
**Algorithms:** preprocessing, feature extraction, and more...



Examples

- Ramin Soltani – asi@csem.ch
- CSEM Signal Processing Group