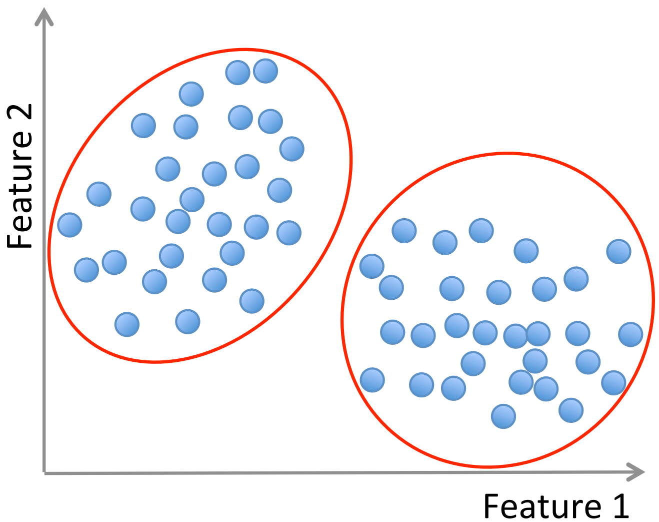
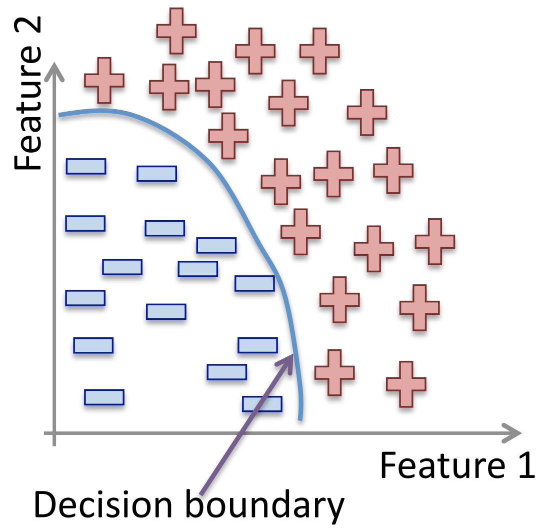
# Machine Learning

## Basic Concepts



# Terminology

Machine Learning, Data Science, Data Mining, Data Analysis, Statistical Learning, Knowledge Discovery in Databases, Pattern Discovery.

# Data everywhere!

1. **Google:** processes 24 peta bytes of data per day.
2. **Facebook:** 10 million photos uploaded every hour.
3. **Youtube:** 1 hour of video uploaded every second.
4. **Twitter:** 400 million tweets per day.
5. **Astronomy:** Satellite data is in hundreds of PB.
6. ...
7. **"By 2020 the digital universe will reach 44 zettabytes..."**

The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things, April 2014.

That's 44 trillion gigabytes!

# Data types

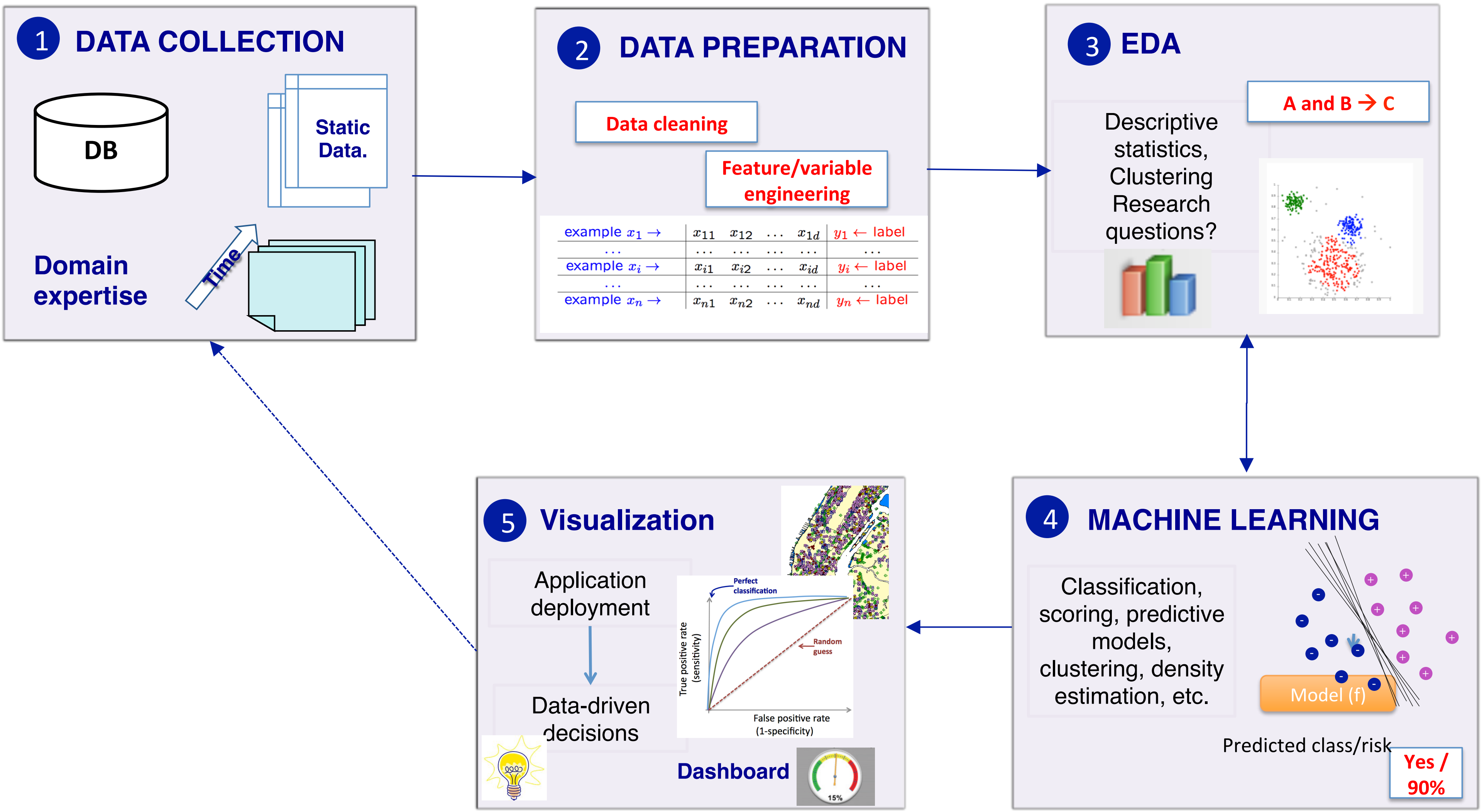
Data comes in different sizes and also flavors (types):

- ☒ **Texts**
- ☒ **Numbers**
- ☒ **Clickstreams**
- ☒ **Graphs**
- ☒ **Tables**
- ☒ **Images**
- ☒ **Transactions**
- ☒ **Videos**
- ☒ **Some or all of the above!**

# Smile, we are 'DATAFIED'!

- Wherever we go, we are "datafied".
- Smartphones are tracking our locations.
- We leave a data trail in our web browsing.
- Interaction in social networks.
- Privacy is an important issue in Data Science.

# The Data Science process



# Applications of ML

- We all use it on a daily basis. Examples:

# Machine Learning

- Spam filtering
- Credit card fraud detection
- Digit recognition on checks, zip codes
- Detecting faces in images
- MRI image analysis
- Recommendation system
- Search engines
- Handwriting recognition
- Scene classification
- etc...

# Interdisciplinary field

# ML versus Statistics

**Statistics:**

- Hypothesis testing
- Experimental design
- Anova
- Linear regression
- Logistic regression
- GLM
- PCA

**Machine Learning:**

- Decision trees
- Rule induction
- Neural Networks
- SVMs
- Clustering method
- Association rules
- Feature selection
- Visualization
- Graphical models
- Genetic algorithm

http://statweb.stanford.edu/~jhf/ftp/dm-stat.pdf

# Machine Learning definition

"How do we create computer programs that improve with experience?"

Tom Mitchell

http://videolectures.net/mlas06_mitchell_itm/

# Machine Learning definition

"How do we create computer programs that improve with experience?"

Tom Mitchell

http://videolectures.net/mlas06_mitchell_itm/

"A computer program is said to **learn** from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$. "

Tom Mitchell. Machine Learning 1997.

# Supervised vs. Unsupervised

**Given:** Training data: $(x_1, y_1), \ldots, (x_n, y_n)$ / $x_i \in \mathbb{R}^d$ and $y_i$ is the label.

| | | | | | |
|---|---|---|---|---|---|
| example $x_1 \rightarrow$ | $x_{11}$ | $x_{12}$ | $\ldots$ | $x_{1d}$ | $y_1 \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_i \rightarrow$ | $x_{i1}$ | $x_{i2}$ | $\ldots$ | $x_{id}$ | $y_i \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_n \rightarrow$ | $x_{n1}$ | $x_{n2}$ | $\ldots$ | $x_{nd}$ | $y_n \leftarrow$ label |

# Supervised vs. Unsupervised

**Given:** Training data: $(x_1, y_1), \ldots, (x_n, y_n)$ / $x_i \in \mathbb{R}^d$ and $y_i$ is the label.

| | | | | | |
|---|---|---|---|---|---|
| example $x_1 \rightarrow$ | $x_{11}$ | $x_{12}$ | $\ldots$ | $x_{1d}$ | $y_1 \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_i \rightarrow$ | $x_{i1}$ | $x_{i2}$ | $\ldots$ | $x_{id}$ | $y_i \leftarrow$ label |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| example $x_n \rightarrow$ | $x_{n1}$ | $x_{n2}$ | $\ldots$ | $x_{nd}$ | $y_n \leftarrow$ label |

| fruit | length | width | weight | label |
|---|---|---|---|---|
| fruit 1 | 165 | 38 | 172 | Banana |
| fruit 2 | 218 | 39 | 230 | Banana |
| fruit 3 | 76 | 80 | 145 | Orange |
| fruit 4 | 145 | 35 | 150 | Banana |
| fruit 5 | 90 | 88 | 160 | Orange |
| ... | | | | |
| fruit n | ... | ... | ... | ... |

# Supervised vs. Unsupervised

| fruit | length | width | weight | label |
|---|---|---|---|---|
| fruit 1 | 165 | 38 | 172 | Banana |
| fruit 2 | 218 | 39 | 230 | Banana |
| fruit 3 | 76 | 80 | 145 | Orange |
| fruit 4 | 145 | 35 | 150 | Banana |
| fruit 5 | 90 | 88 | 160 | Orange |
| ... | | | | |
| fruit n | ... | ... | ... | ... |

**Unsupervised learning:**

Learning a model from **unlabeled** data.

**Supervised learning:**

Learning a model from **labeled** data.

# Unsupervised Learning

**Training data**: "examples" $x$.

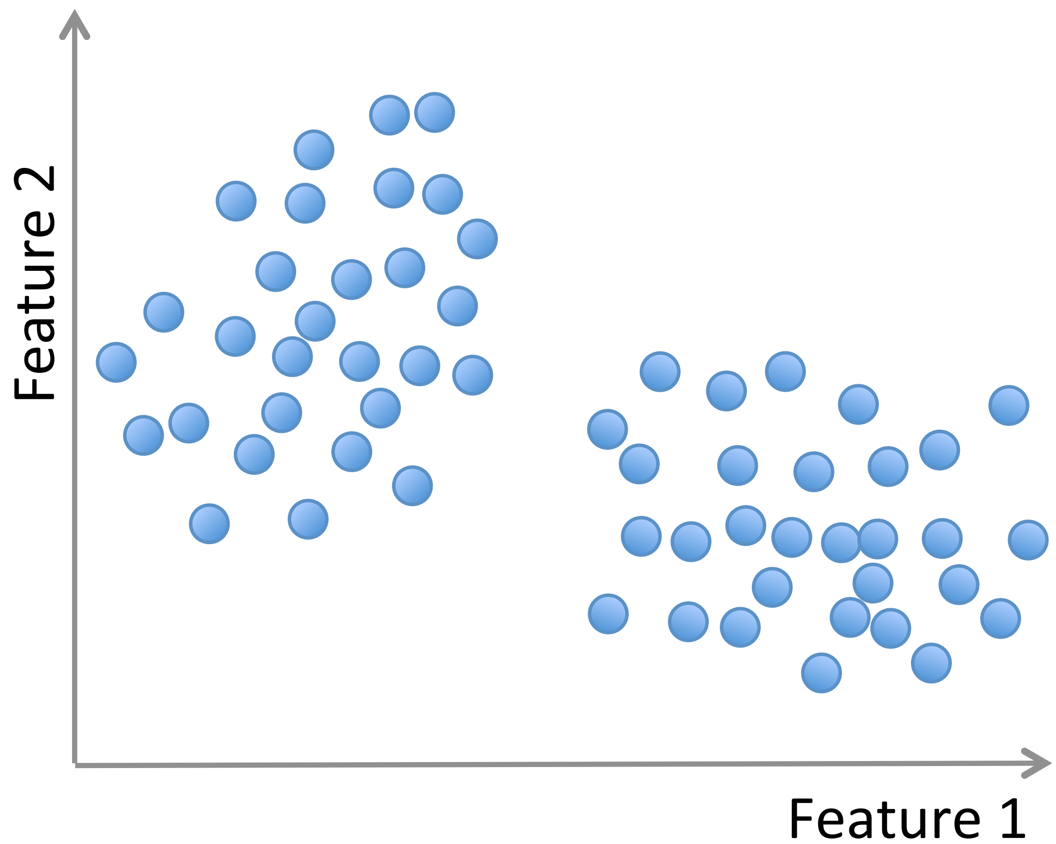
$$x_1, \ldots, x_n, \; x_i \in X \subset \mathbb{R}^n$$

- **Clustering/segmentation:**

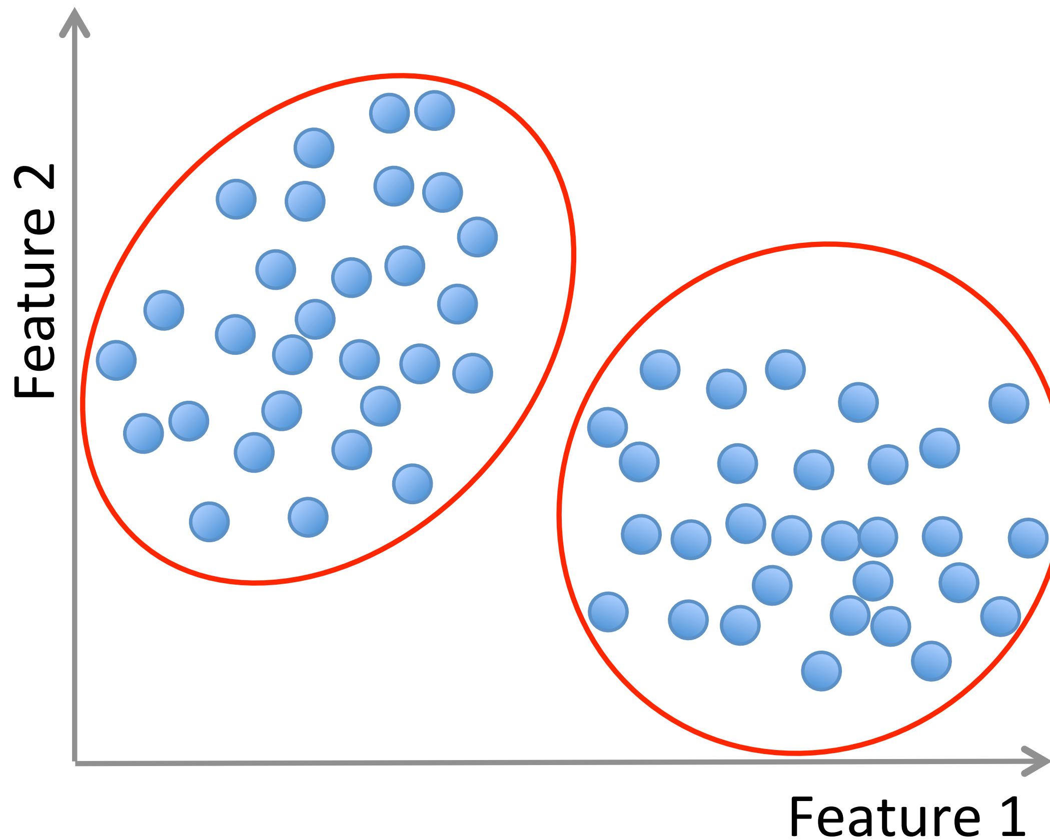
  $f : \mathbb{R}^d \longrightarrow \{C_1, \ldots C_k\}$ (set of clusters).

  Example: Find clusters in the population, fruits, species.

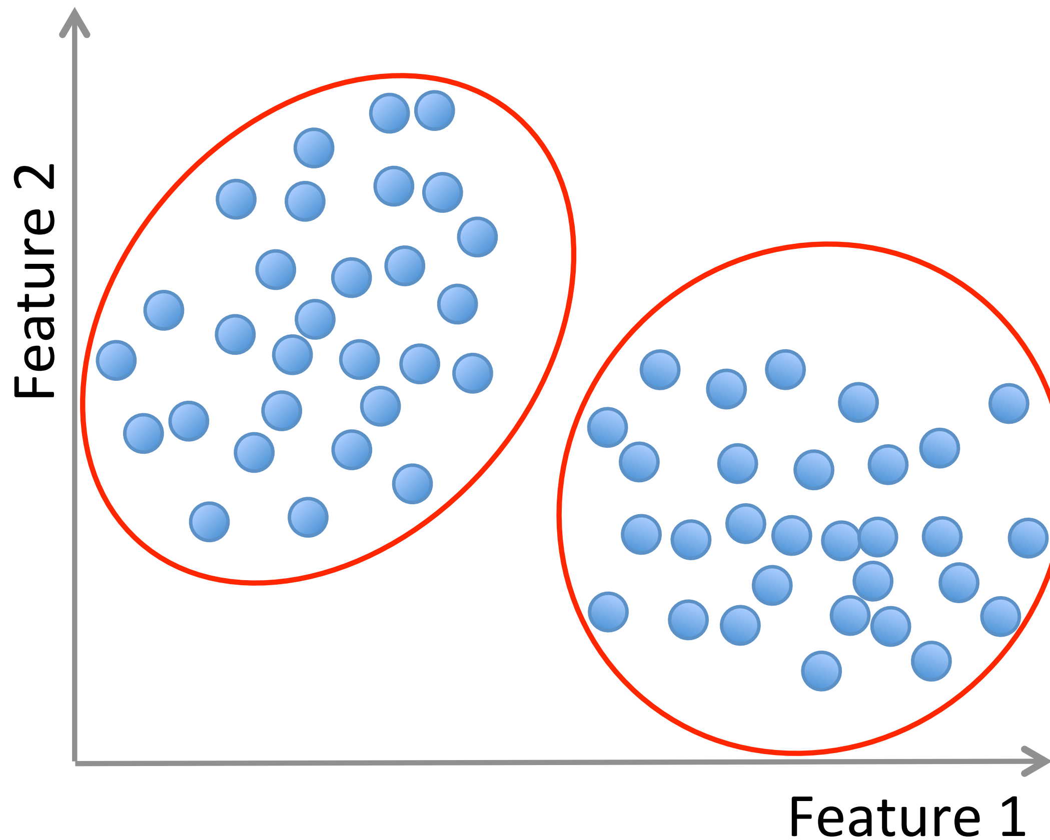
# Unsupervised learning



# Unsupervised learning



# Unsupervised learning




**Methods**: K-means, gaussian mixtures, hierarchical clustering, spectral clustering, etc.

# Supervised learning

**Training data**: "examples" $x$ with "labels" $y$.

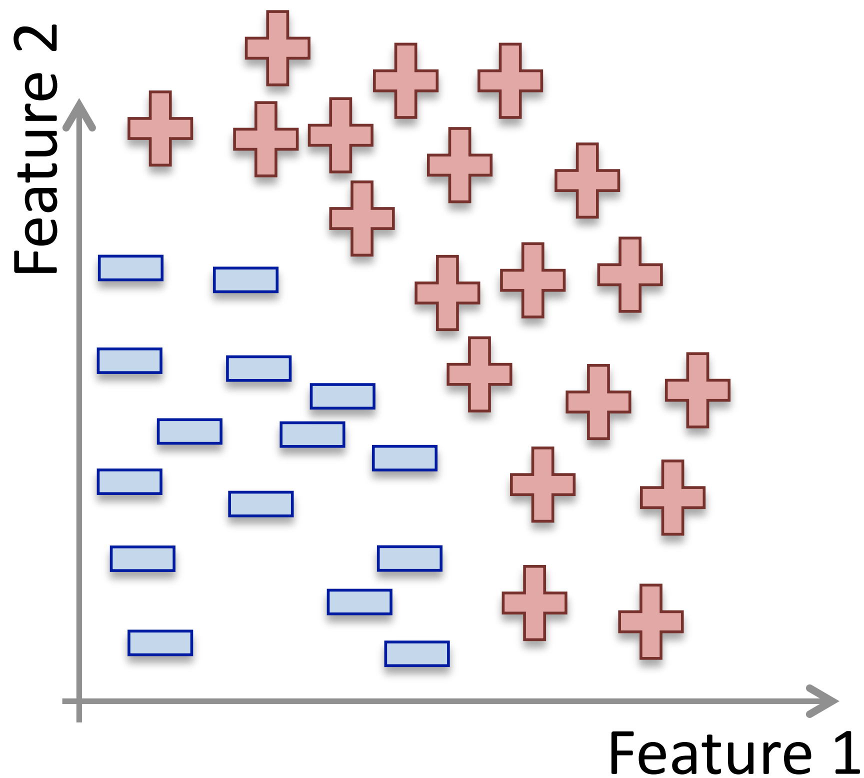
$$(x_1, y_1), \ldots, (x_n, y_n) \;/\; x_i \in \mathbb{R}^d$$

- **Classification:** $y$ is discrete. To simplify, $y \in \{-1, +1\}$

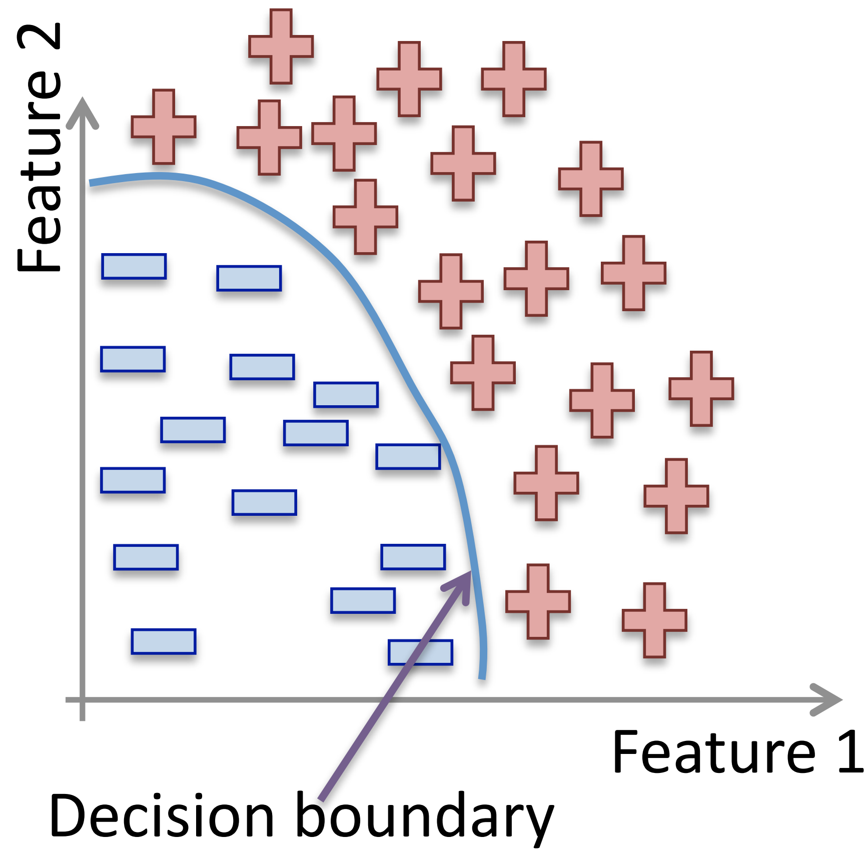
  $f : \mathbb{R}^d \longrightarrow \{-1, +1\}$ $f$ is called a **binary classifier**.

  Example: Approve credit yes/no, spam/ham, banana/orange.

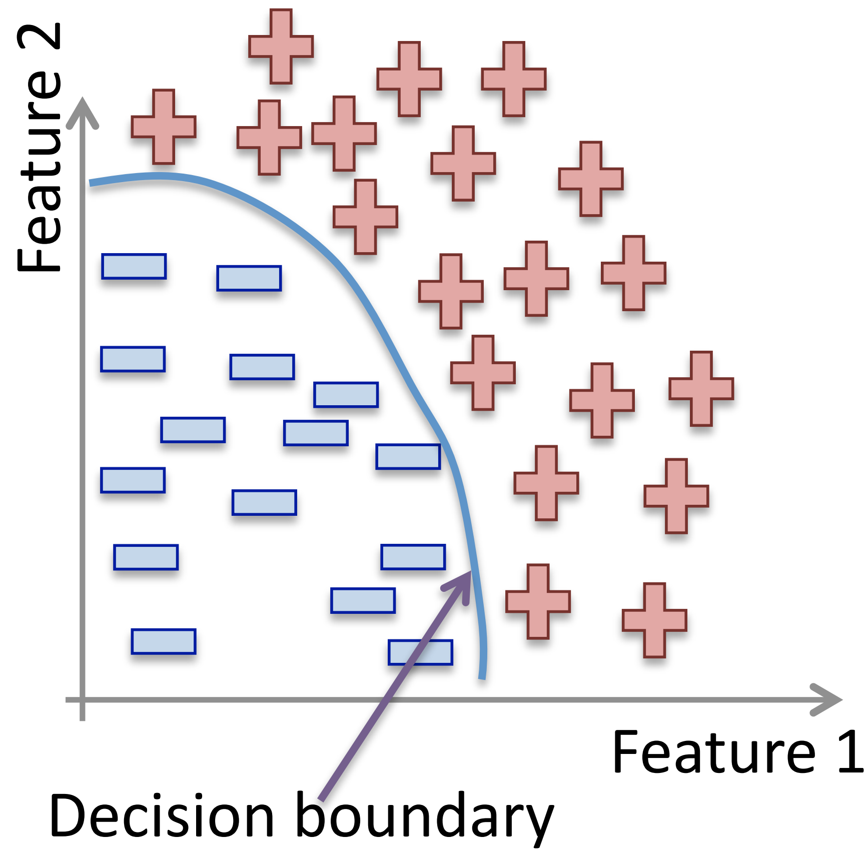
# Supervised learning



# Supervised learning



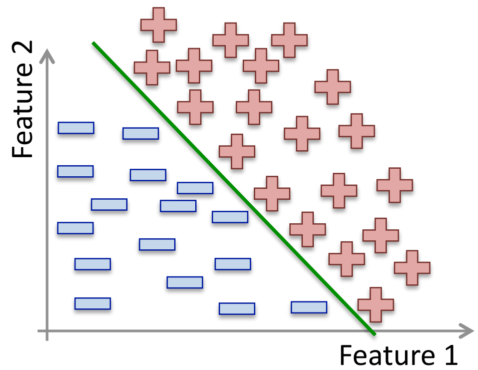
# Supervised learning

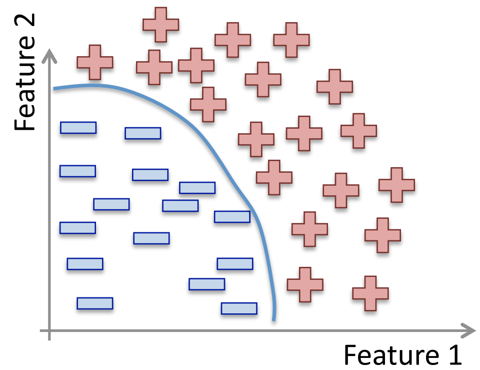


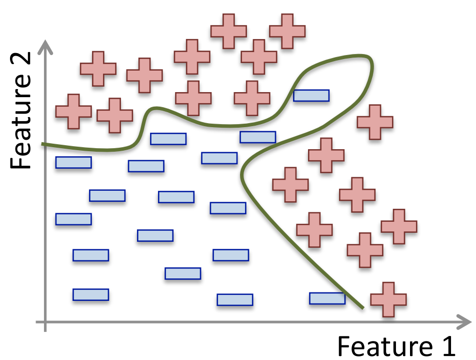

**Methods:** Support Vector Machines, neural networks, decision trees, K-nearest neighbors, naive Bayes, etc.

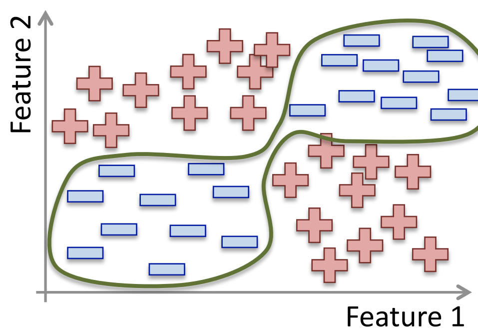
# Supervised learning

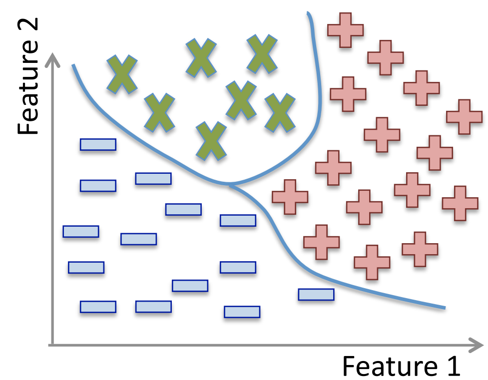
## Classification:



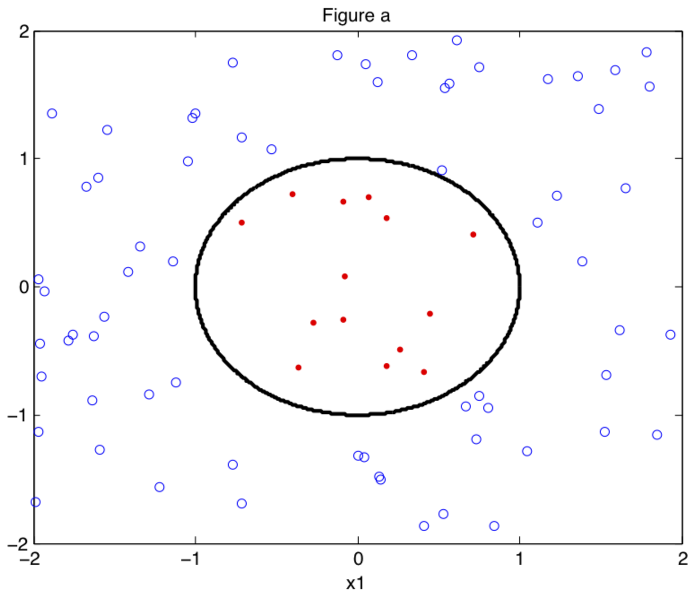



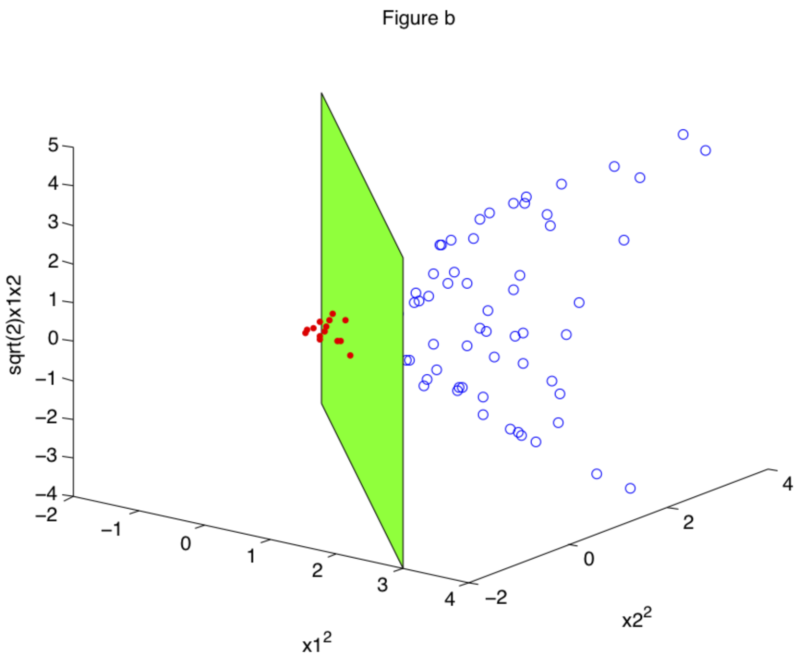



# Supervised learning

## Non linear classification



# Supervised learning

**Training data**: "examples" $x$ with "labels" $y$.

$$(x_1, y_1), \ldots, (x_n, y_n) \;/\; x_i \in \mathbb{R}^d$$

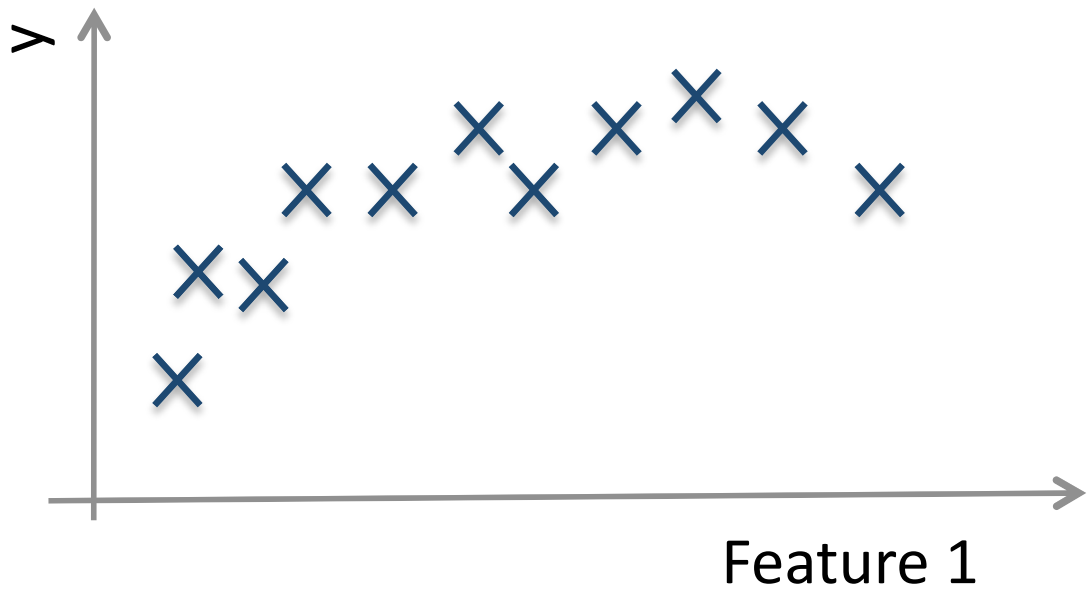
- **Regression:** $y$ is a real value, $y \in \mathbb{R}$

  $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ $\qquad$ $f$ is called a **regressor**.

  Example: amount of credit, weight of fruit.

# Supervised learning

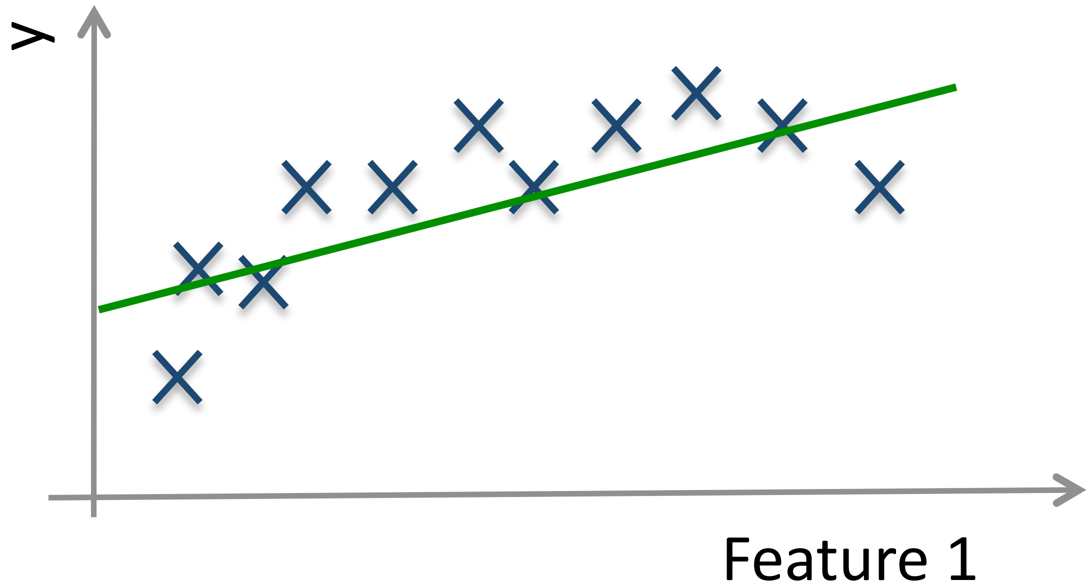
**Regression:**




Example: Income in function of age, weight of the fruit in function of its length.

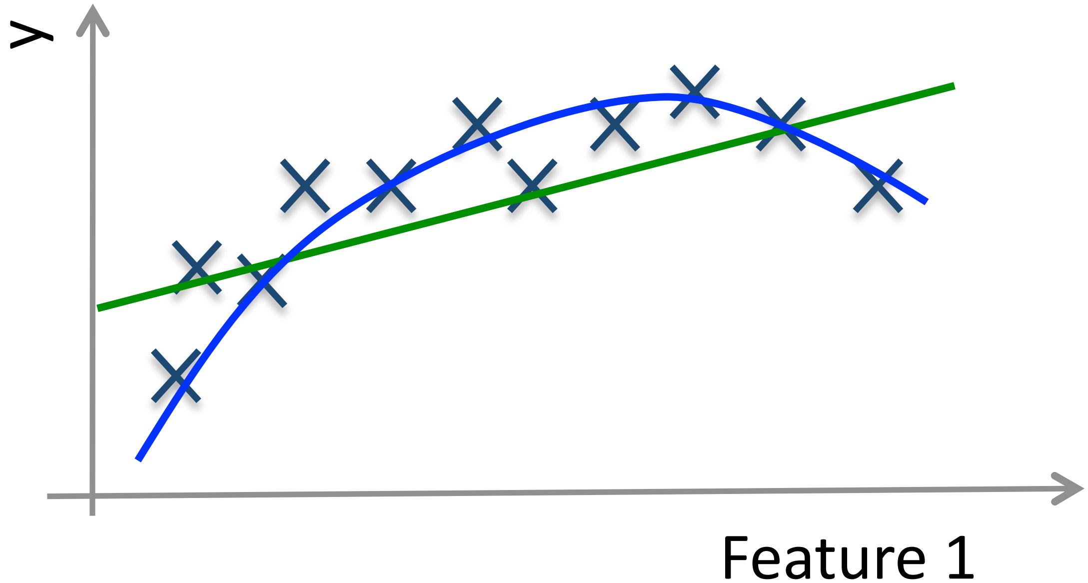
# Supervised learning

**Regression:**



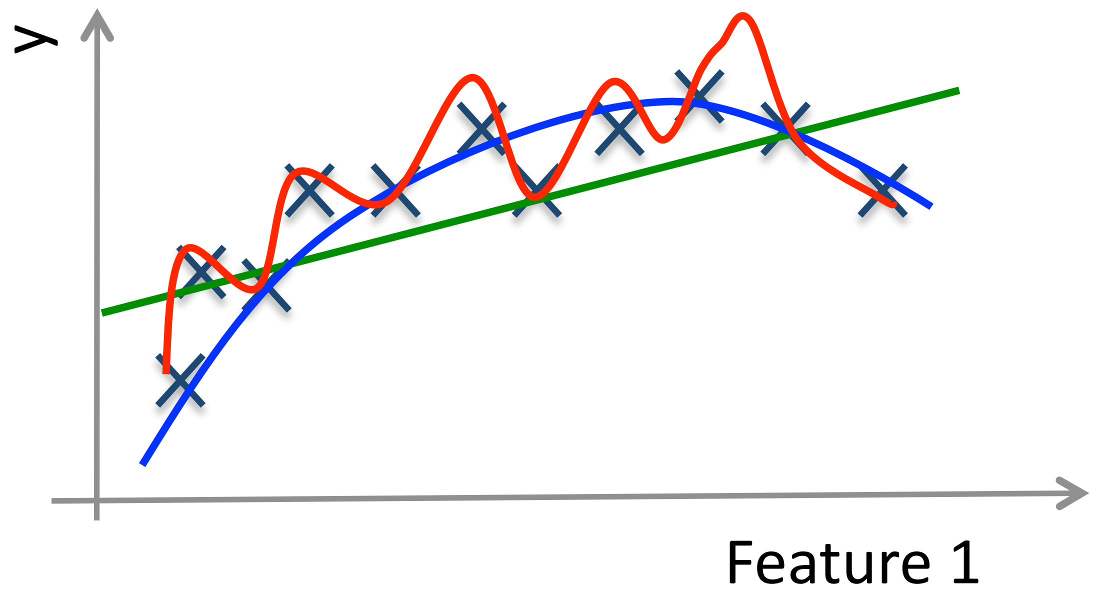
# Supervised learning

**Regression:**

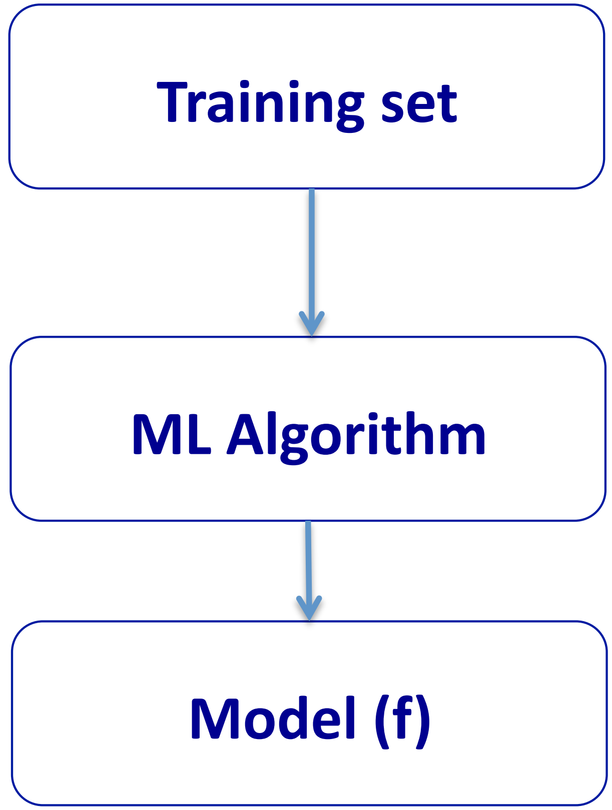


# Supervised learning

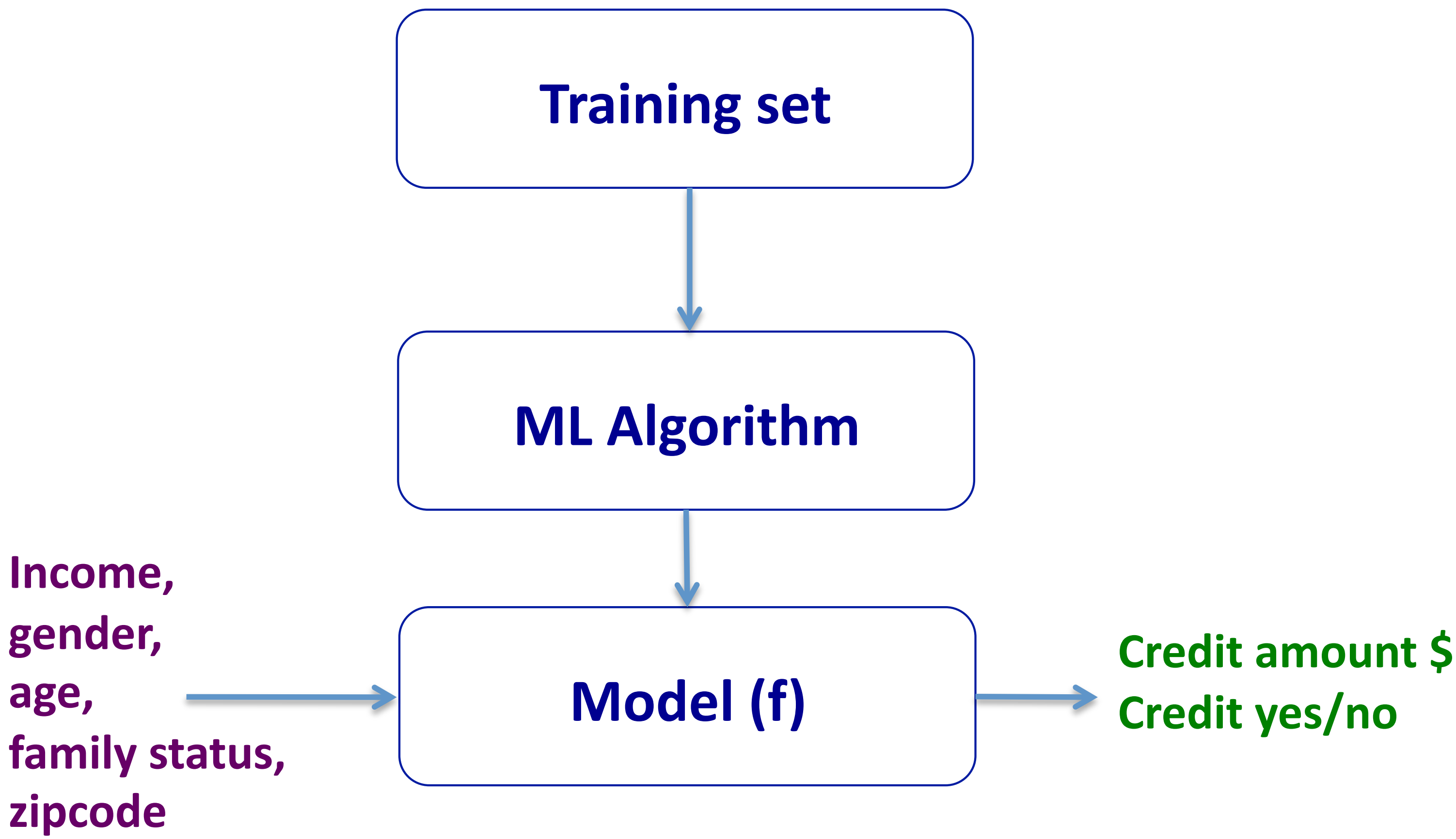
**Regression:**



# Training and Testing



# Training and Testing



# K-nearest neighbors

- Not every ML method builds a model!
- Our first ML method: KNN.
- Main idea: Uses the **similarity** between examples.
- Assumption: Two similar examples should have same labels.
- Assumes all examples (instances) are points in the $d$ dimensional space $\mathbb{R}^d$.

# K-nearest neighbors

- KNN uses the standard **Euclidian distance** to define nearest neighbors.
  Given two examples $x_i$ and $x_j$:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

# K-nearest neighbors

**Training algorithm:**

Add each training example $(x, y)$ to the dataset $\mathcal{D}$.

$x \in \mathbb{R}^d$, $y \in \{+1, -1\}$.

# K-nearest neighbors

**Training algorithm:**

Add each training example $(x, y)$ to the dataset $\mathcal{D}$.

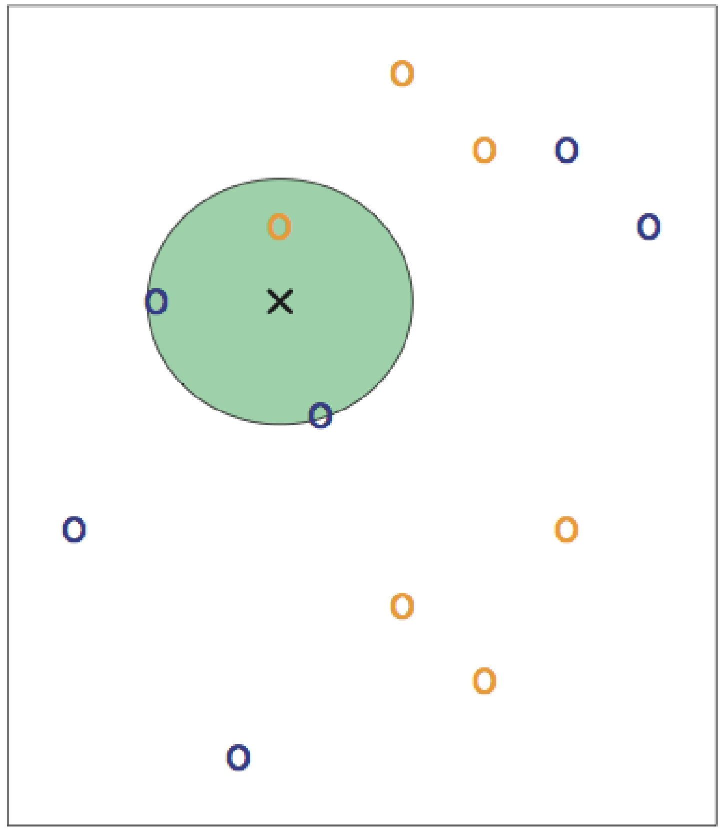
$x \in \mathbb{R}^d$, $y \in \{+1, -1\}$.

**Classification algorithm:**

Given an example $x_q$ to be classified. Suppose $N_k(x_q)$ is the set of the K-nearest neighbors of $x_q$.

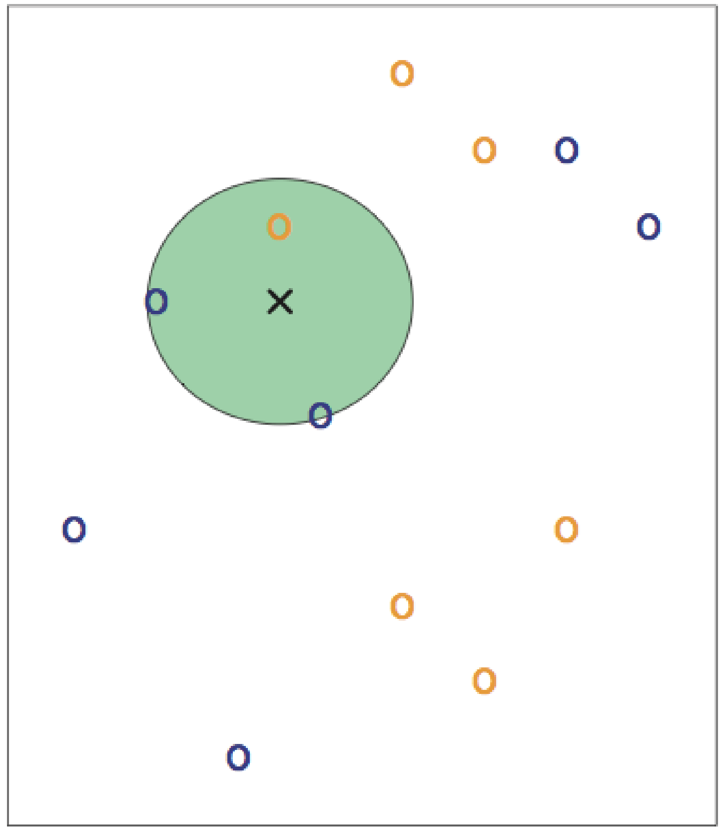
$$\widehat{y}_q = sign(\sum_{x_i \in N_k(x_q)} y_i)$$

# K-nearest neighbors



3-NN. Credit: Introduction to Statistical Learning.

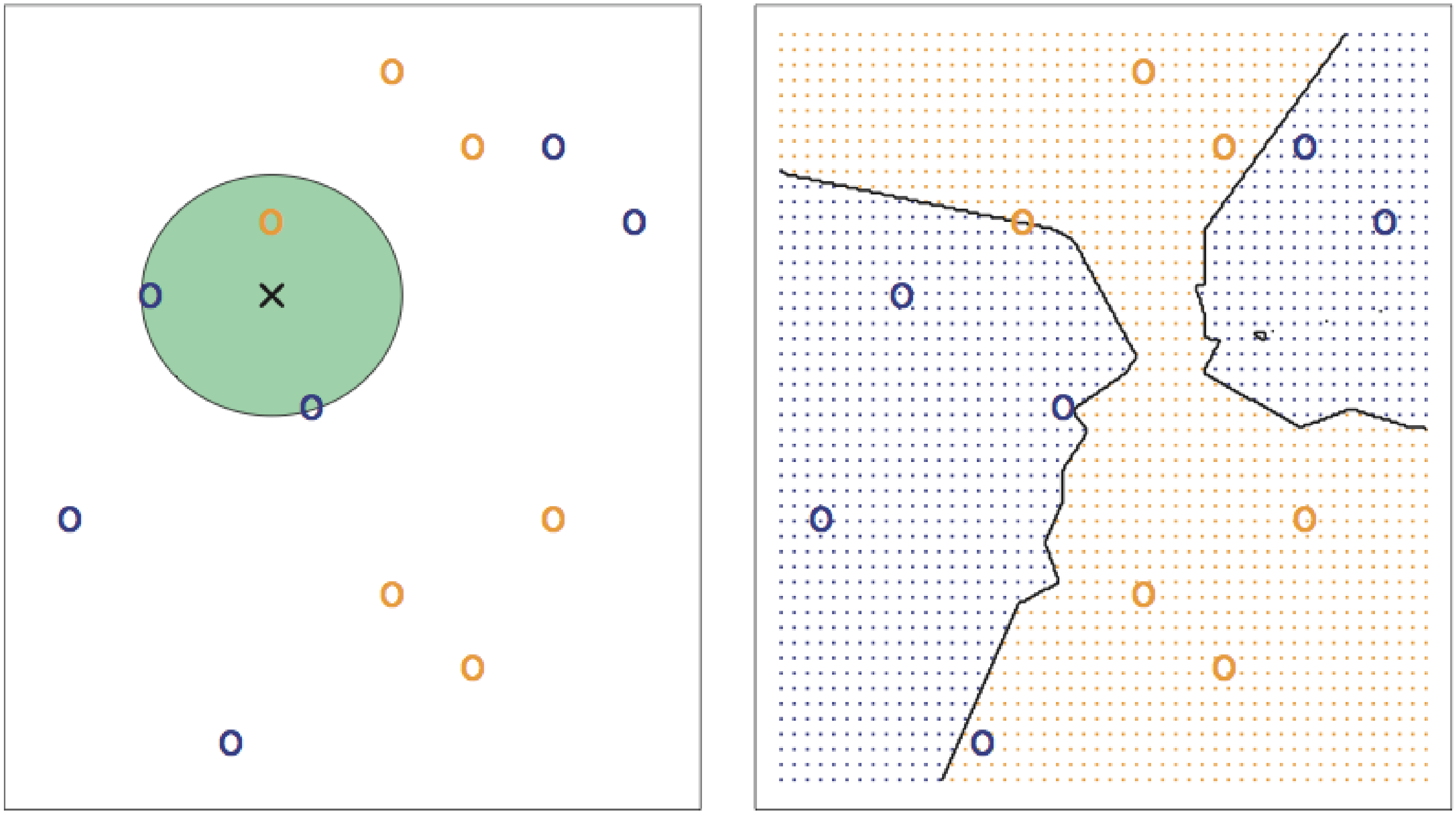
# K-nearest neighbors



3-NN. Credit: Introduction to Statistical Learning.

**Question: Draw an approximate decision boundary for $K = 3$?**

# K-nearest neighbors



Credit: Introduction to Statistical Learning.

# K-nearest neighbors

**Question: What are the pros and cons of K-NN?**

# K-nearest neighbors

**Question: What are the pros and cons of K-NN?**

**Pros:**

+ Simple to implement.
+ Works well in practice.
+ Does not require to build a model, make assumptions, tune parameters.
+ Can be extended easily with news examples.

# K-nearest neighbors

**Question: What are the pros and cons of K-NN?**

**Pros:**

+ Simple to implement.
+ Works well in practice.
+ Does not require to build a model, make assumptions, tune parameters.
+ Can be extended easily with news examples.

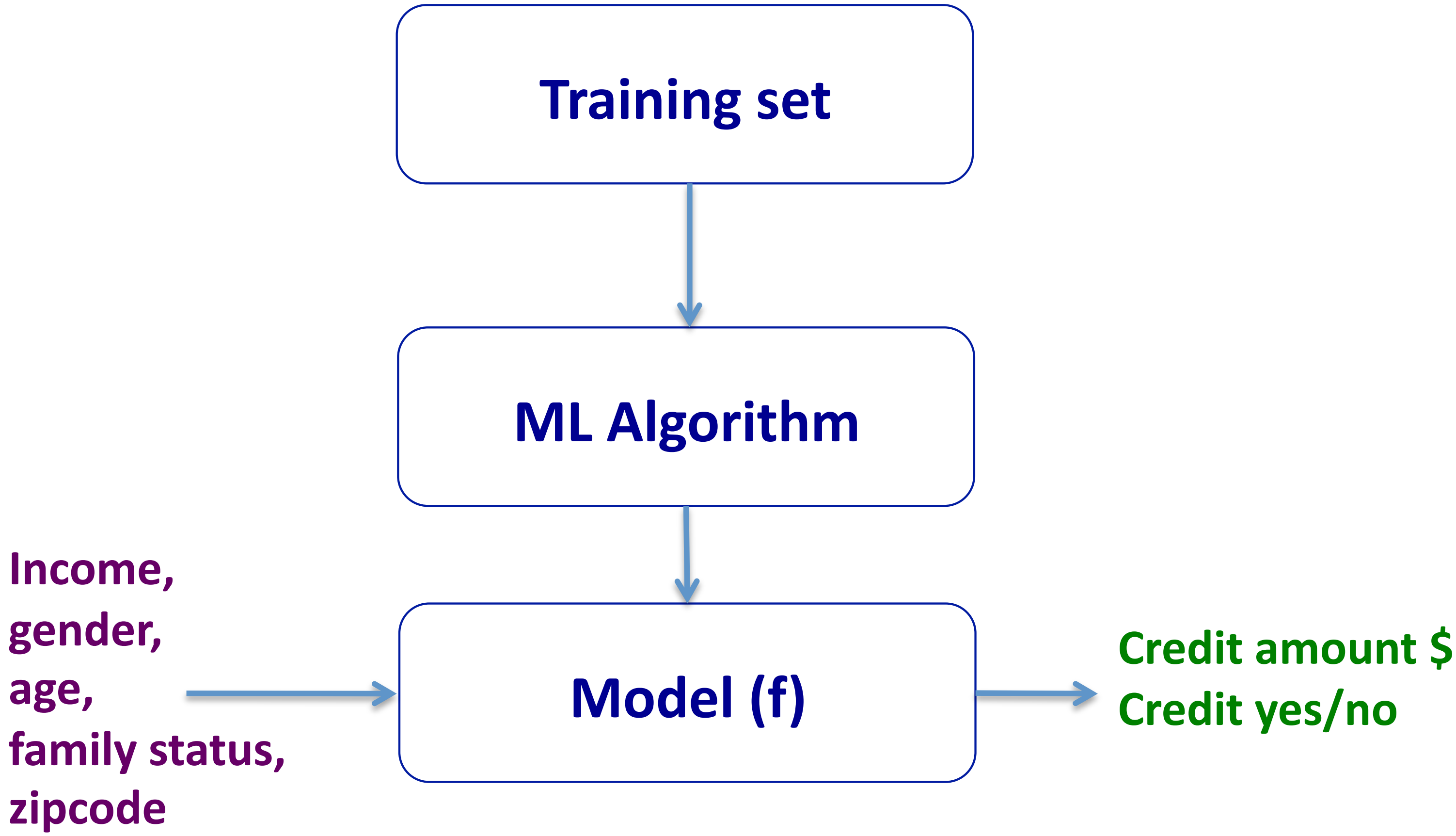
**Cons:**

- Requires large space to store the entire training dataset.
- Slow! Given $n$ examples and $d$ features. The method takes $O(n \times d)$ to run.
- Suffers from the *curse of dimensionality*.

# Applications of K-NN

1. Information retrieval.
2. Handwritten character classification using nearest neighbor in large databases.
3. Recommender systems (user like you may like similar movies).
4. Breast cancer diagnosis.
5. Medical data mining (similar patient symptoms).
6. Pattern recognition in general.

# Training and Testing




Question: How can we be confident about $f$?

# Training and Testing

- We calculate $E^{train}$ the in-sample error (training error or empirical error/risk).

$$E^{train}(f) = \sum_{i=1}^{n} \ell oss(y_i, f(x_i))$$

# Training and Testing

- We calculate $E^{train}$ the in-sample error (training error or empirical error/risk).

$$E^{train}(f) = \sum_{i=1}^{n} \ell oss(y_i, f(x_i))$$

- Examples of loss functions:

  – **Classification error:**

$$\ell oss(y_i, f(x_i)) = \begin{cases} 1 & \text{if } sign(y_i) \neq sign(f(x_i)) \\ 0 & \text{otherwise} \end{cases}$$

# Training and Testing

- We calculate $E^{train}$ the in-sample error (training error or empirical error/risk).

$$E^{train}(f) = \sum_{i=1}^{n} \ell oss(y_i, f(x_i))$$

- Examples of loss functions:

  - **Classification error:**

    $$\ell oss(y_i, f(x_i)) = \begin{cases} 1 & \text{if } sign(y_i) \neq sign(f(x_i)) \\ 0 & \text{otherwise} \end{cases}$$

  - **Least square loss:**

    $\ell oss(y_i, f(x_i)) = (y_i - f(x_i))^2$

# Training and Testing

- We calculate $E^{train}$ the in-sample error (training error or empirical error/risk).

$$E^{train}(f) = \sum_{i=1}^{n} \ell oss(y_i, f(x_i))$$

- We aim to have $E^{train}(f)$ small, i.e., minimize $E^{train}(f)$

# Training and Testing

- We calculate $E^{train}$ the in-sample error (training error or empirical error/risk).

$$E^{train}(f) = \sum_{i=1}^{n} \ell oss(y_i, f(x_i))$$

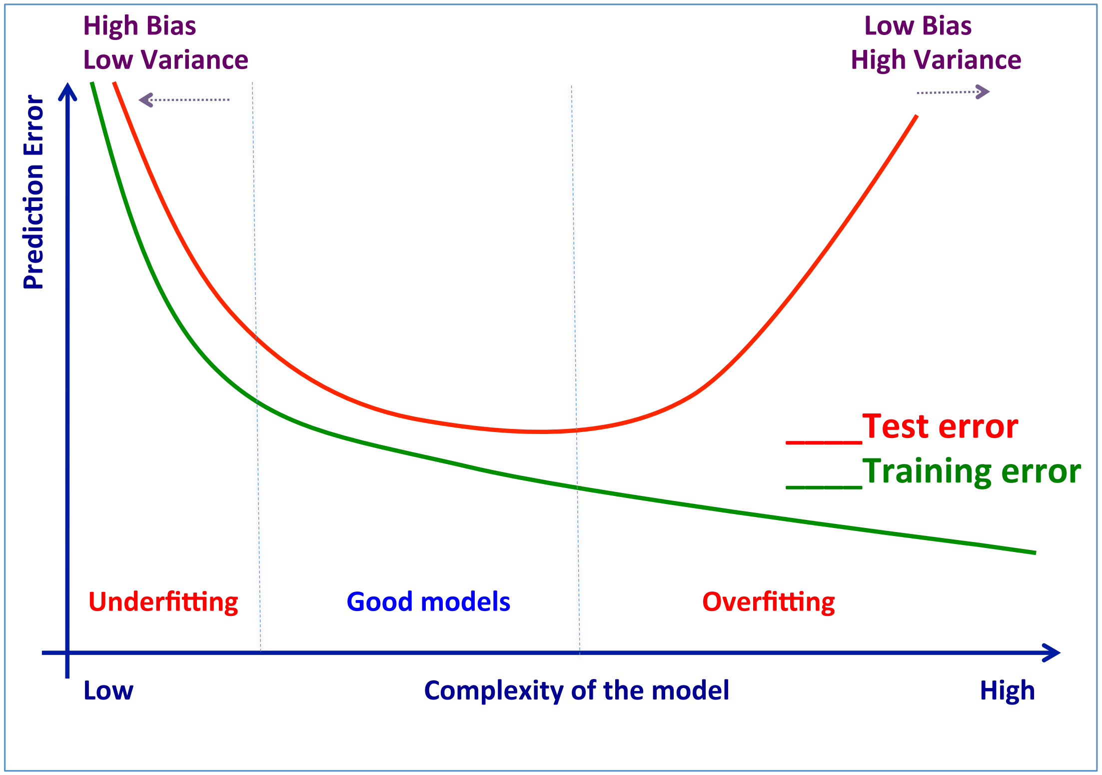
- We aim to have $E^{train}(f)$ small, i.e., minimize $E^{train}(f)$

- We hope that $E^{test}(f)$, the out-sample error (test/true error), will be small too.

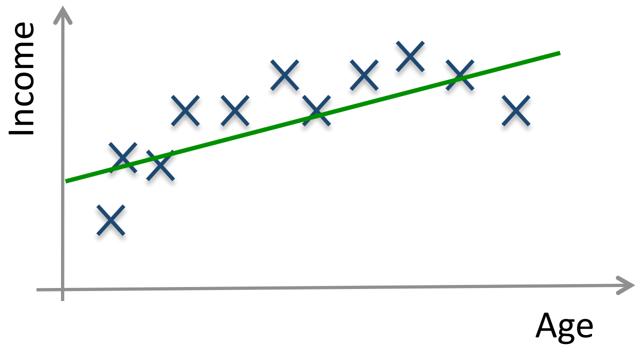
# Overfitting/underfitting

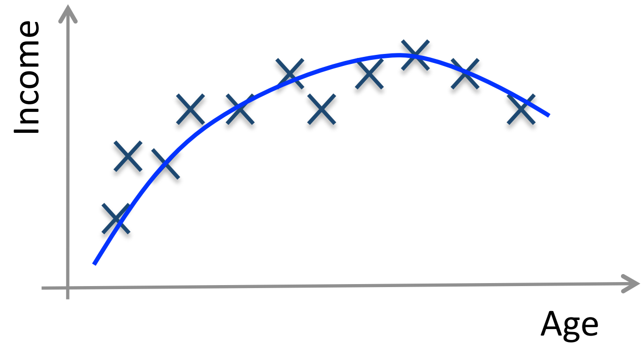
**An intuitive example**

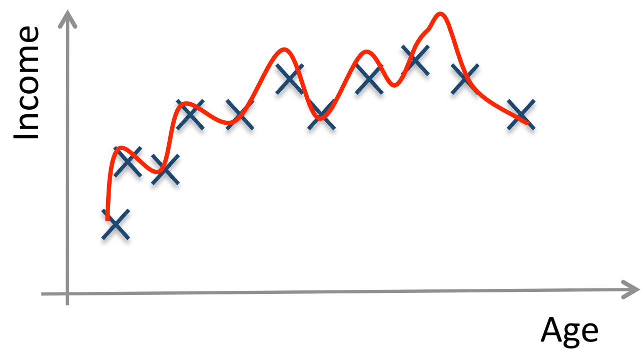
# Structural Risk Minimization



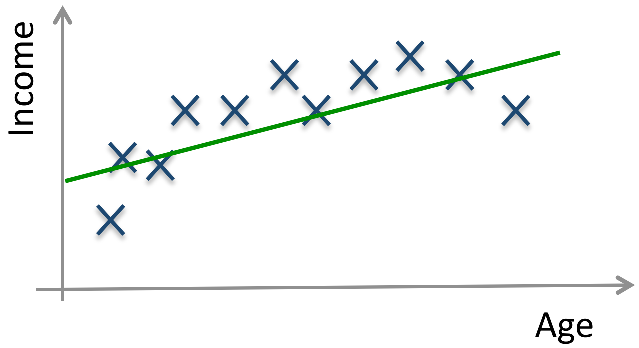

# Training and Testing

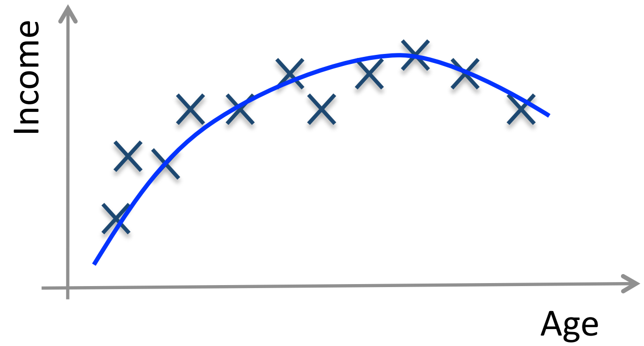


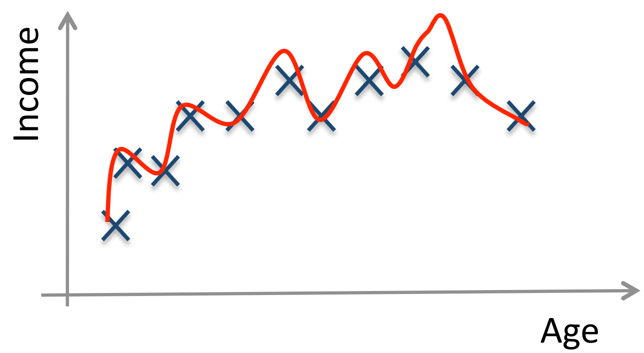



# Training and Testing

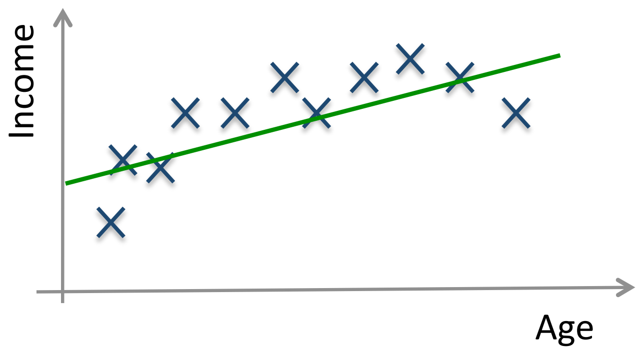


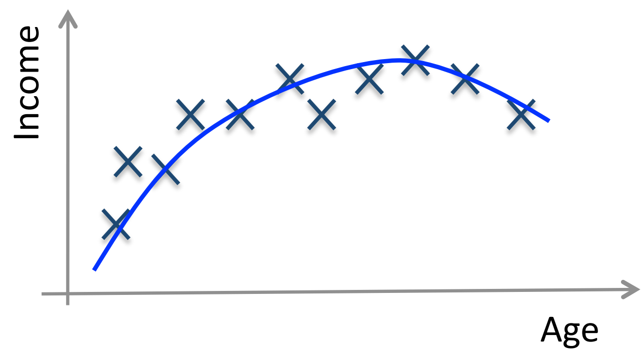




**High bias (underfitting)**

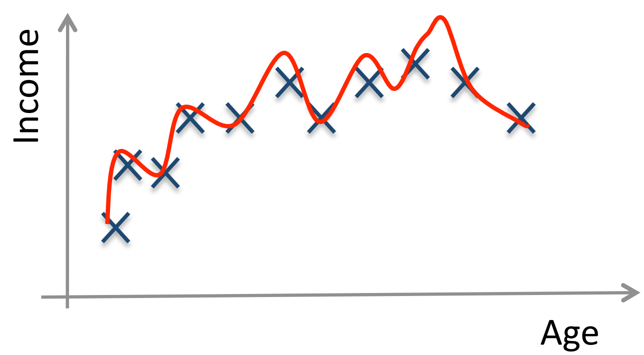


# Training and Testing



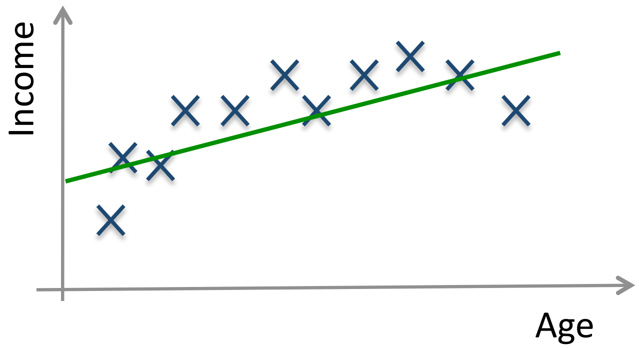




High bias (underfitting)

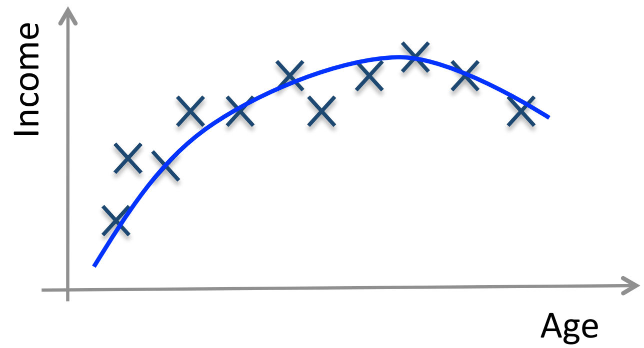



High variance (overfitting)

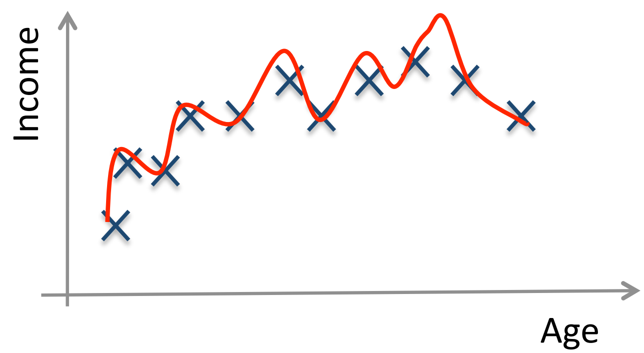
# Training and Testing




High bias (underfitting)




Just right!




High variance (overfitting)

# Avoid overfitting

In general, use simple models!

- **Reduce the number** of features manually or do feature selection.
- Do a **model selection** (ML course).
- Use **regularization** (keep the features but reduce their importance by setting small parameter values) (ML course).
- Do a **cross-validation** to estimate the test error.

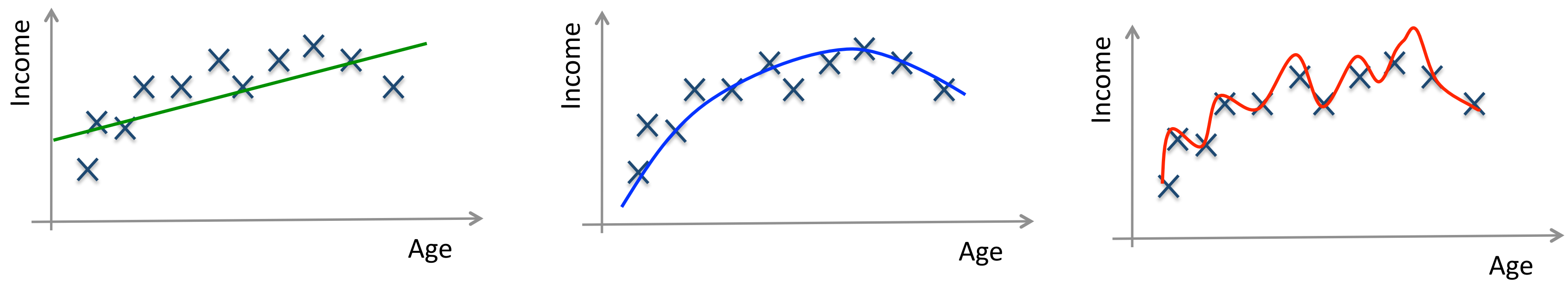
# Regularization: Intuition

We want to minimize:

$$\text{Classification term} + C \times \text{Regularization term}$$

$$\sum_{i=1}^{n} \ell oss(y_i, f(x_i)) + C \times R(f)$$

# Regularization: Intuition



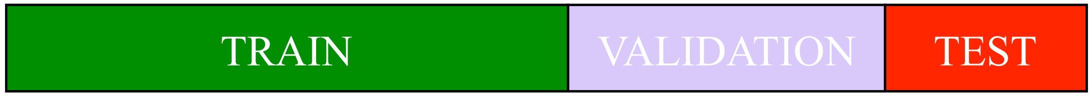

$$f(x) = \lambda_0 + \lambda_1 x \; \dots \; (1)$$

$$f(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 \; \dots \; (2)$$

$$f(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3 + \lambda_4 x^4 \; \dots \; (3)$$

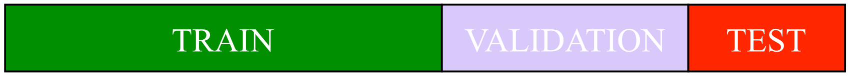
Hint: Avoid high-degree polynomials.

# Train, Validation and Test



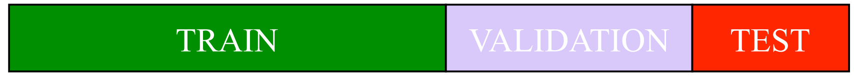

**Example:** Split the data randomly into 60% for training, 20% for validation and 20% for testing.

# Train, Validation and Test




1. Training set is a set of examples used for learning a model (e.g., a classification model).

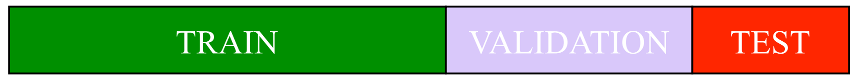
# Train, Validation and Test




1. Training set is a set of examples used for learning a model (e.g., a classification model).

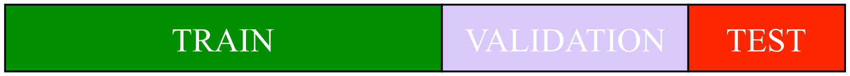
2. Validation set is a set of examples that cannot be used for learning the model but can help tune model parameters (e.g., selecting K in K-NN). Validation helps control overfitting.

# Train, Validation and Test




1. Training set is a set of examples used for learning a model (e.g., a classification model).
2. Validation set is a set of examples that cannot be used for learning the model but can help tune model parameters (e.g., selecting K in K-NN). Validation helps control overfitting.
3. Test set is used to assess the performance of the final model and provide an estimation of the test error.

# Train, Validation and Test




1. Training set is a set of examples used for learning a model (e.g., a classification model).

2. Validation set is a set of examples that cannot be used for learning the model but can help tune model parameters (e.g., selecting K in K-NN). Validation helps control overfitting.

3. Test set is used to assess the performance of the final model and provide an estimation of the test error.

**Note: Never use the test set in any way to further tune the parameters or revise the model**.

# K-fold Cross Validation

A method for estimating test error using training data.

**Algorithm:**

Given a learning algorithm $\mathcal{A}$ and a dataset $\mathcal{D}$

**Step 1:** Randomly partition $\mathcal{D}$ into $k$ equal-size subsets $\mathcal{D}_1, \ldots, \mathcal{D}_k$

**Step 2:**
For $j = 1$ to $k$
  Train $\mathcal{A}$ on all $\mathcal{D}_i$, $i \in 1, \ldots k$ and $i \neq j$, and get $f_j$.
  Apply $f_j$ to $\mathcal{D}_j$ and compute $E^{\mathcal{D}_j}$

**Step 3:** Average error over all folds.

$$\sum_{j=1}^{k} (E^{\mathcal{D}_j})$$

# Confusion matrix

| | | Actual Label | |
|---|---|---|---|
| | | Positive | Negative |
| **Predicted Label** | Positive | **True Positive (TP)** | **False Positive (FP)** |
| | Negative | **False Negative (FN)** | **True Negative (TN)** |

# Evaluation metrics

| | | Actual Label | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted Label | Positive | True Positive (TP) | False Positive (FP) |
| | Negative | False Negative (FN) | True Negative (TN) |

| | | |
|---|---|---|
| **Accuracy** | (TP + TN) / (TP + TN + FP + FN) | The percentage of predictions that are correct |
| **Precision** | TP / (TP + FP) | The percentage of positive predictions that are correct |
| **Sensitivity (Recall)** | TP / (TP + FN) | The percentage of positive cases that were predicted as positive |
| **Specificity** | TN / (TN + FP) | The percentage of negative cases that were predicted as negative |

# Terminology review

**Review the concepts and terminology:**

**Instance, example, feature, label, supervised learning, unsupervised learning, classification, regression, clustering, prediction, training set, validation set, test set, K-fold cross validation, classification error, loss function, overfitting, underfitting, regularization.**

# Machine Learning Books


1. Tom Mitchell, Machine Learning.

2. Abu-Mostafa, Yaser S. and Magdon-Ismail, Malik and Lin, Hsuan-Tien, Learning From Data, AMLBook.

3. The elements of statistical learning. Data mining, inference, and prediction T. Hastie, R. Tibshirani, J. Friedman.

4. Christopher Bishop. Pattern Recognition and Machine Learning.

5. Richard O. Duda, Peter E. Hart, David G. Stork. Pattern Classification. Wiley.

# Machine Learning Resources

- Major journals/conferences: ICML, NIPS, UAI, ECML/PKDD, JMLR, MLJ, etc.
- Machine learning video lectures:
  http://videolectures.net/Top/Computer_Science/Machine_Learning/
- Machine Learning (Theory):
  http://hunch.net/
- LinkedIn ML groups: "Big Data" Scientist, etc.
- Women in Machine Learning:
  https://groups.google.com/forum/#!forum/women-in-machine-learning
- KDD nuggets http://www.kdnuggets.com/

# Credit

- The elements of statistical learning. Data mining, inference, and prediction. 10th Edition 2009. T. Hastie, R. Tibshirani, J. Friedman.
- Machine Learning 1997. Tom Mitchell.