

# Data Science

Deriving Knowledge from Data at Scale

Wee Hyong Tok

April 12, 2017



X



Like

Comment

Share

# Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
- Adaboost
  - Friedman, Hastie, and Tibshirani, “Additive logistic regression: a statistical view of boosting”, Annals of Statistics, 2000
- SVMs
  - <http://www.support-vector.net/icml-tutorial.pdf>



# Assignment 1 Discussion





Deriving Knowledge from Data at Scale



Robert M. Parker, Jr.

## Wine Quality

Orley Ashenfelter (Princeton Economist)

**Wine Quality** = 12.145 + 0.00117 x (winter rainfall) + 0.0614 x (average growing season temperature)-0.00386 x (harvest rainfall).

# Assignment 1

Design a simple, low-cost sensor that can distinguish between red wine and white wine for at least 95% of the samples. Your technology is capable of sensing the following wine attributes:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Density
- Free sulfur dioxide
- Total sulfur dioxide
- Sulfates
- pH
- Alcohol

Sense as few of these attributes as possible to meet ~95%.



**Objective: Gain familiarity with using a machine learning platform**

**Objective: Experience building and evaluating decision trees;**

**Objective: Interpreting, add/drop attributes – not just for making a prediction;**

# Overview

- Doing Data Science
- Bias and Variance
- Supervised and Unsupervised
- Classification



“The goal of machine learning is to build computer systems that can adapt and learn from their experience.”

– Tom Dietterich

# Doing Data Science



# The Data Science Workflow

## Review

Stay in the **immediate zone** during exploratory modeling, to extent possible:

- 5 to 10 minutes per experiment, results in 100's per day;
- Small, statistically sound/relevant samples;
- Linear modelling during feature exploration;
- Don't write ML algorithms, use packages, do learn to write data manipulation code;

Start with a sample of data, *as soon as you can get it...*

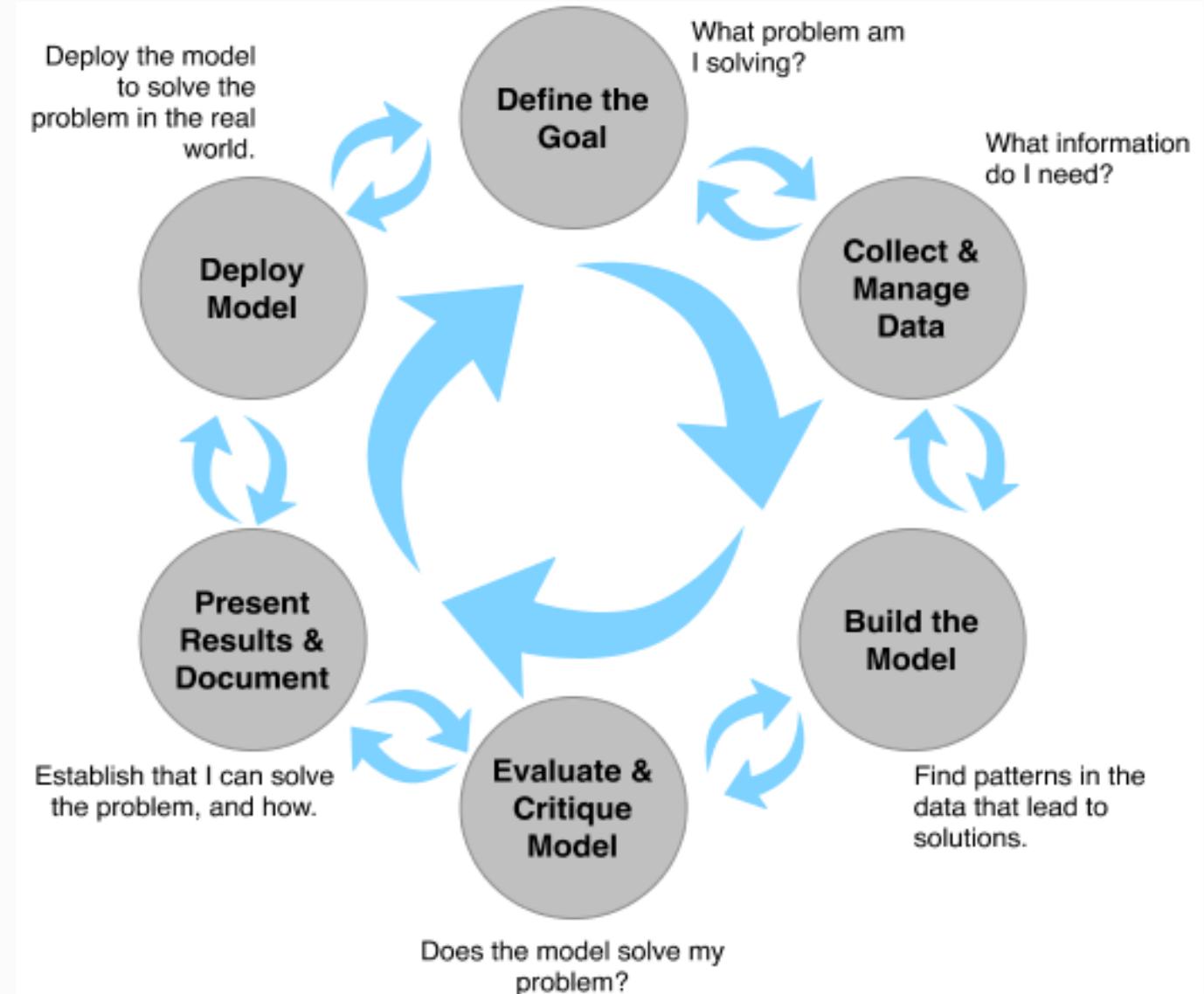
- You will learn a considerable amount from that first data sample
- Quality of the data, what fields are being collected, possible missing values and/or missing fields, and the questions will begin to flow (dialogue with customer will be at a much more meaningful level).

If your customer can't quantify it, you can't change/improve it...



# The Data Science Workflow

## *Loops within loops...*



# The Data Science Workflow

## Define the Goal

The first task in a data science project is to define a **measurable & quantifiable** goal. At this stage, *learn all that you can about the context of your project.*

- Why do the project sponsors want the project in the first place? What do they lack, and what do they need?
- What are they doing to solve the problem now, and why isn't that good enough?
- What resources will you need: what kind of data, how much staff, will you have domain experts to collaborate with, what are the computational resources?
- How do the project sponsors plan to deploy your results? What are the constraints that have to be met for successful deployment?

**Not** We want to get better at finding bad loans.

**But** We want to reduce our rate of **loan charge-offs by at least 10%**, using a model that predicts which loan applicants are likely to default.



*A concrete goal begets concrete **stopping conditions** and concrete **acceptance criteria**. The less specific the goal, the likelier that the project will go unbounded, because no result will be "good enough." If you don't know what you want to achieve, you don't know when to stop trying – or even what to try. When the project eventually terminates – because either time or resources run out – **no one will be happy with the outcome...***

# The Data Science Workflow

## Data Collection and Management

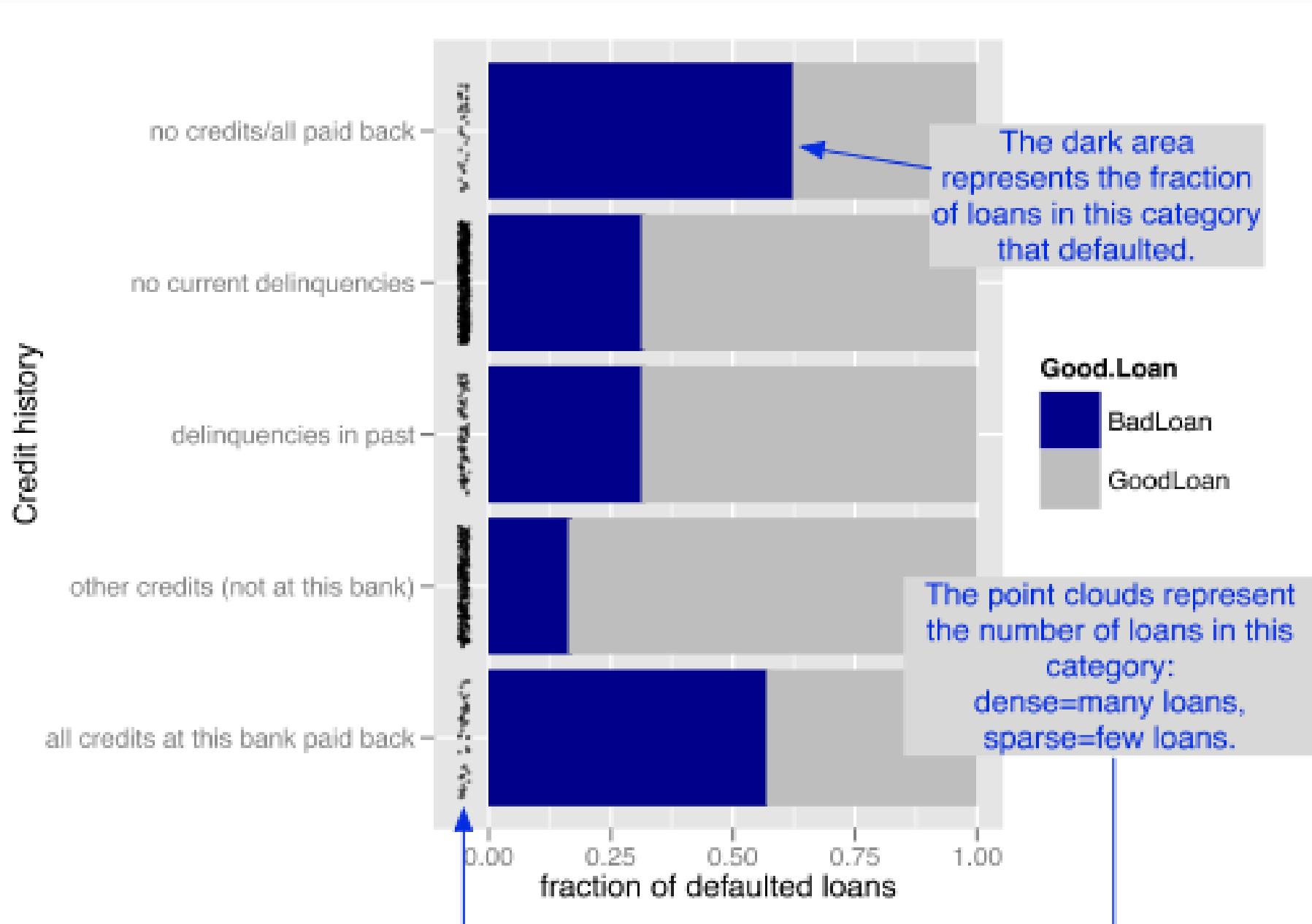
This step encompasses identifying the data you need, exploring it, and conditioning it to be suitable for analysis. This stage is often the most time-consuming step in the process. It's also one of the most important.

- What data is available to me?
- Will it help me solve the problem?
- Is it enough?
- Is it of good enough quality?

**Rule** First piece of data is *very informative*, data set utility is roughly logarithmic in size.

**Prefer** Direct measurements, but if they are not available identify proxy variables.





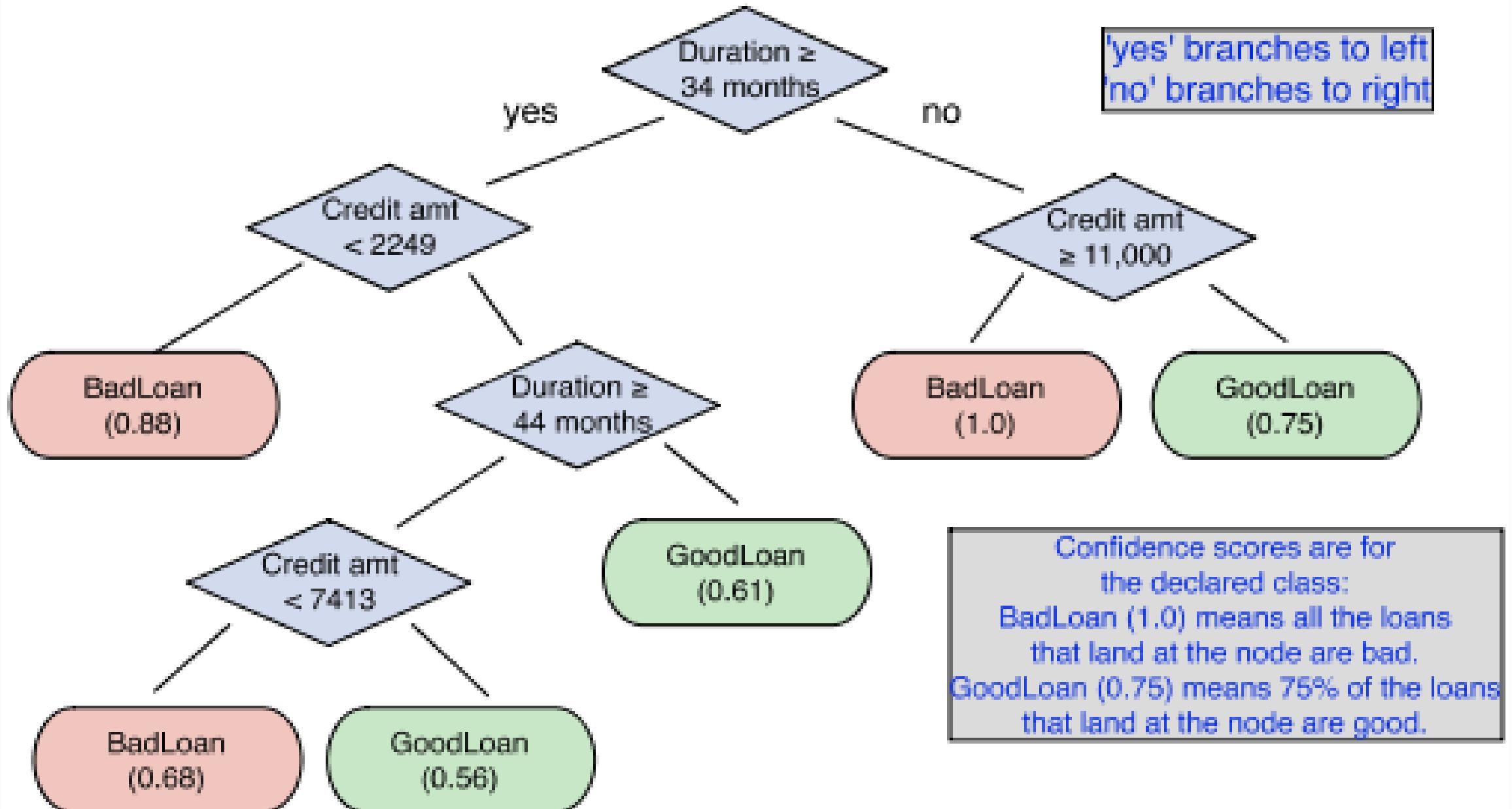
# The Data Science Workflow

## Modeling

We get to statistics and machine learning during the modeling stage. Here is where you try to extract useful insights from the data. Many modeling procedures make specific assumptions about data distribution and relationships, *there will be back-and-forth between the modeling and data cleaning stages as you try to find the best way to represent and model the data.* The most common data science modeling tasks are:

- **Classification:** deciding if something belongs to one category or another.
- **Scoring:** predicting or estimating a numeric value such as a price or probability.
- **Ranking:** learning to order items by preferences.
- **Clustering:** grouping items into most similar groups.
- **Finding Relations:** finding correlations or potential causes of effects seen in the data.
- **Characterization:** very general plotting and report generation from data.





# The Data Science Workflow

## Model Evaluation and Critique

Once you have a model, you need to determine if it meets your goals.

- Is it accurate enough for your needs?
- Does it generalize well?
- Does it perform better than "the obvious guess"? Better than whatever estimate you currently use?
- Do the results of the model (coefficients, clusters, rules) make sense in the context of the problem domain?

If you've answered "no" to any of the above questions, it's time to loop back to the modeling step — or decide that the data doesn't support the goal you are trying to achieve.



# The Data Science Workflow

## Model Deployment and Maintenance

Finally, the model is put into operation.

In many organizations this means the data scientist no longer has primary responsibility for the day-to-day operation of the model. However, you still should ***ensure that the model will run smoothly*** and will not make disastrous unsupervised decisions. You also want to make sure that the ***model can be updated*** as its environment changes. And in many situations, the model will initially be ***deployed in a small pilot program***.

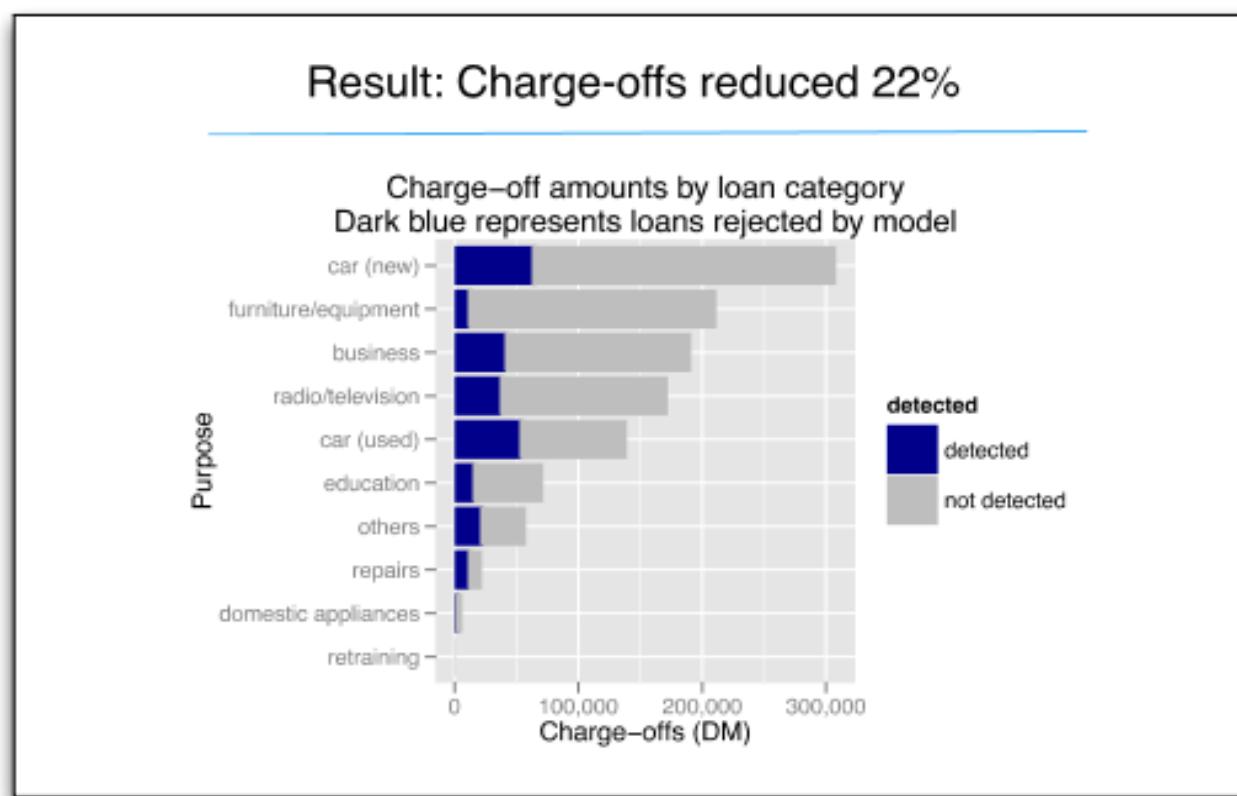
The test might bring out issues that you didn't anticipate, and you will have to adjust the model appropriately.



# The Data Science Workflow

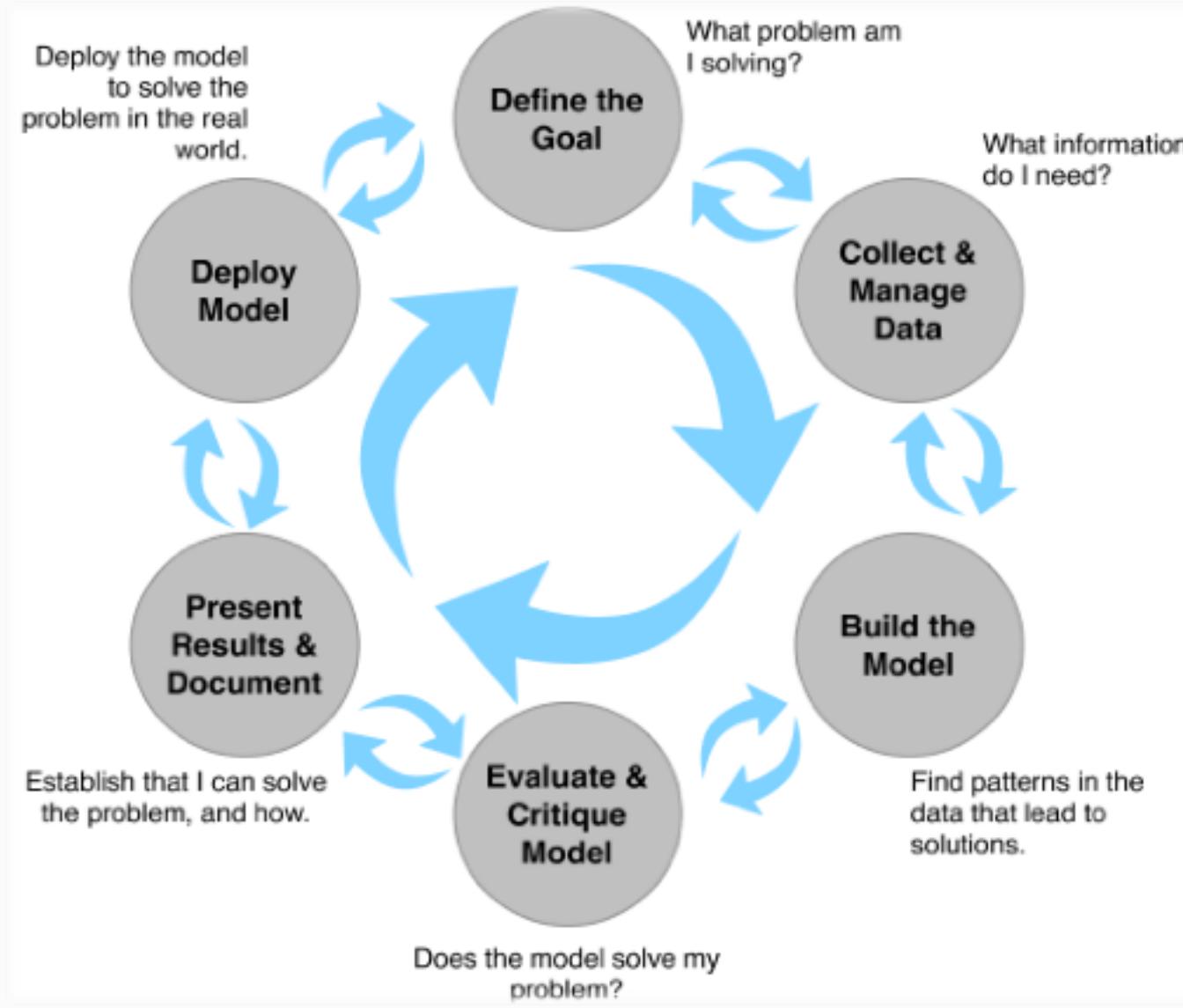
## Presentation and Documentation

Once you have a model that meets your success criteria, you will present your results to your project sponsor and other stakeholders. You must also document the model for those in the organization who are responsible for using, running, and maintaining the model once it has been deployed. ***Model interpretability may be an issue...***



# The Data Science Workflow

## *Loops within loops...*



# Supervised vs Un-supervised



# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.



# Example Application 1 - Healthcare

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed:** whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem:** to predict **high-risk patients** and discriminate them from **low-risk patients**.



# Example Application 2 – Financial Institution

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- **Problem:** to decide whether an application should be approved, or to classify applications into two categories, **approved** and **not approved**.



# The data and the goal

- **Data:** A set of data records (also called examples, instances or cases) described by
  - $k$  attributes:  $A_1, A_2, \dots A_k$ .
  - a class: Each example is labelled with a pre-defined class.
- **Goal:** To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# An example: data (loan application)

*Approved or not*

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

# An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

# Supervised vs. Unsupervised Learning

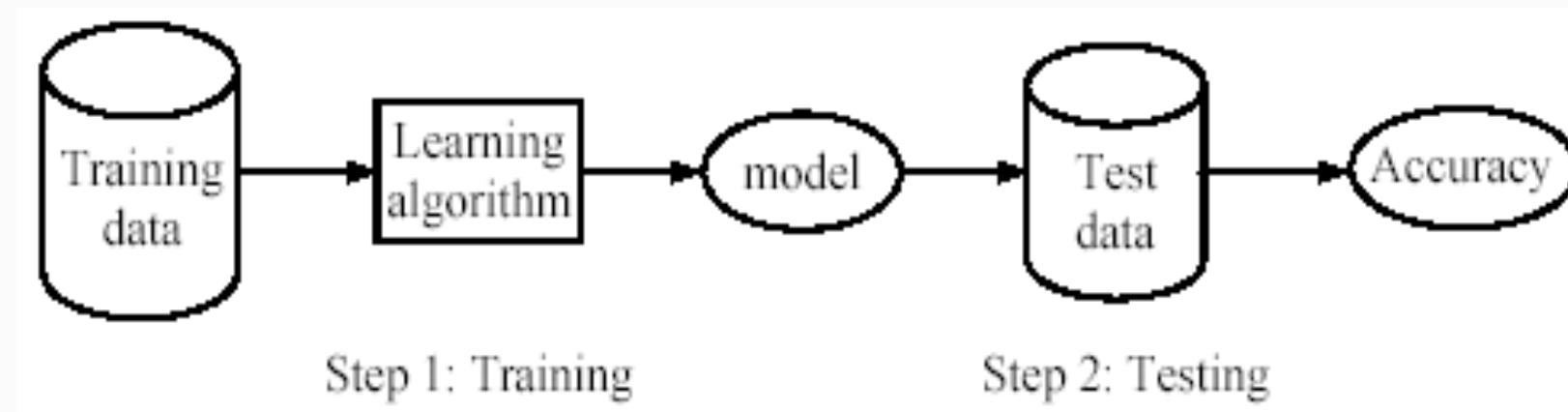
- **Supervised learning:** classification is seen as supervised learning from examples.
  - **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (**supervision**).
  - Test data are classified into these classes too.
- **Unsupervised learning (clustering)**
  - **Class labels of the data are unknown**
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised learning process: two steps

**Learning (training):** Learn a model using the **training data**

**Testing:** Test the model using **unseen test data** to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# What do we mean by learning?

- Given
    - a data set  $D$ ,
    - a task  $T$ , and
    - a performance measure  $M$ ,
  - In other words, the learned model helps the system to perform  $T$  better as compared to no learning.
- a computer system is said to **learn** from  $D$  to perform the task  $T$  if after learning the system's performance on  $T$  improves as measured by  $M$ .



# An example

- **Data:** Loan application data
- **Task:** Predict whether a loan should be approved or not.
- **Performance measure:** accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

$$\text{Accuracy} = 9/15 = 60\%.$$

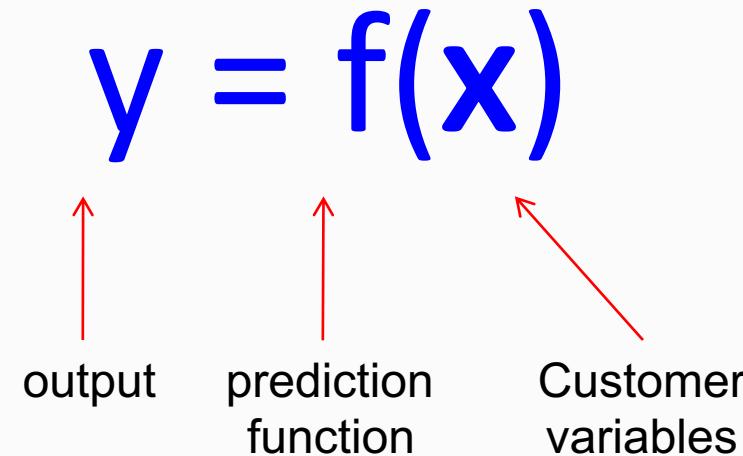
- We can do better than 60% with learning.

# Fundamental assumption of learning

**Assumption:** The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

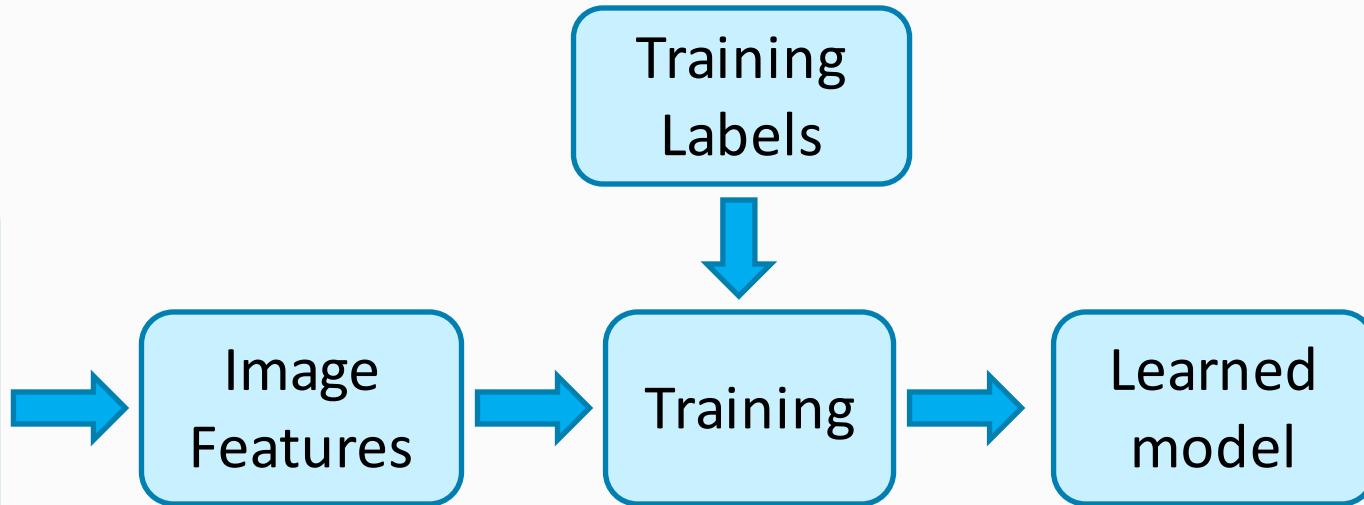
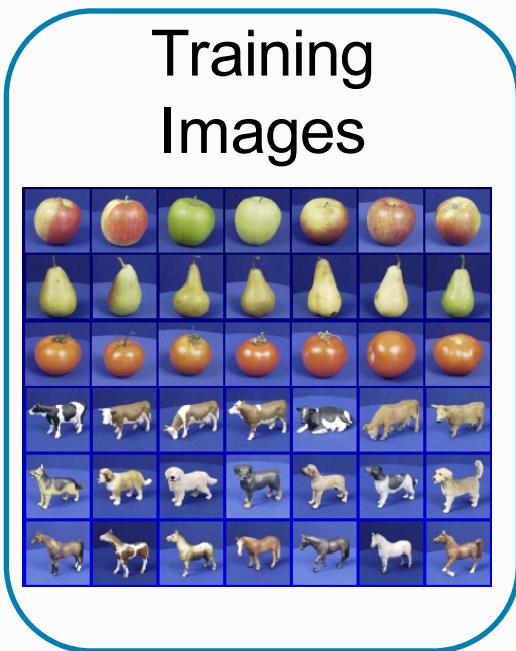
# The machine learning framework



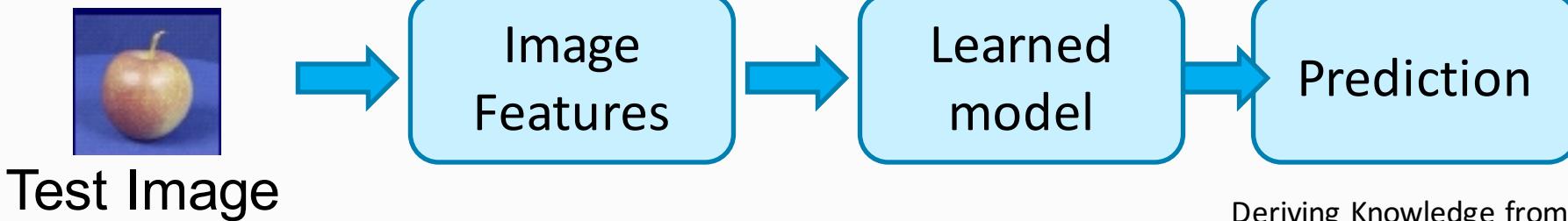
- Training: given a *training set* of labeled examples  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ , estimate the prediction function  $f$  by minimizing the prediction error on the training set
- Testing: apply  $f$  to a never before seen *test example*  $\mathbf{x}$  and output the predicted value  $y = f(\mathbf{x})$

# Steps

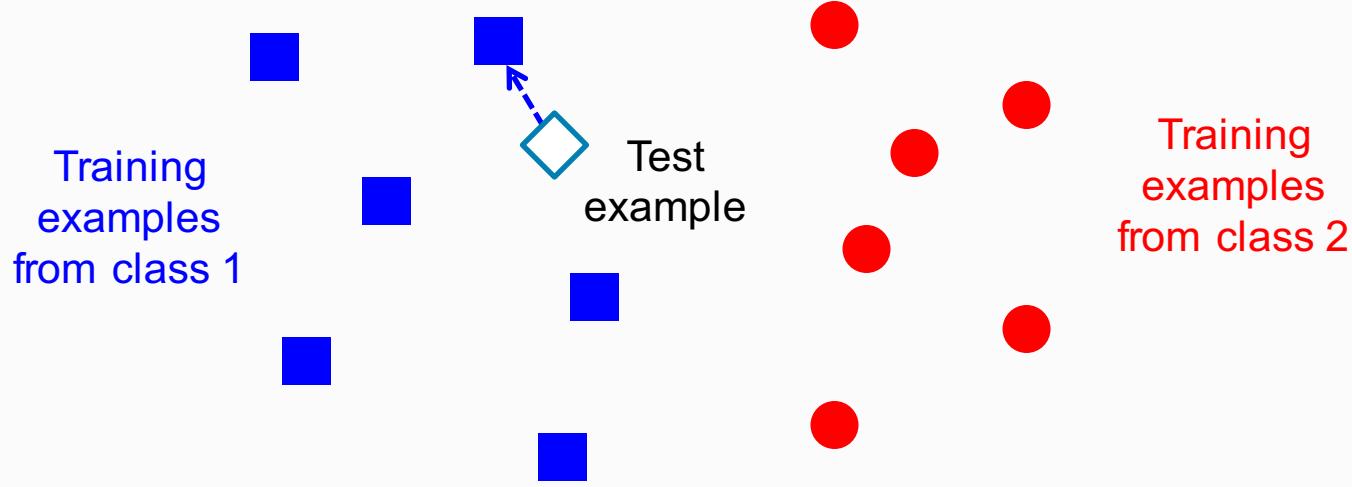
## Training



## Testing



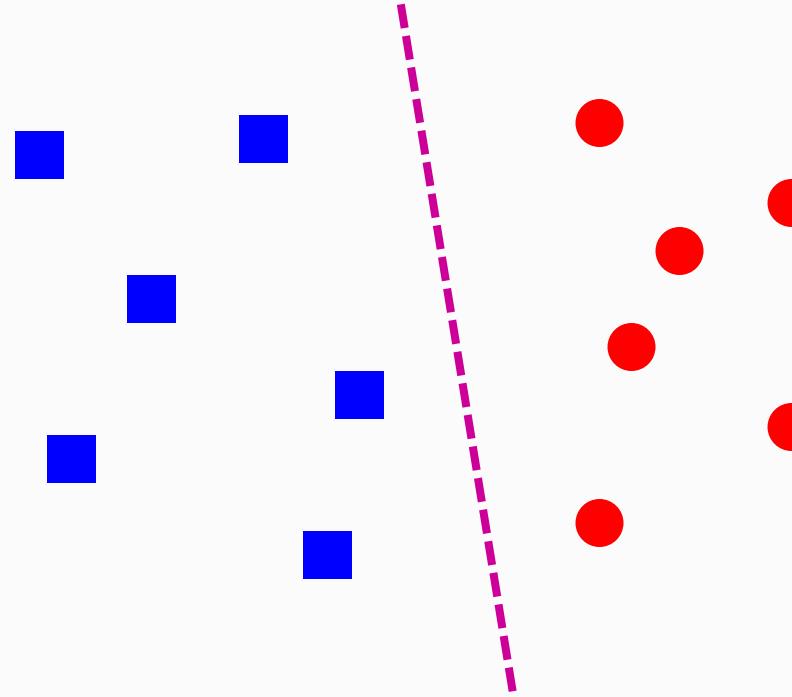
# Classifiers: Nearest neighbor



$f(x) = \text{label of the training example nearest to } x$

- All we need is a distance function for our inputs
- No training required!

# Classifiers: Linear



- Find a *linear function* to separate the classes:

$$f(x) = \text{sgn}(w \cdot x + b)$$

# Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- Etc.

*Which is the best one?*



# Generalization



Training set (labels known)



Test set (labels unknown)

How well does a learned model generalize from the data it was trained on to a new test set?

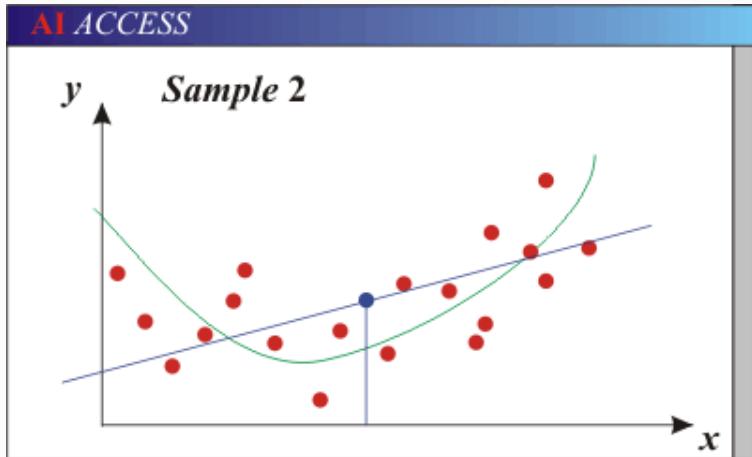
# Generalization

- Components of generalization error
  - Bias: how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - Variance: how much models estimated from different training sets differ from each other
- Underfitting: model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

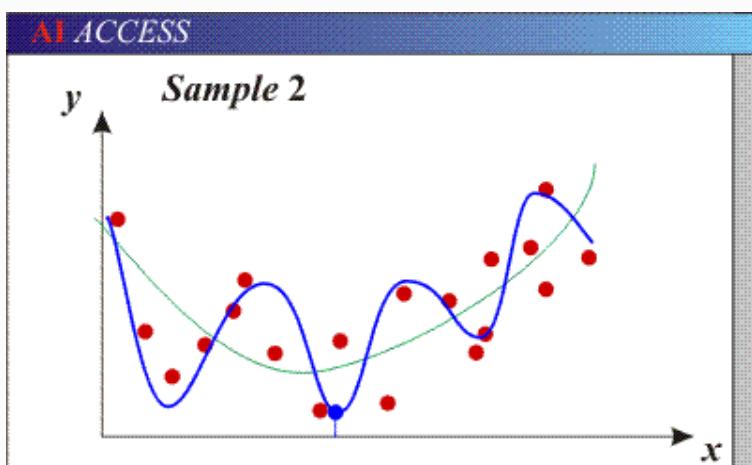
# Bias Variance Tradeoff



# Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

# Bias-Variance Trade-off

$$E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable  
error

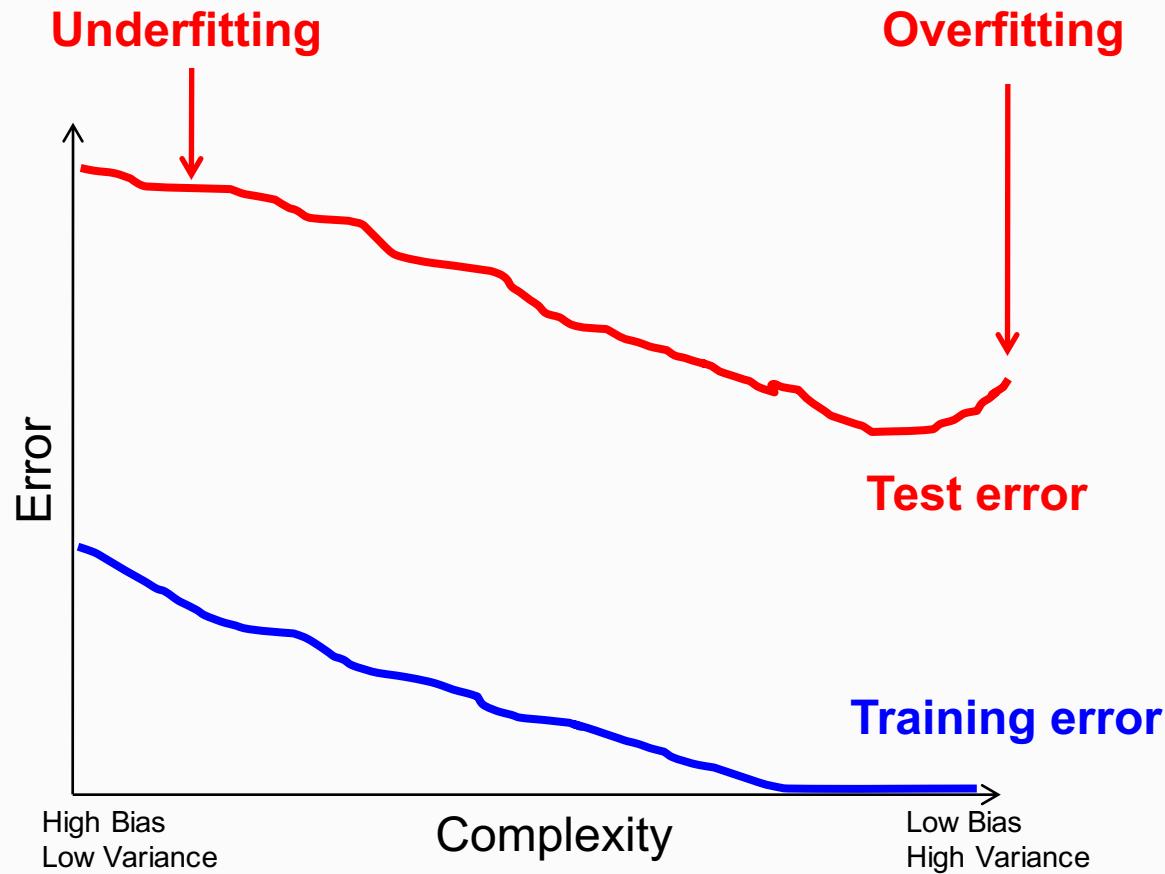
Error due to  
incorrect  
assumptions

Error due to variance  
of training samples

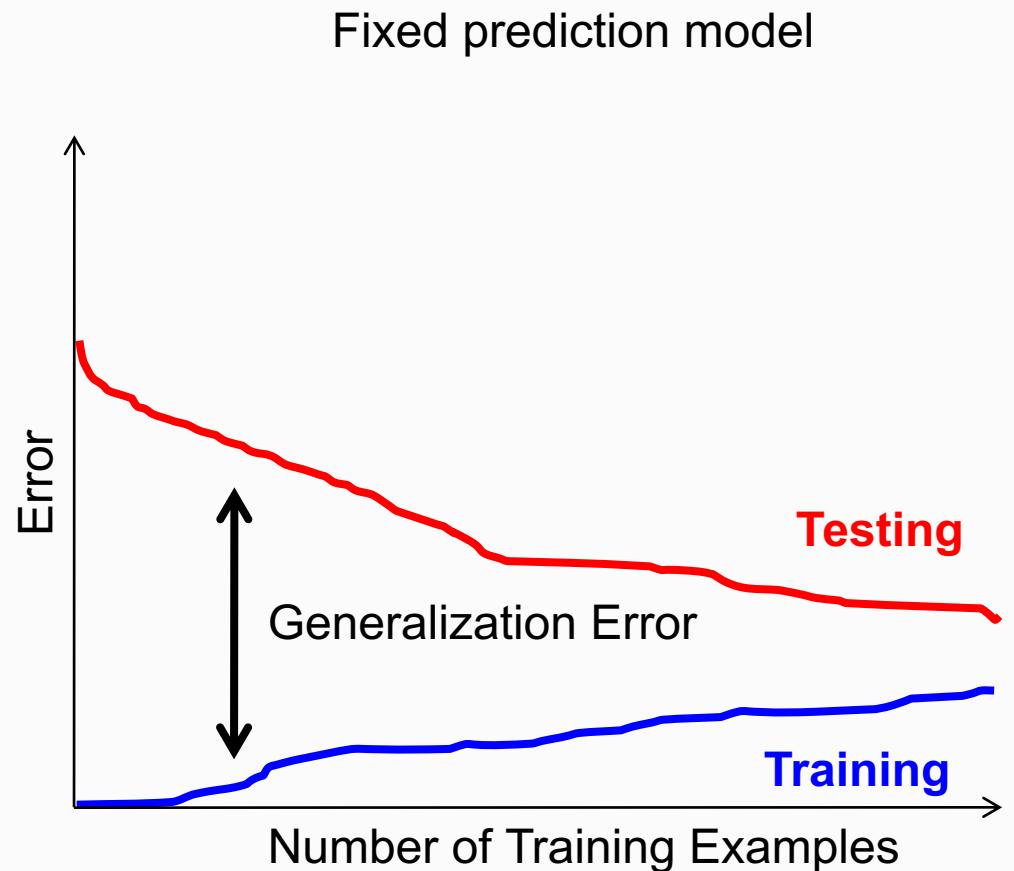
See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- <http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

# Bias-variance tradeoff



# Effect of Training Size



# Classification



# Remember...

- No classifier is inherently better than any other: you need to make assumptions to generalize
- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data



# How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data



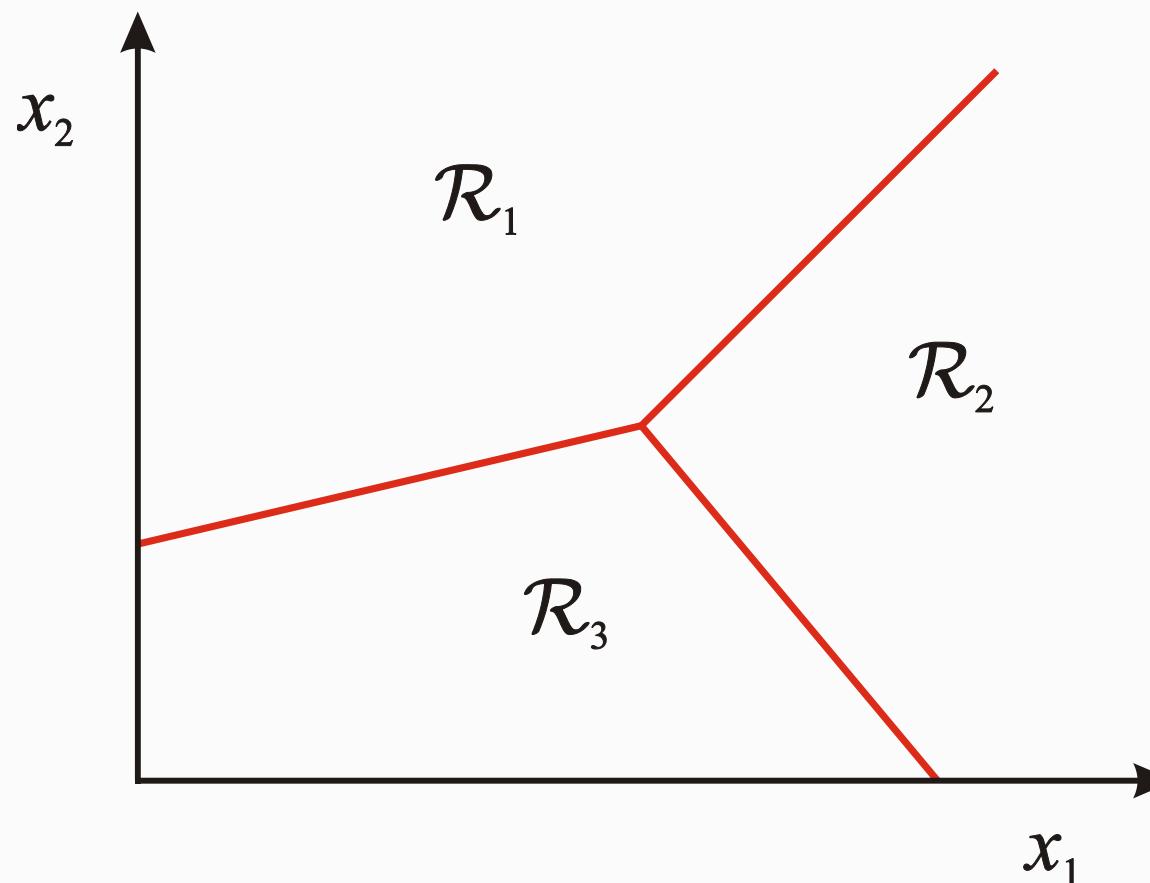
# Very brief tour of some classifiers

- K-nearest neighbor
- SVM
- Boosted Decision Trees
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Etc.



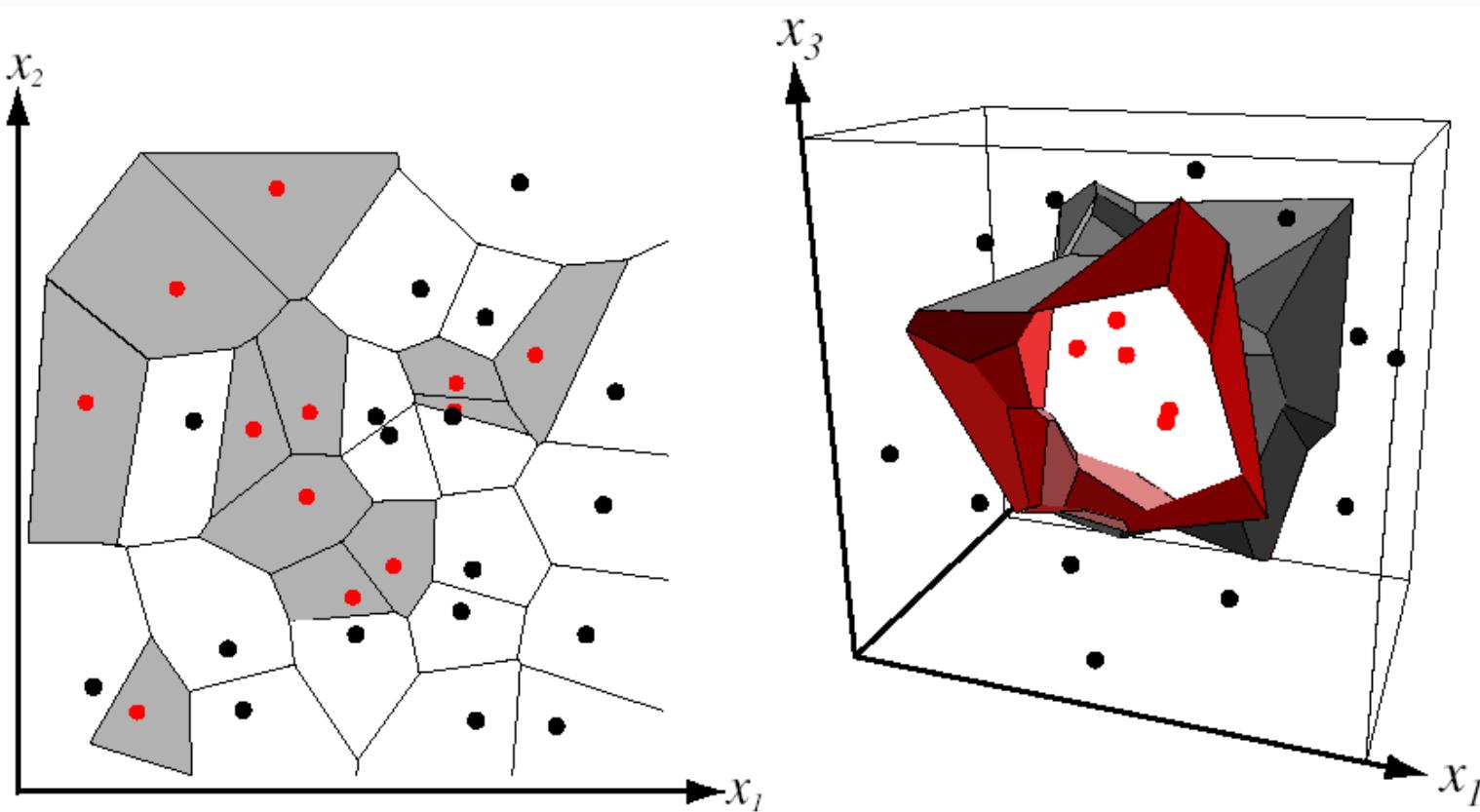
# Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



# Nearest Neighbor Classifier

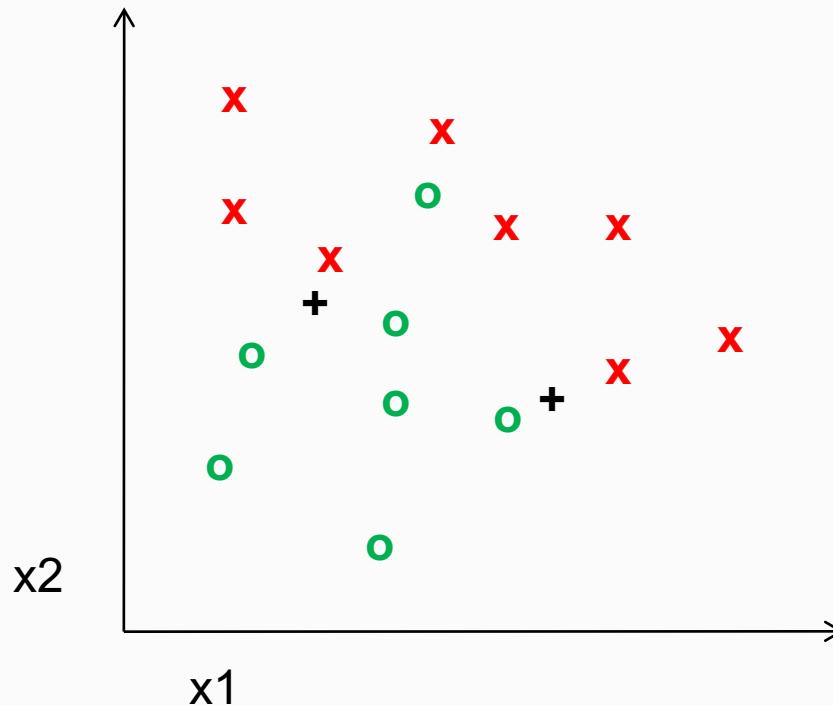
- Assign label of nearest training data point to each test data point



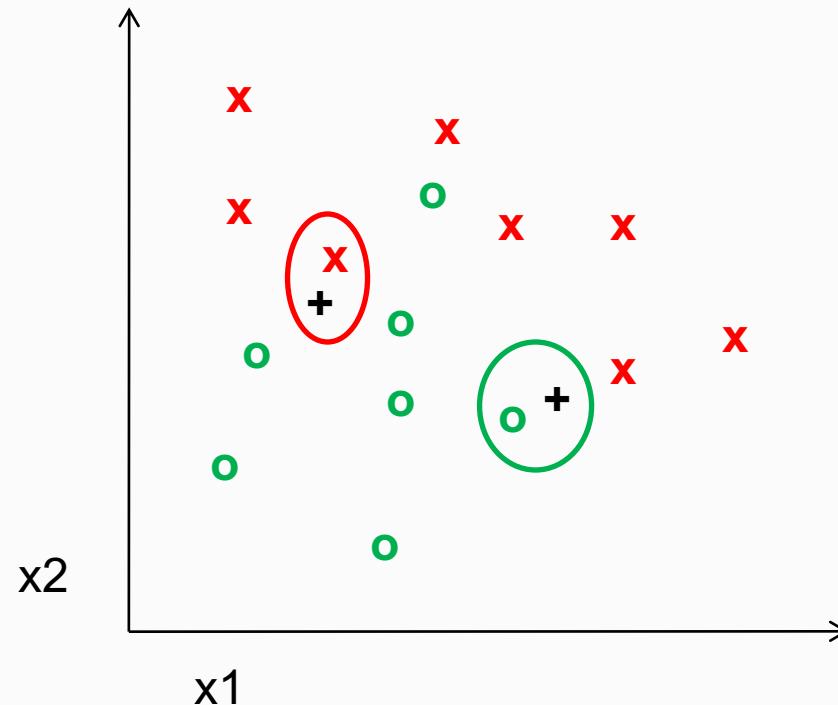
from Duda *et al.*

Voronoi partitioning of feature space  
for two-category 2D and 3D data

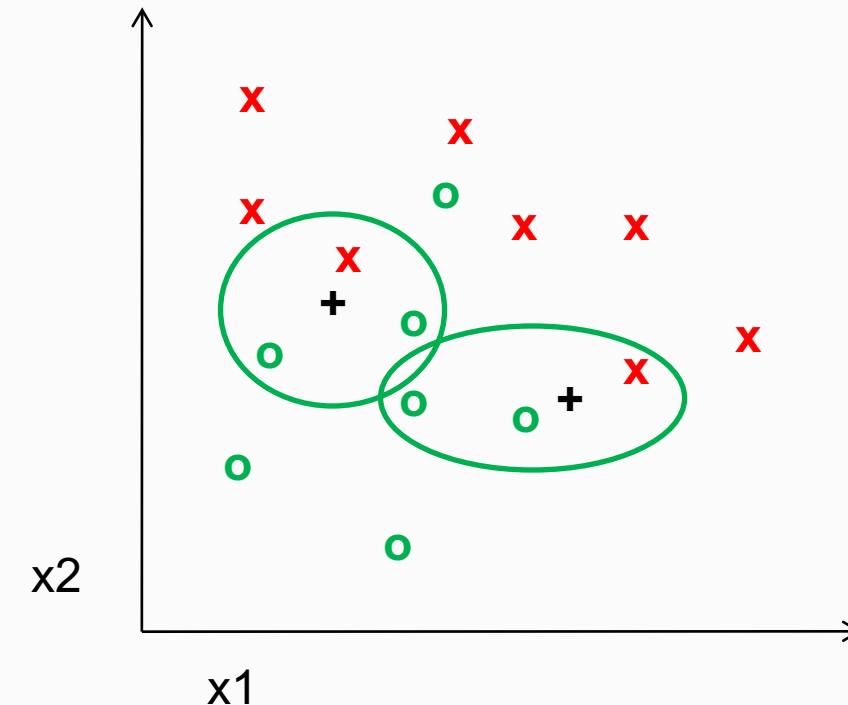
# K-nearest neighbor



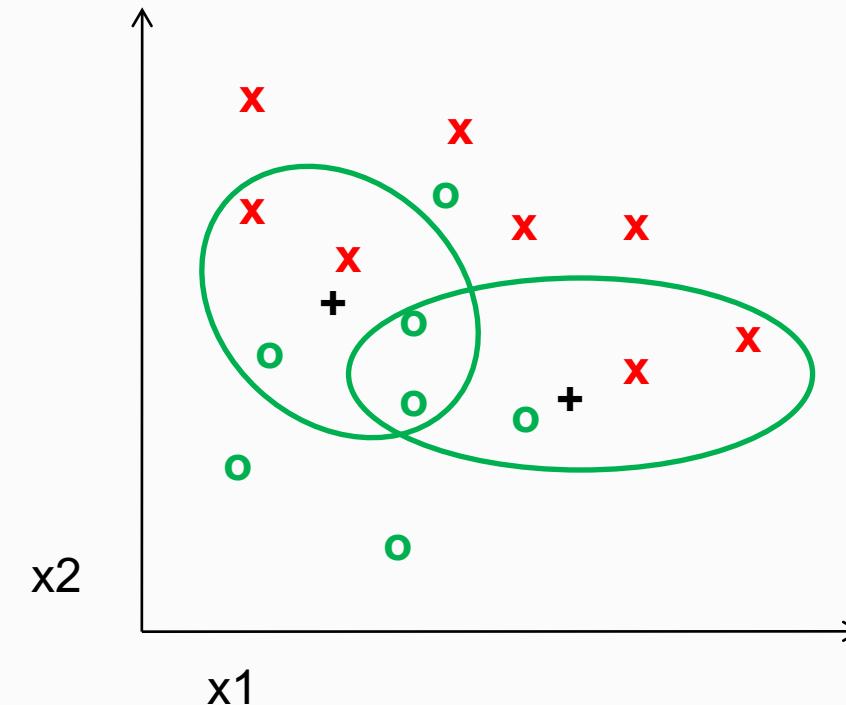
# 1-nearest neighbor



# 3-nearest neighbor



# 5-nearest neighbor

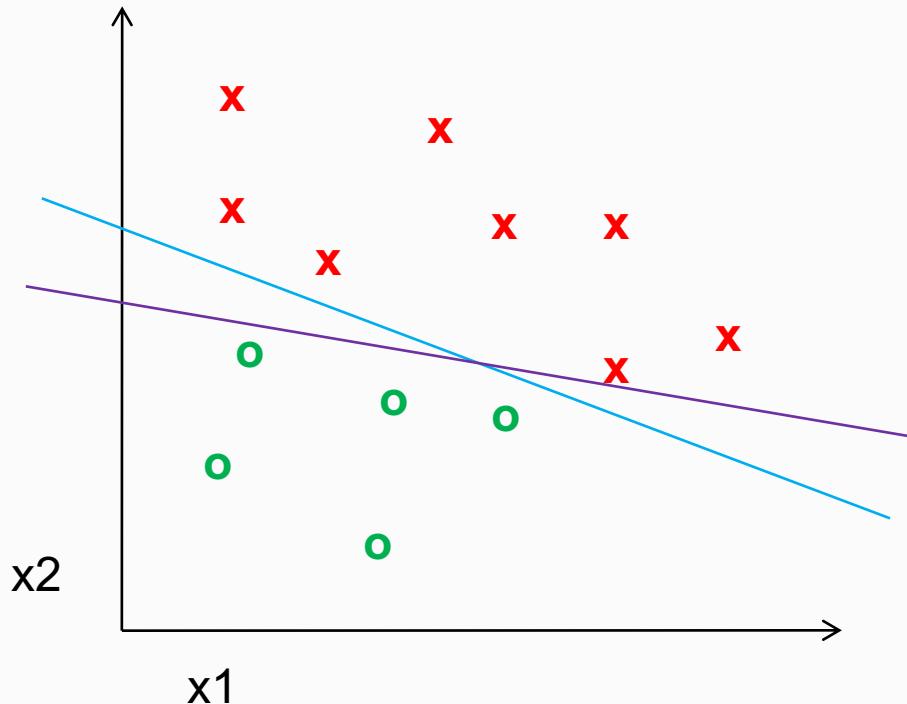


# Using K-NN

- Simple, a good one to try first
- With infinite examples, 1-NN probably has error that is at most twice Bayes optimal error



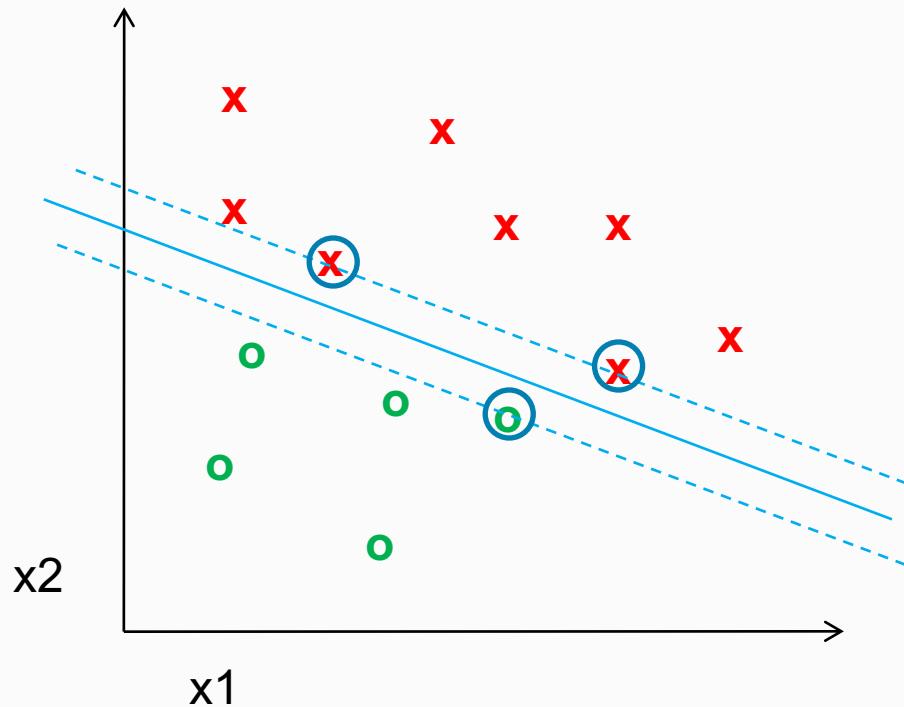
# Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

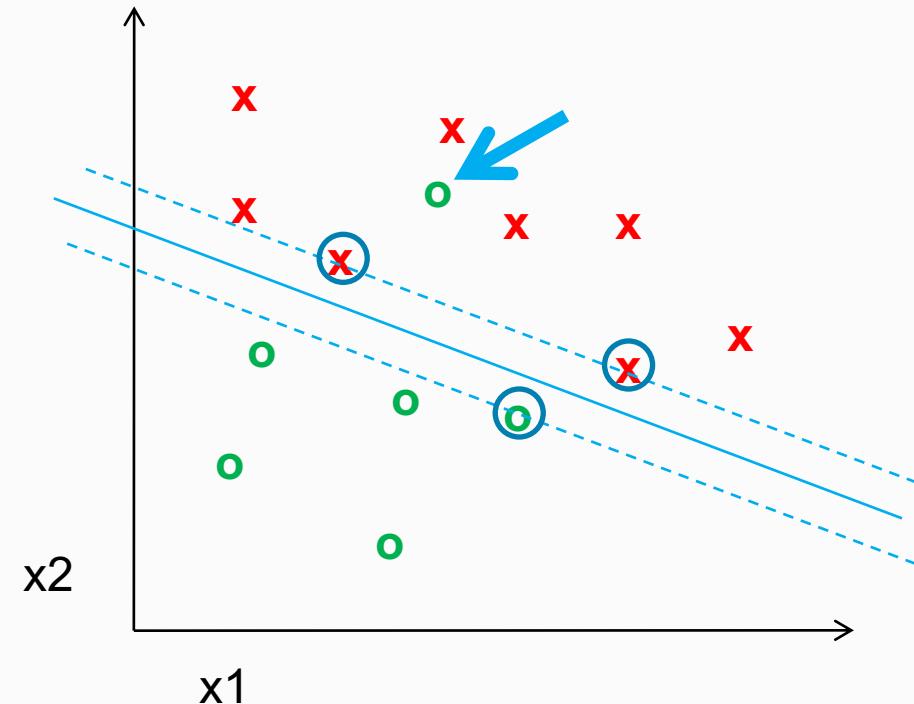
# Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Classifiers: Linear SVM

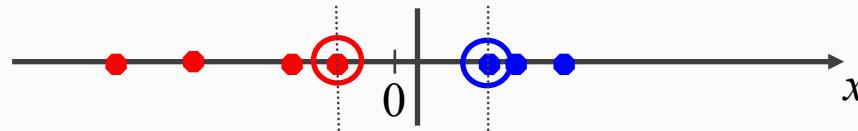


- Find a *linear function* to separate the classes:

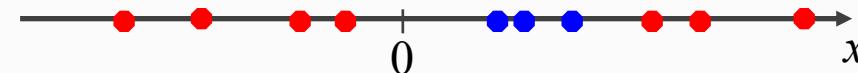
$$f(x) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Nonlinear SVMs

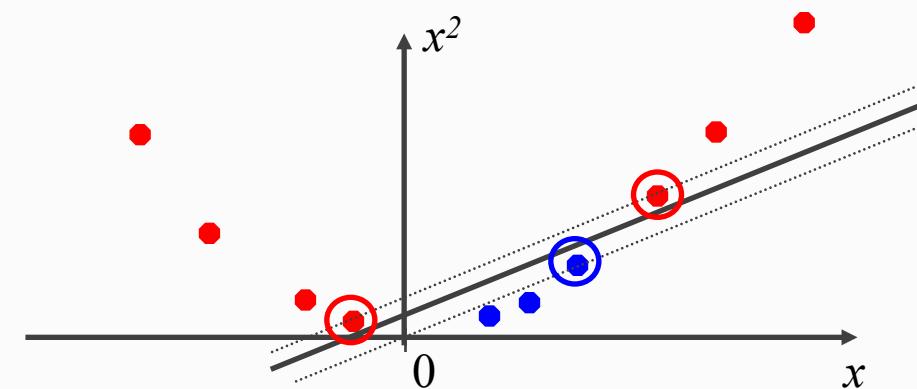
- Datasets that are linearly separable work out great:



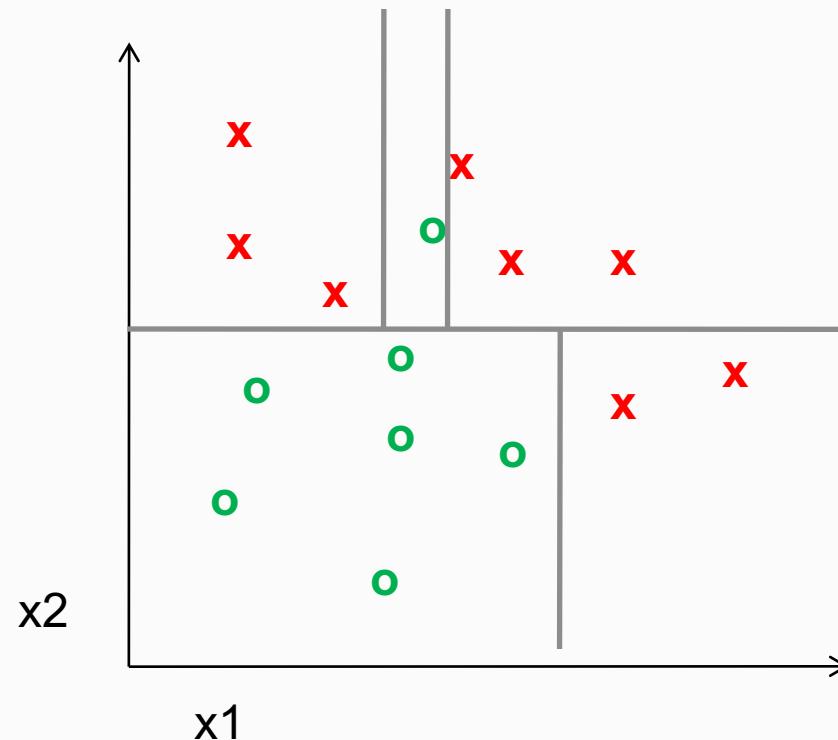
- But what if the dataset is just too hard?



- We can map it to a higher-dimensional space:



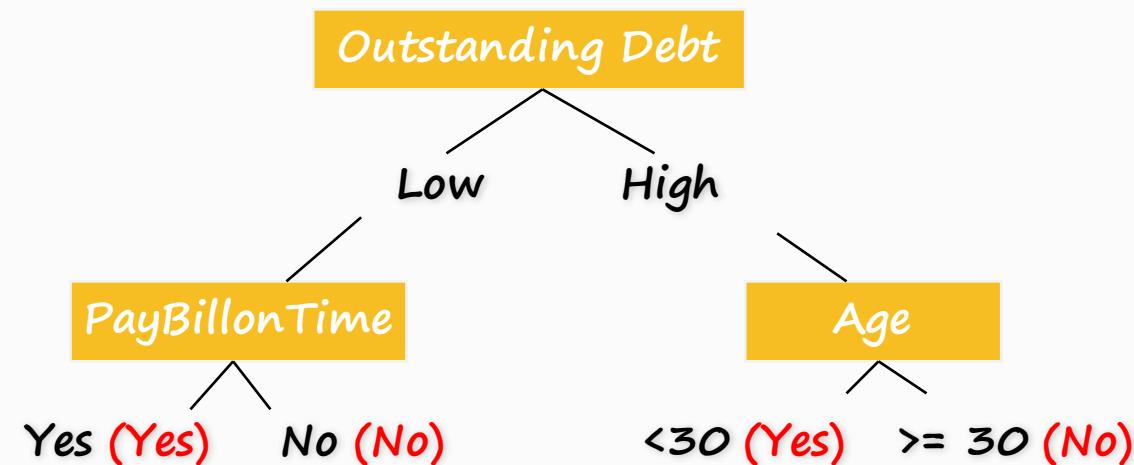
# Classifiers: Decision Trees



# Let's build our first decision tree

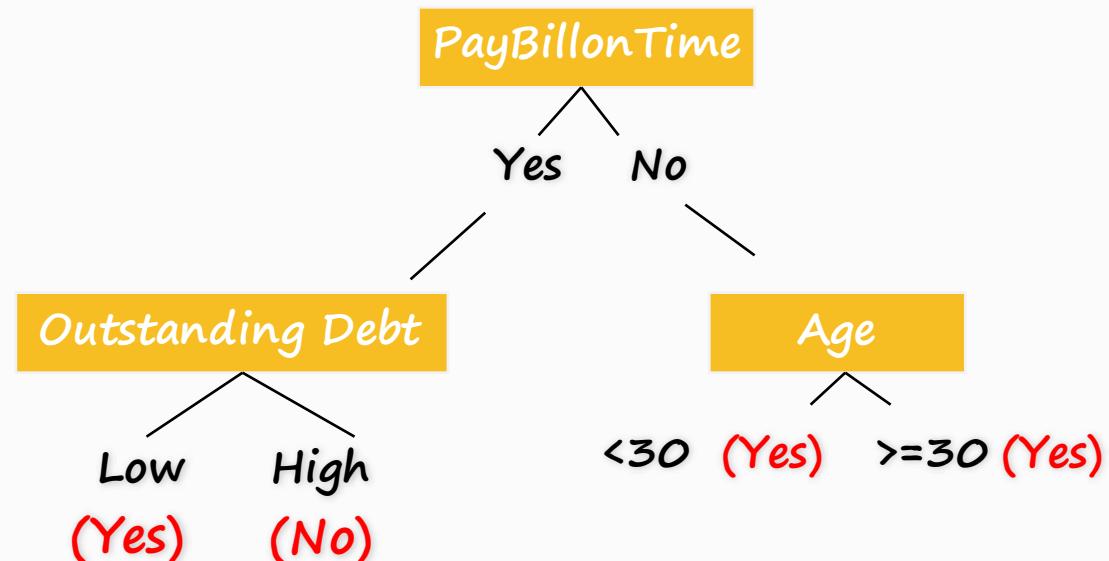
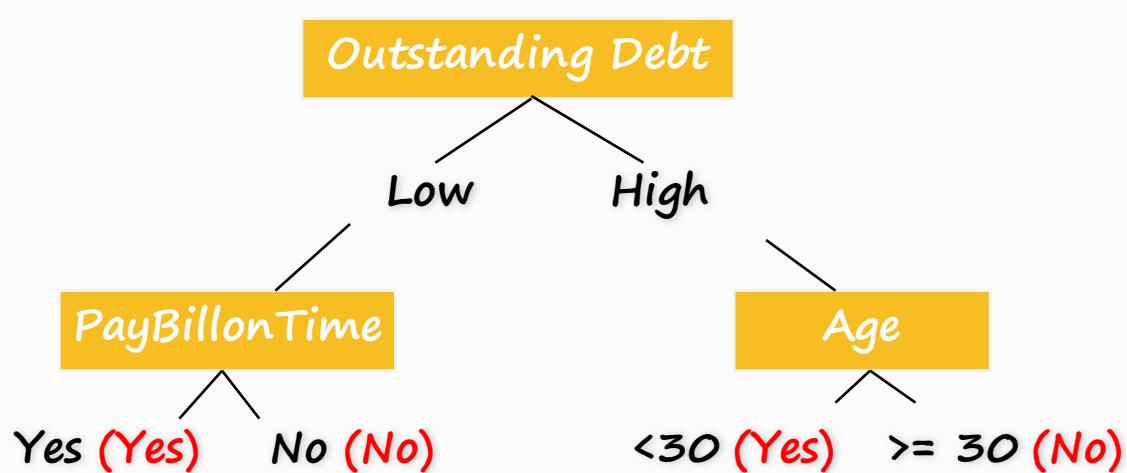


Decision trees classify instances by sorting them from the root of the tree to some leaf nodes (which provides the classification of the instance)



# How do we split?

- Which attribute to be used first?
  - Outstanding Debt?
  - Pay bill on time
  - Age



# Determining the Split Attribute

- Each attribute is assigned a score
- An attribute that maximizes the score during each iteration is chosen as the Split attribute
- Different methods to compute the score  
Example: Information Gain



# Figuring out the Entropy

- Entropy – Measures the impurity/purity of an arbitrary collection of examples
- Binary Classification - Given a collection S, with positive (+) and negative (-) example of a target concept

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

# Example – Computing the Entropy

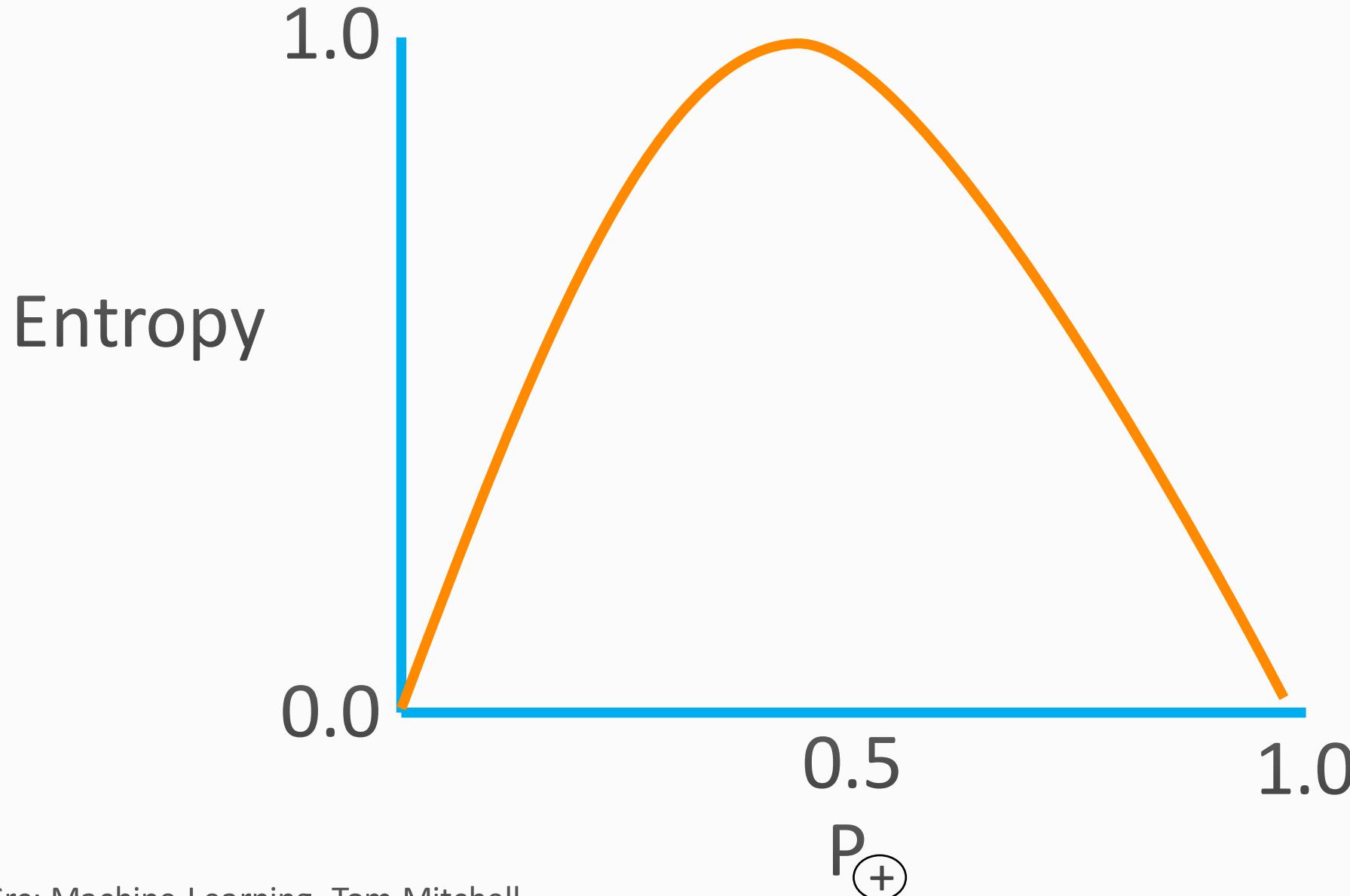
- $S$  is a collection of 14 examples
- 9 are positive examples, 5 are negative examples  
Notation: [9+, 5-]
- Entropy( $S$ )  
 $= -(9/14)\log_2 (9/14) - (5/14)\log_2(5/14)$   
 $= 0.940$

Question:

When will entropy be 1?

When will entropy be 0?

# Entropy vs $P_+$



Src: Machine Learning, Tom Mitchell

# Information Gain

- Information Gain – Expected reduction in entropy if attribute was chosen as the splitting attribute

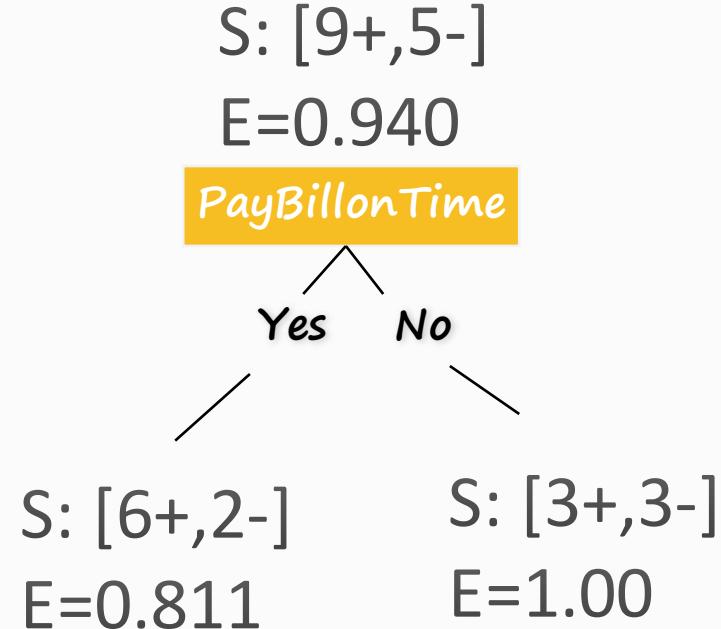
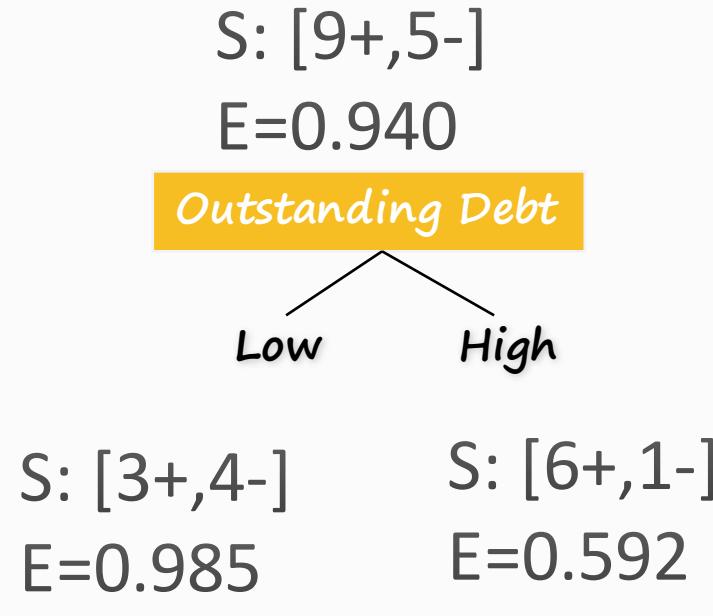
$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

# Example – Computing the Entropy

- $S$  is a collection of 14 examples
- 9 are positive examples, 5 are negative examples  
Notation: [9+, 5-]
- PayBillOnTime = True : 6+ and 2-  
PayBillOnTime = False: 3+ and 3-
- $\text{Values}(\text{PayBillOnTime}) = \text{True, False}$
- $\text{Gain}(S, \text{LoanApproved}) = 0.940$ 
  - $(8/14)\text{Entropy}(\text{PayBillOnTime})$
  - $6/14\text{Entropy}(\text{Does not pay Bill on Time})$
- $= 0.940 - (8/14) 0.811 - (6/14) 1.00$
- $= 0.048$

Src: Machine Learning, Tom Mitchell

# How do we split?



Gain(S, OutstandingDebt) = 0.151

Gain(S, PayBillOnTime) = 0.048

Which provides more gain?

# Model Validation



# Model Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
- **Resample** when there is **few data**
- **Resample** from **the minority** when there is **data skew**

# Evaluating Models

- Infinite data is best, but...
- N (**N=10**) Fold cross validation
  - Create N folds or subsets from the training data (approximately equally distributed with approximately the same number of instances).
  - Build N models, each with a different set of N-1 folds, and evaluate each model on the remaining fold
  - Error estimate is average error over all N models

# What to remember about classifiers

- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)



# Data Science

Deriving Knowledge from Data at Scale

*That's all for tonight...*

