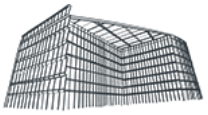
Introduction to Machine Learning:

Nearest Neighbors and Random Forests

Ben Meuleman, Ph.D.
Swiss Center for Affective Sciences
May 8, 2018, Geneva

R Lunch

UNIVERSITÉ
DE GENÈVE



Contents

0. Today's R Lunch

About me…

- 2003 – 2007 Bachelor in Psychology
- 2007 – 2009 Master in Psychology
- **2009 – 2010 Master in Statistical Data Analysis**
- 2011 – 2015 Ph.D. in Psychology
- 2016 – 2018 Postdoc in emotion research

- Master thesis in data analysis was written about "Bayesian methods in artificial neural networks".

- Doctoral dissertation—including two publications—written on the application of machine learning to emotion data.

- Part-time statistical assistant at CISA (2011–2018) and contributor of course materials in artificial neural networks (University of Ghent, 2012–2018).

Background literature – Technical

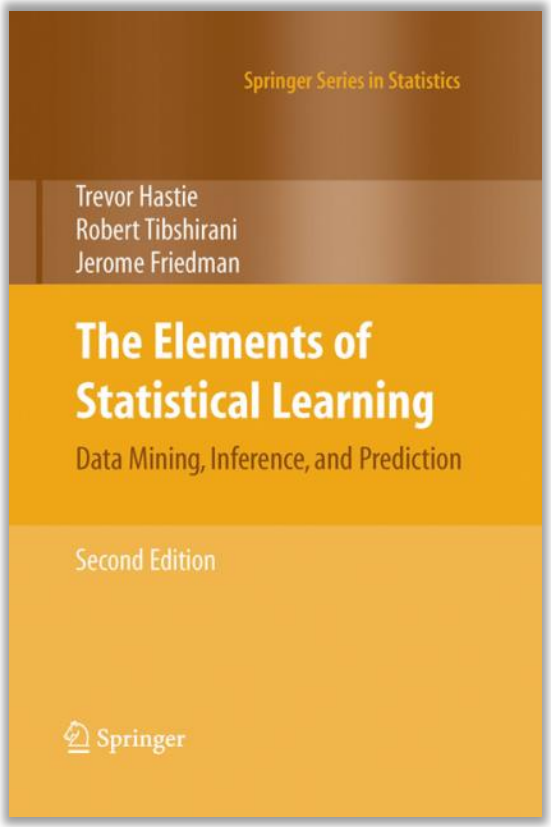
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

- Accessible and comprehensive treatment of supervised and unsupervised methods for machine learning.

- Illustrated with practical data problems and many graphs.

- Other books on machine learning tend to be very mathematical

- Available online for free (via UNIGE)!



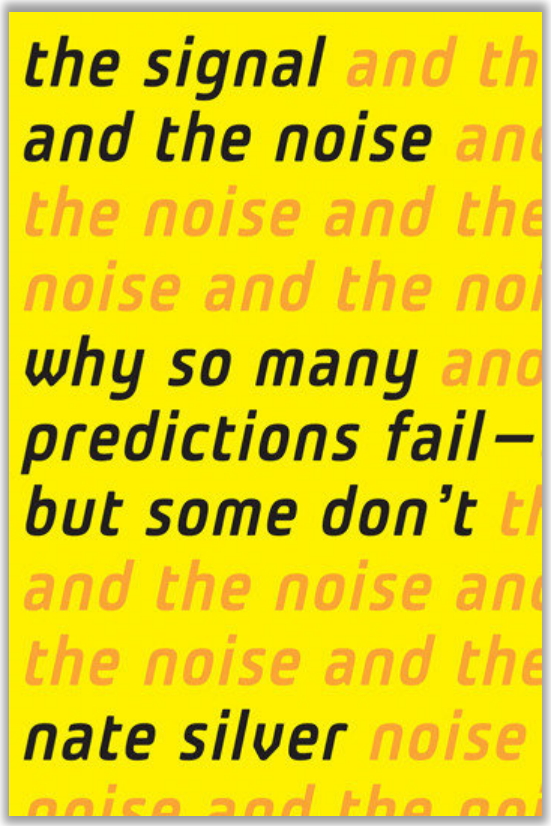
Background literature – Non-technical

- Silver, N. (2012). *The Signal and the Noise : Why So Many Predictions Fail – but Some Don't*. Penguin.

- History of machine learning. Covers recent successes and failures of machine learning.

- Chapters are organised according to specific prediction problems, e.g., weather forecasting, earthquake detection, sports prediction, chess, etc.

- Not mathematical but nevertheless offers an accessible introduction to many technical concepts (e.g., Bayesian inference).



Today's goals

Warning

- This seminar is *not* a proper course on machine learning. There is not enough time here to go into technical details. I will not discuss many important concepts such as model selection, feature selection, validation, etc.

- This slideshow was based on a 2-part, 5-hour workshop. If interested, you can obtain the full slides upon request (ben.meuleman@unige.ch).

Today's goals

- Today I will introduce to you basic concepts about machine learning, as a kind of crash course. At the end of this presentation you should be able to understand the following:

  1. That machine learning is just another type of statistical modelling
  2. That machine learning is connected to linear regression
  3. That there exist models with a radically different approach from linear regression to analyzing data
  4. The concepts of overfitting, decision boundaries, ensemble learning, and bootstrap aggregation.

R – Pack lunch

- The R part of this R lunch will be somewhat light, as I prefer to focus on explaining to you the important concepts, rather than the somewhat boring details of how to run a model in R.

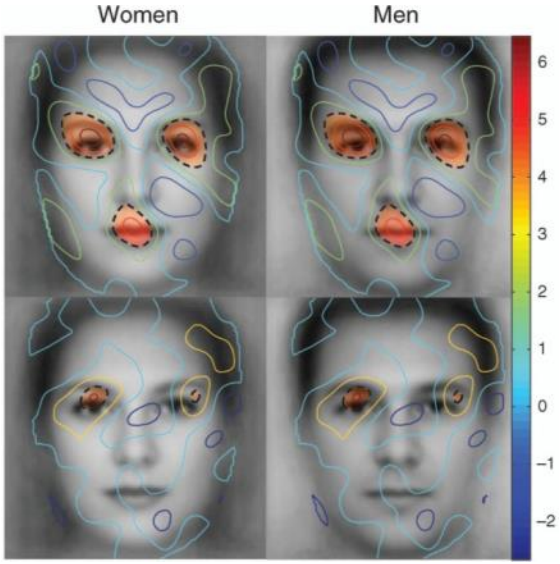
- In fact, running machine learning models is often surprisingly simple in R. With the right package loaded, the code often takes up less lines than a conventional linear regression analysis would…

- However, I will provide some code as examples and reference the packages that you need to install to run the models I present today.

- Don't forget to check the R platform's fabulous task views page on available packages: https://cran.r-project.org/web/views/MachineLearning.html

1. Introduction

The hype…



Face recognition

Speech recognition

Credit card fraud detection

The hype…

Google

Web page search ranking

Self-driving vehicles

Artistic style recognition/fusion
https://deepart.io

DeepArt

DeepArt

In the style of Vincent Van Gogh

DeepArt

In the style of Juan Gris

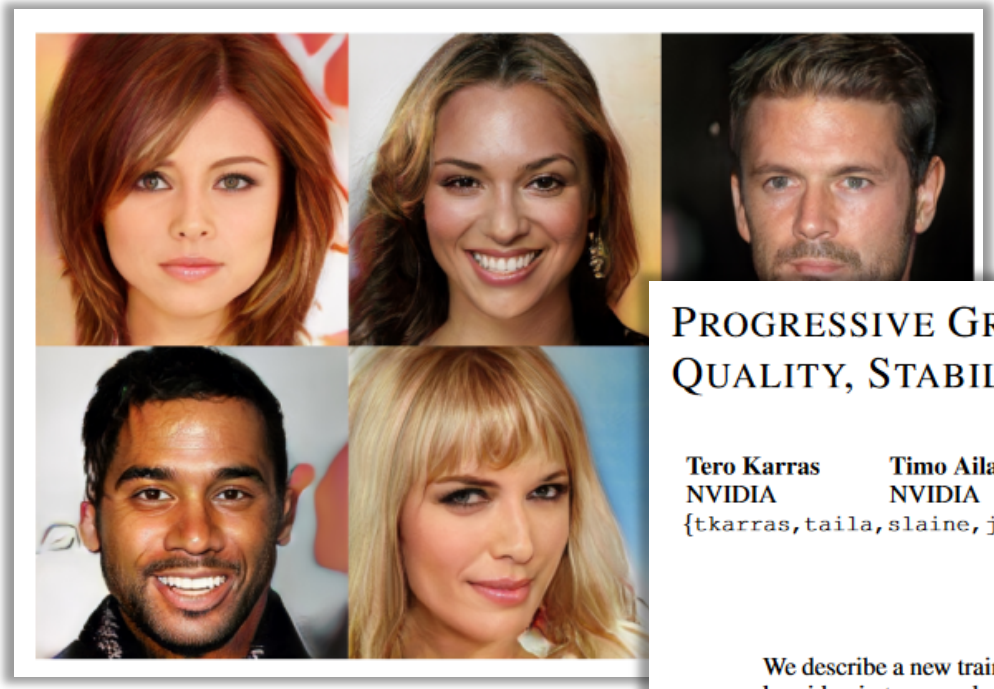
DeepArt

In the style of a toddler

In the news

Can you tell what is wrong with these faces…?

In the news



PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Tero Karras NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA
Jaakko Lehtinen NVIDIA and Aalto University
{tkarras,taila,slaine,jlehtinen}@nvidia.com

ABSTRACT

We describe a new training methodology for generative adversarial networks. The key idea is to grow both the generator and discriminator progressively: starting from a low resolution, we add new layers that model increasingly fine details as training progresses. This both speeds the training up and greatly stabilizes it, allowing us to produce images of unprecedented quality, e.g., CELEBA images at 1024^2. We also propose a simple way to increase the variation in generated images, and achieve a record inception score of 8.80 in unsupervised CIFAR10. Additionally, we describe several implementation details that are important for discouraging unhealthy competition between the generator and discriminator. Finally, we suggest a new metric for evaluating GAN results, both in terms of image quality and variation. As an additional contribution, we construct a higher-quality version of the CELEBA dataset.

https://sploid.gizmodo.com/watching-this-neural-network-render-truly-photorealisti-1819957128

In the news

INDEPENDENT

News Voices Culture Lifestyle Tech Sport Daily Edition

Lifestyle › Tech › News

Google AlphaGo computer beats professional at 'world's most complex board game' Go

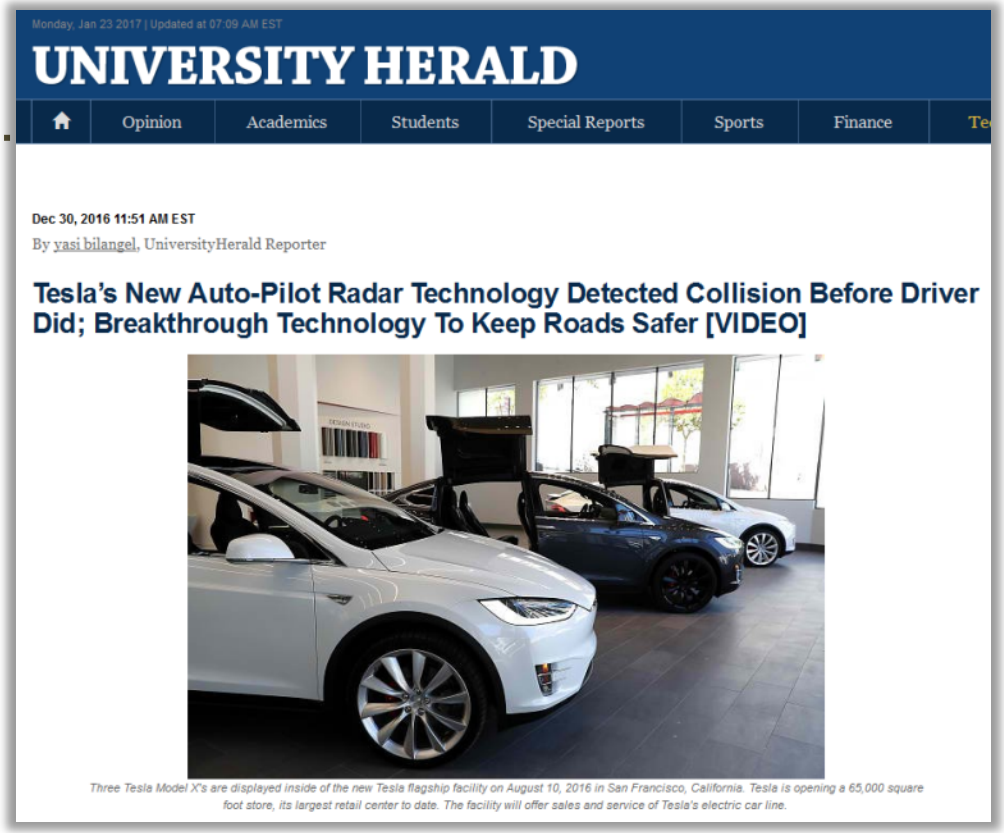
Milestone in AI research likened to defeat of world chess champion Garry Kasparov in 1997 by IBM's Deep Blue computer

Steve Connor Science Editor | @SteveAConnor | Wednesday 27 January 2016 | 18 comments

500 shares

Like Click to follow The Independent Tech

More complex than chess: the Chinese board game Go *Wikimedia/Creative Commons*



Monday, Jan 23 2017 | Updated at 07:09 AM EST

UNIVERSITY HERALD

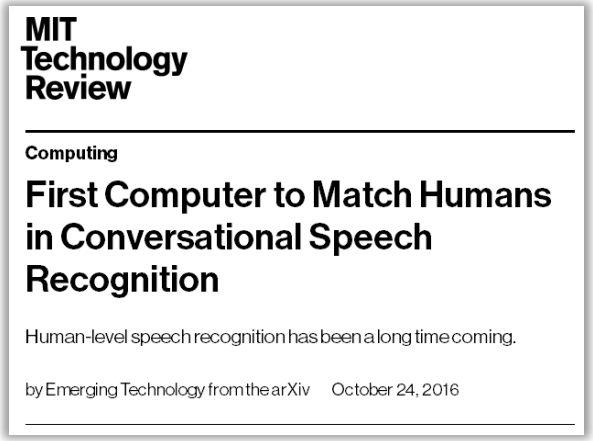
Opinion Academics Students Special Reports Sports Finance Te

Dec 30, 2016 11:51 AM EST

By yasi bilangel, UniversityHerald Reporter

Tesla's New Auto-Pilot Radar Technology Detected Collision Before Driver Did; Breakthrough Technology To Keep Roads Safer [VIDEO]

Three Tesla Model X's are displayed inside of the new Tesla flagship facility on August 10, 2016 in San Francisco, California. Tesla is opening a 65,000 square foot store, its largest retail center to date. The facility will offer sales and service of Tesla's electric car line.



MIT Technology Review

Computing

First Computer to Match Humans in Conversational Speech Recognition

Human-level speech recognition has been a long time coming.

by Emerging Technology from the arXiv October 24, 2016

Machine learning as statistical data analysis

- In the media, machine learning is often presented as a kind of *artificial intelligence*, implemented in robots capable of imitating human intelligence. This is a slightly sensationalized version of machine learning.

- What is important to understand is that machine learning is just an extended type of statistical modelling, with as a goal extracting complex patterns from observed data sets.

- Sometimes data patterns are extracted without a specific dependent variable (=unsupervised machine learning, e.g., clustering), but most of the time data patterns are extracted with the intention of predicting well a specific dependent variable (=supervised machine learning, e.g., linear regression).

- Machine learning is also known under various synonyms, such as artificial intelligence, pattern recognition, and data mining.

2. Machine learning and linear regression

Linear regression

- When you analyze data, you are very likely using linear regression as the underlying model, which includes all of the following special cases:

• One-way regression	One continuous independent
• Multiple regression	Multiple continuous independents
• Independent samples *t*-test	One categorical independent (2 levels)
• One-way ANOVA	One categorical independent (>2 levels)
• Multi-way ANOVA	Multiple categorical independents
• ANCOVA	Mix of categorical and continues independents
• Pearson correlation	One continuous independent. Dependent and independent standardized.
• One-sample *t*-test	Intercept-only as independent.
• Paired *t*-test	Difference score as dependent. Intercept-only as independent.

Linear regression and hypothesis testing

- In linear regression, one tries to predict a dependent variable (Y), given a number of independent variables (the Xs), which are also called predictors.

- The predictors can be observational measurements (e.g., personality scores, age, gender) or experimentally manipulated factors (e.g., drug dosage, type of therapy).

- The goal of analyses in the social sciences is largely to find out which predictors have a significant/relevant relationship to the dependent variable. This goal is achieved by statistical inference using hypothesis testing.

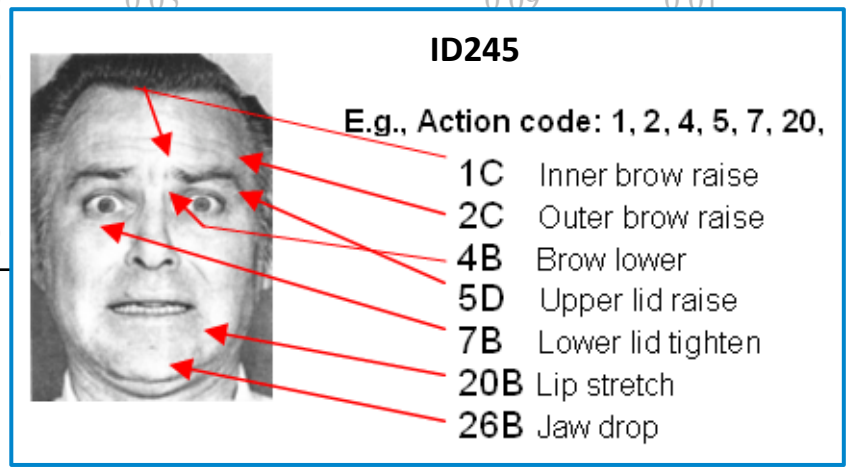
- This comes down to testing the probability of observing a given effect (or larger) under the assumption that there is actually no effect (≈ null hypothesis). This probability is captured by the familiar *p*-value. Conventionally, an effect is considered significant when its $p < 0.05$.

A regression data set

PhotoID	AU1	AU2	AU4	AU7	AU12	…	AU25	Anger intensity
ID245	0.00	0.05	0.88	0.61	0.00	…	0.00	0.64
ID17	0.00	0.00	0.25	0.27	0.00	…	0.00	0.33
ID301	0.00	0.00	0.18	0.00	0.00	…	0.00	0.00
ID70	0.56	0.64	0.00	0.00	0.00	…	0.17	0.00
ID68	0.23	0.79	0.30	0.00	0.03	…	0.09	0.01
ID143	0.00	0.10	0.89	0.00	0.01	…	0.00	0.74
…	…	…	…	…	…	…	…	…
ID199	0.97	0.90	0.00	0.00	0.31	…	0.94	0.00

A regression data set

PhotoID	AU1	AU2	AU4	AU7	AU12	…	AU25	Anger intensity
ID245	0.00	0.05	0.88	0.61	0.00	…	0.00	0.64
ID17	0.00	0.00	0.25	0.27	0.00	…	0.00	0.33
ID301	0.00	0.00	0.18	0.00	0.00	…	0.00	0.00
ID70	0.56	0.64	0.00	0.00	0.00	…	0.17	0.00
ID68	0.23	0.79	0.30	0.00	0.03		0.09	0.01
ID143	0.00	0.10	0.89	0.00				
…	…	…	…	…				
ID199	0.97	0.90	0.00	0.00				



A regression data set

PhotoID	AU1	AU2	AU4	AU7	AU12	…	AU25	Anger intensity
ID245	0.00	0.05	0.88	0.61	0.00	…	0.00	0.64
ID17	0.00	0.00	0.25	0.27	0.00	…	0.00	0.33
ID301	0.00	0.00	0.18	0.00	0.00	…	0.00	0.00
ID70	0.56	0.64	0.00	0.00	0.00	…	0.17	0.00
ID68	0.23	0.79	0.30	0.00	0.03	…	0.09	0.01
ID143	0.00	0.10	0.89	0.00	0.01	…	0.00	0.74
…	…	…	…	…	…	…	…	…
ID199	0.97	0.90	0.00	0.00	0.31	…	0.94	0.00

Independents/predictors
*X*s

Dependent
Y

Hypothesis testing on regression coefficients

PhotoID	AU1	AU2	AU4	AU7	AU12	…	AU25	Anger intensity
ID245	0.00	0.05	0.88	0.61	0.00	…	0.00	0.64
ID17	0.00	0.00	0.25	0.27	0.00	…	0.00	0.33
ID301	0.00	0.00	0.18	0.00	0.00	…	0.00	0.00
ID70	0.56	0.64	0.00	0.00	0.00	…	0.17	0.00
ID68	0.23	0.79	0.30	0.00	0.03	…	0.09	0.01
ID143	0.00	0.10	0.89	0.00	0.01	…	0.00	0.74
…	…	…	…	…	…	…	…	…
ID199	0.97	0.90	0.00	0.00	0.31	…	0.94	0.00
Coef.	β_{AU1}	β_{AU2}	β_{AU4}	β_{AU7}	β_{AU12}	…	β_{AU25}	
t	0.02	0.08	7.56	3.01	2.22	...	0.00	
p	0.46	0.64	0.00	0.01	0.07	…	0.13	

$R^2 = 0.57$

Hypothesis testing on regression coefficients

PhotoID	AU1	AU2	**AU4**	**AU7**	AU12	…	AU25	**Anger intensity**
ID245	0.00	0.05	0.88	0.61	0.00	…	0.00	0.64
ID17	0.00	0.00	0.25	0.27	0.00	…	0.00	0.33
ID301	0.00	0.00	0.18	0.00	0.00	…	0.00	0.00
ID70	0.56	0.64	0.00	0.00	0.00	…	0.17	0.00
ID68	0.23	0.79	0.30	0.00	0.03	…	0.09	0.01
ID143	0.00	0.10	0.89	0.00	0.01	…	0.00	0.74
…	…	…	…	…	…	…	…	…
ID199	0.97	0.90	0.00	0.00	0.31	…	0.94	0.00
Coef.	β_{AU1}	β_{AU2}	$\boldsymbol{\beta_{AU4}}$	$\boldsymbol{\beta_{AU7}}$	β_{AU12}	…	β_{AU25}	
t	0.02	0.08	7.56	3.01	2.22	…	0.00	
p	0.46	0.64	0.00	0.01	0.07	…	0.13	

$R^2 = 0.57$

Machine learning vs classic regression

- Linear regression is itself a statistical "learner" and can be considered a machine learning model. Moreover, many "advanced" models for machine learning can be rewritten or expressed in ordinary regression language.

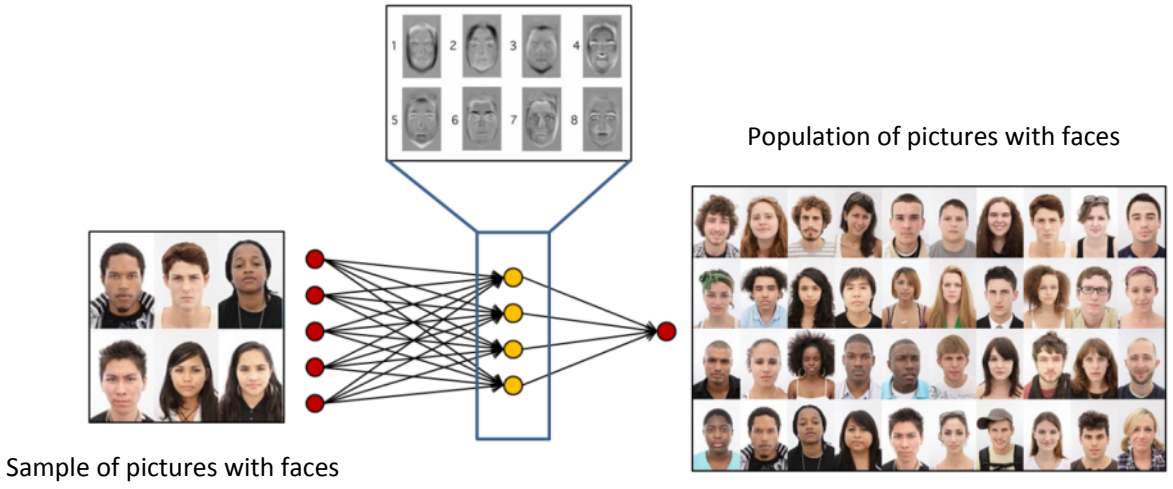
- In linear regression we often evaluate **(a)** the relevance/significance of predictors with *p*-values and **(b)** the model's fit to the dependent data with an R^2 value. These two objectives have more general counterparts in machine learning.

Goal	Classic regression	Machine learning
Relevance evaluation	Hypothesis testing	Feature selection
Model fit	Proportion of explained variance (R^2)	Predictive strength (e.g., cross-validated R^2)

Machine learning vs classic regression

- In contrast to classic regression (especially in social sciences), feature selection in machine learning is often a *secondary interest* to achieving a model that fit/predicts the dependent variable well.

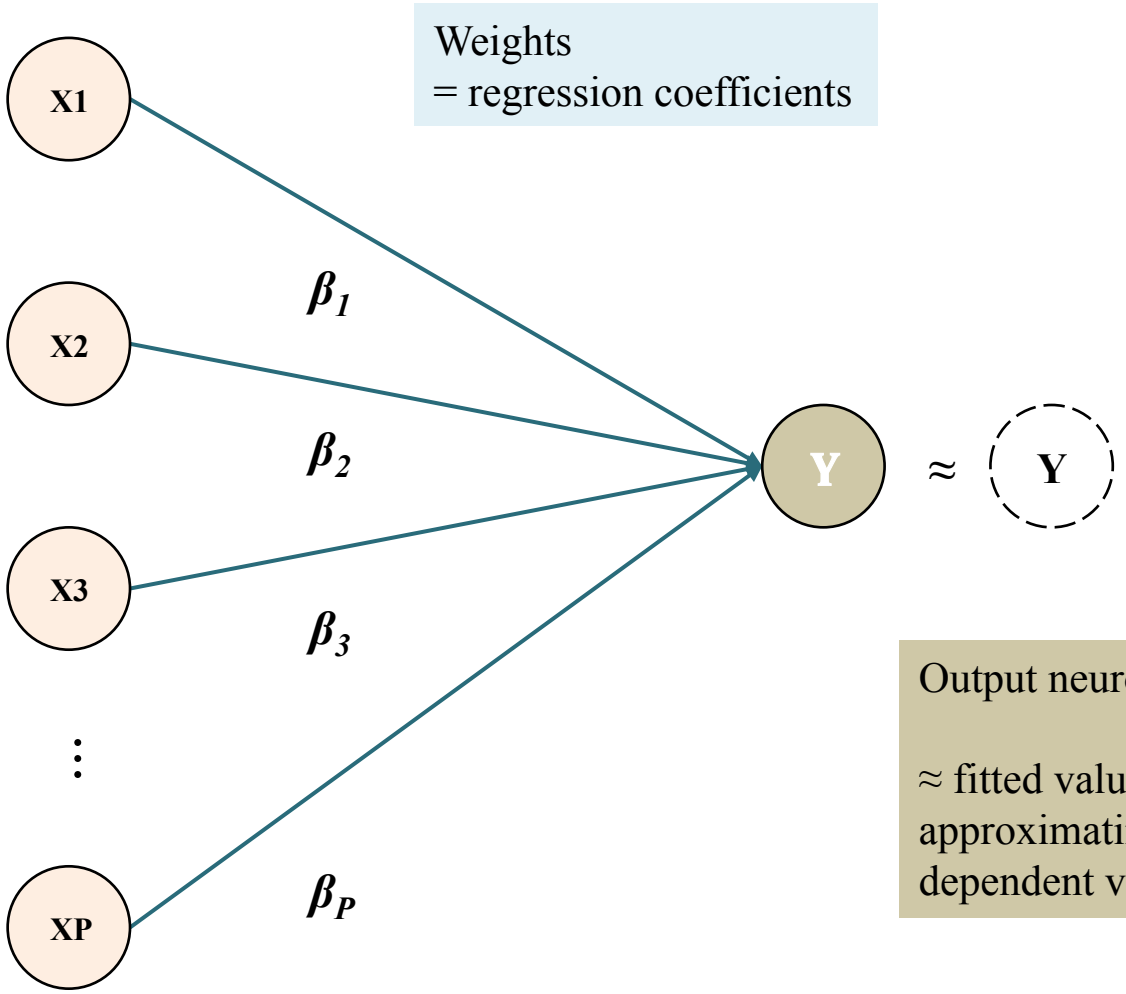
- In machine learning, model fit is primarily evaluated in terms of predictive strength —or generalisation capacity. That is, we want a model to be accurate not just on the fitted data, but also new data that the model has not yet seen (see later...).



More terminological connections…

Concept	Classic regression term		Machine learning term
Data	Observations		Cases / Patterns
***Y* variables**	Dependents		Responses / Targets / Outcomes
***X* variables**	Independents / Predictors / Regressors		Features / Inputs
Parameters	Coefficients		Weights
Parameter estimation	Least squares estimation		Training / Learning
Model predictions	Fitted values		Predicted values
Relevance evaluation of X variables	Hypothesis testing	≈	Feature selection
Model fit	Proportion of explained variance (R^2)		Predictive strength (e.g., cross-validated R^2)

Regression as a "neural network"



Machine learning as statistical data analysis

- All statistical models—in supervised learning—ultimately serve the same purpose. We want to find out if there is a systematic relation between a dependent variable (a response) and a set of independent variables (features).

- We estimate these relations with a model, based on observed data . This model can be simple (e.g., basic regression) or as complex (e.g., SVM) as you choose.

- These basic principles make clear that machine learning is not about esoteric artificial intelligence. It is simply a kind of statistical modelling.

Examples of modelling problems

Response variable / Dependent (DV)	Features / Independents (IVs)
• Is a credit card purchase fraudulent, yes or no?	Person's purchasing history (type shop, type product, country of purchase, time of day, time since last purchase,…)
• Academic performance (e.g., number of publications per year)	Person's academic history, personality trait scores, social scores, misc demographic characteristics
• Does a photographed face contain a smile, yes or no?	Images of faces: potentially decomposed into higher level features such as facial action unit activity
• Next speed and direction for a self-driving car	Current and past speed and direction, detected features of the road ahead, detected interfering activity (pedestrians, other cars), etc.
• Is an e-mail spam or not spam?	Message title, contents, sender, punctuation, spelling, grammar, etc.
• Should a patient be diagnosed with a particular illness, affliction, or disorder, yes or no?	Medical images, observational data (e.g., test performance), demographical characeristics.

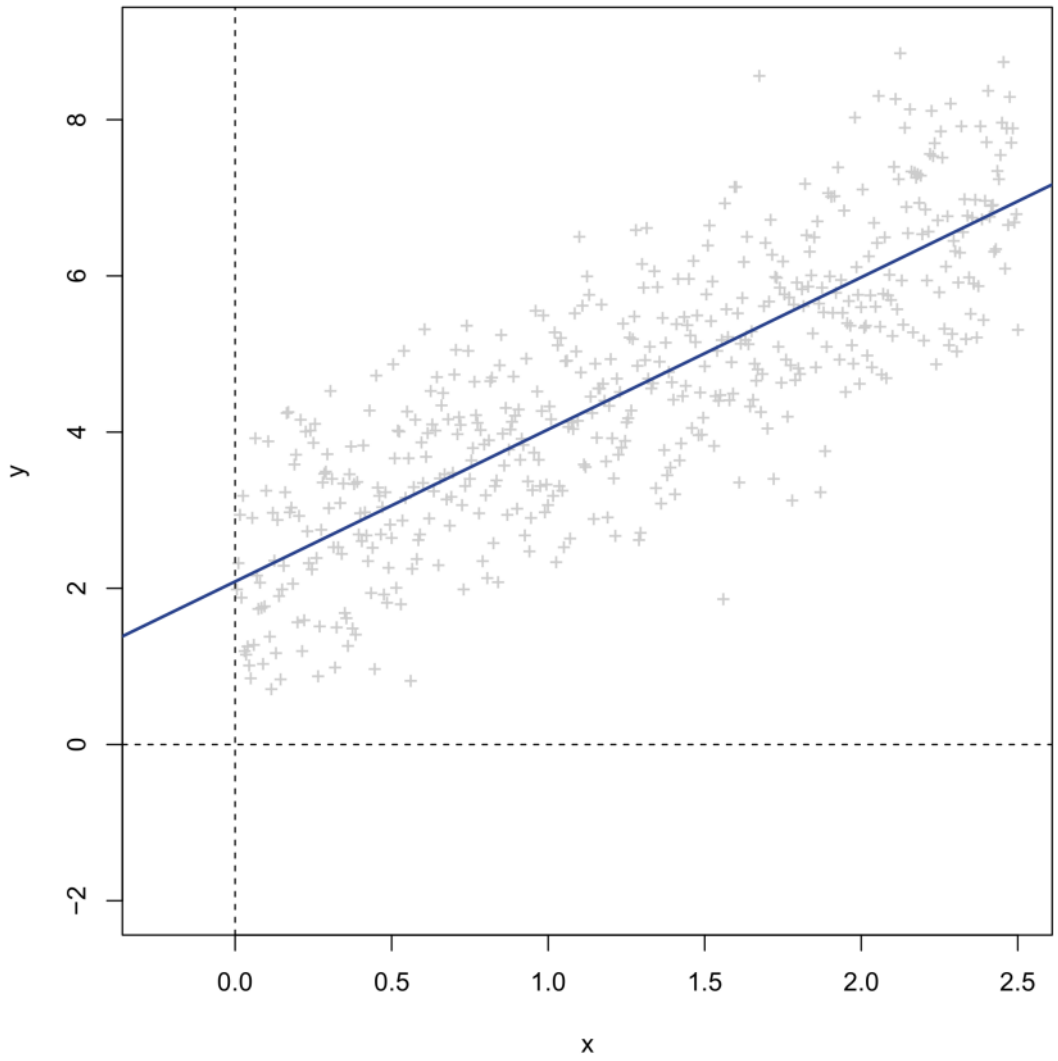
* Note that these are all observational data problems. Machine learning is typically not suited to the analysis of experimental designs!

Reasons why social scientists are confused

- For social scientists machine learning can appear confusing. This is because you have to get used to the following differences:

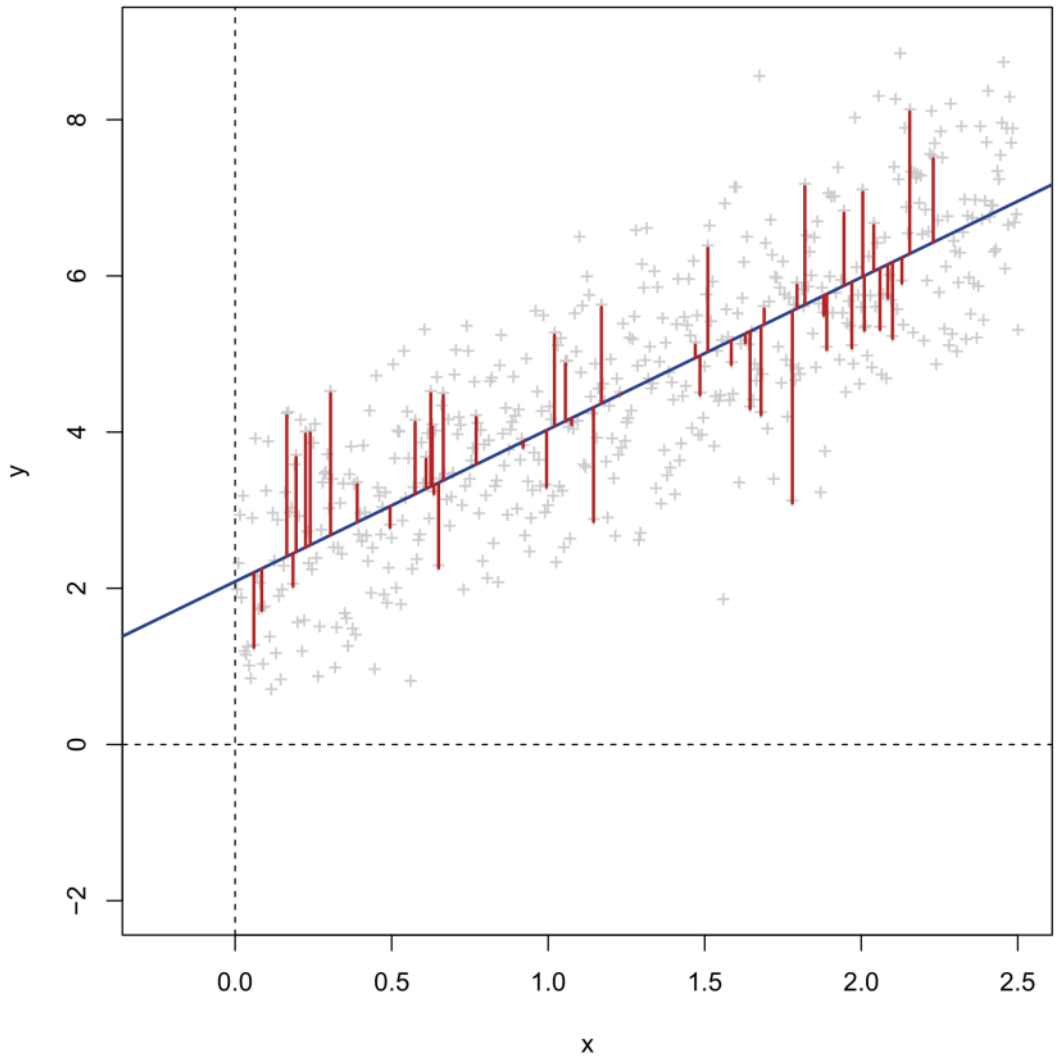
1. In machine learning evaluating variable relevance is typically not of interest.
2. Models that implement relevance assessment never use criteria such as *p*-values.
3. Some models have no parameters in a conventional sense (e.g., weights) that can be inspected.
4. Some models are so complex that inspecting the parameters would not yield any insight at all (so-called black box models, e.g., SVM, neural networks).
5. Social scientists are not used to predicting *new* data with a model.
6. Machine learning is typically not concerned with checking (distributional) assumptions.
7. Some of the R code is surprisingly/disappointingly simple.

Linear regression recap



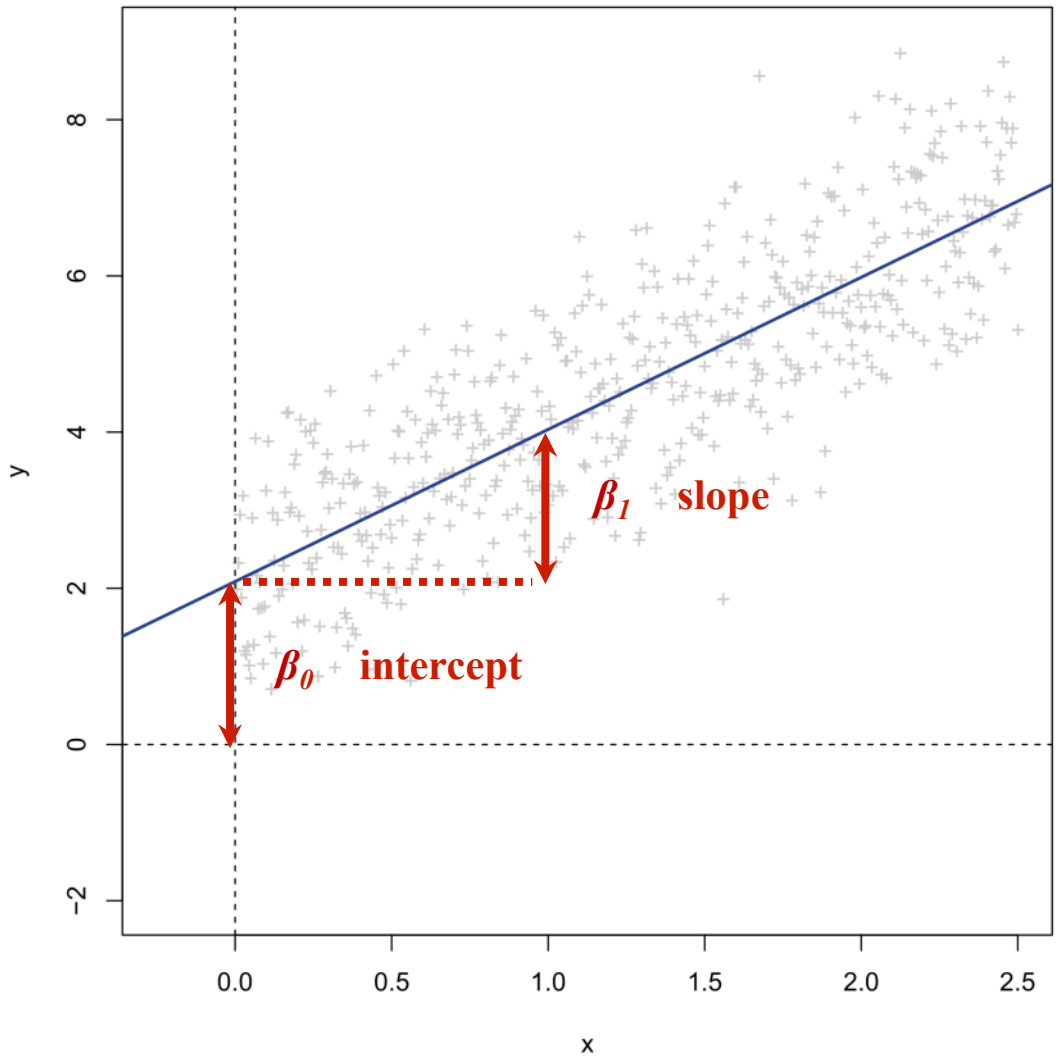
- Continuous dependent Y
- Continuous independent X
- Linear regression of Y on X
- = Linear approximation of Y using X

Linear regression recap



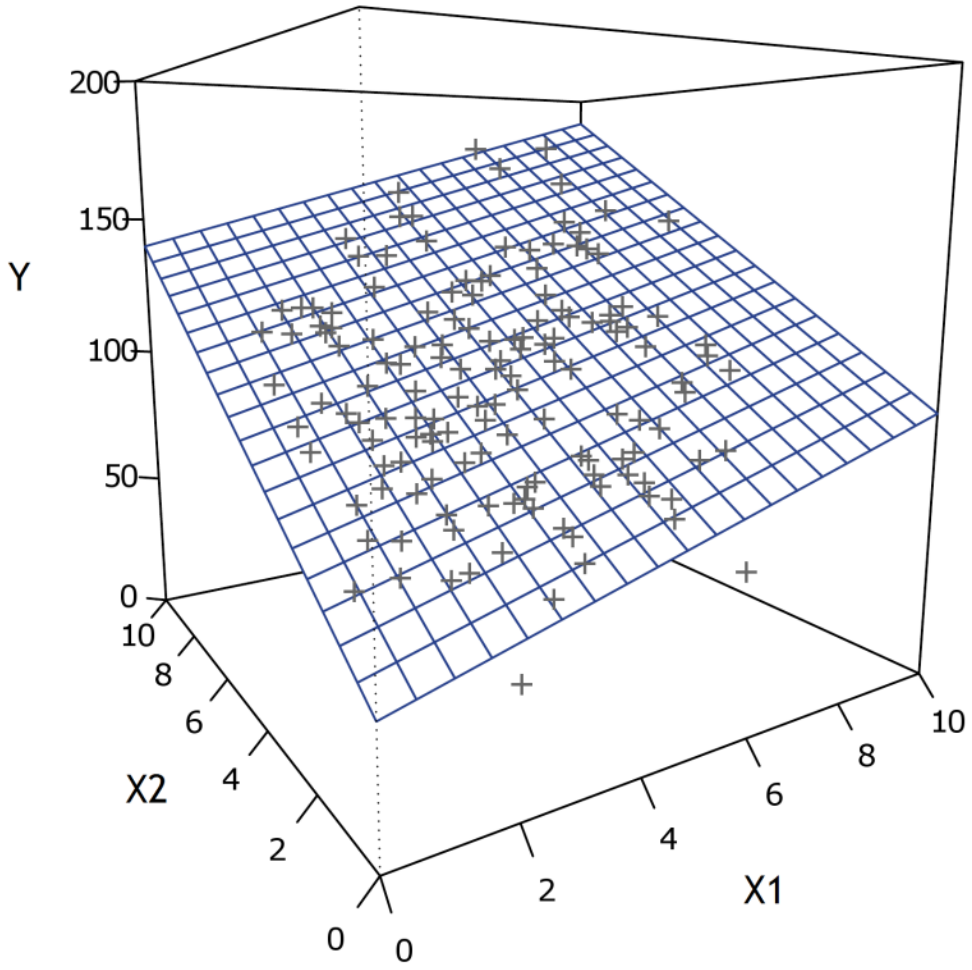
- Regression line minimizes the sum of the squared distances (SSE) of the observations to the line
- These distances are called residuals for the fitted model
- $\varepsilon_i = Y_i^{fit} - Y_i^{obs}$
- SSE is an amount of error that we accept, around our regression line

Linear regression recap



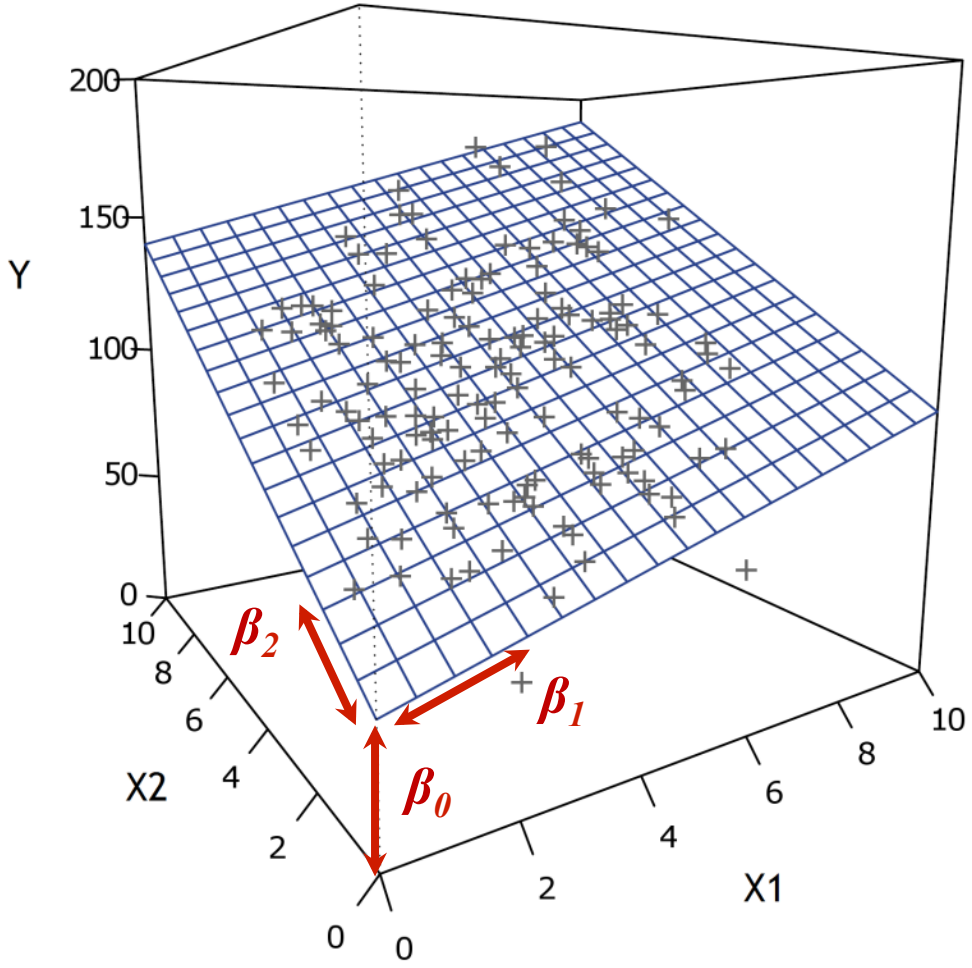
- $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$

Linear regression recap



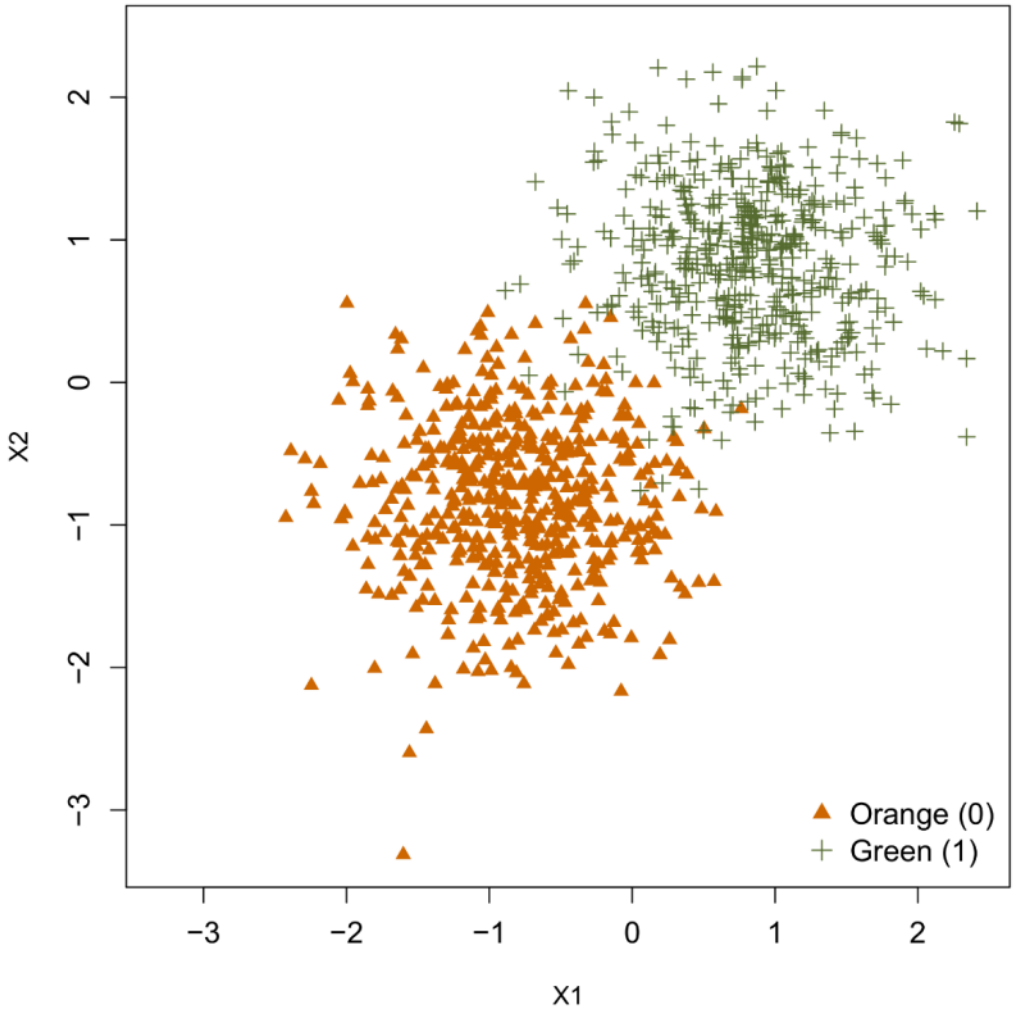
- $Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \varepsilon_i$

Linear regression recap



- $Y_i = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip} + \varepsilon_i$

Binary logistic (linear) regression



- Categorical dependent *Y*
  - Orange = 0
  - Green = 1
- Two continuous independents *X1* and *X2*
- Predicting the class of Y (0 or 1) given the predictors X1 and X2 can be solved by binary logistic regression.

Binary logistic (linear) regression

- Binary logistic regression—often abbreviated to logistic regression—models a two-class categorical dependent, whose levels are recoded to numerical values 0 and 1. Logistic regression predicts the probability of class 1 versus class 0, given a number of predictor variables. Model equation has the form:

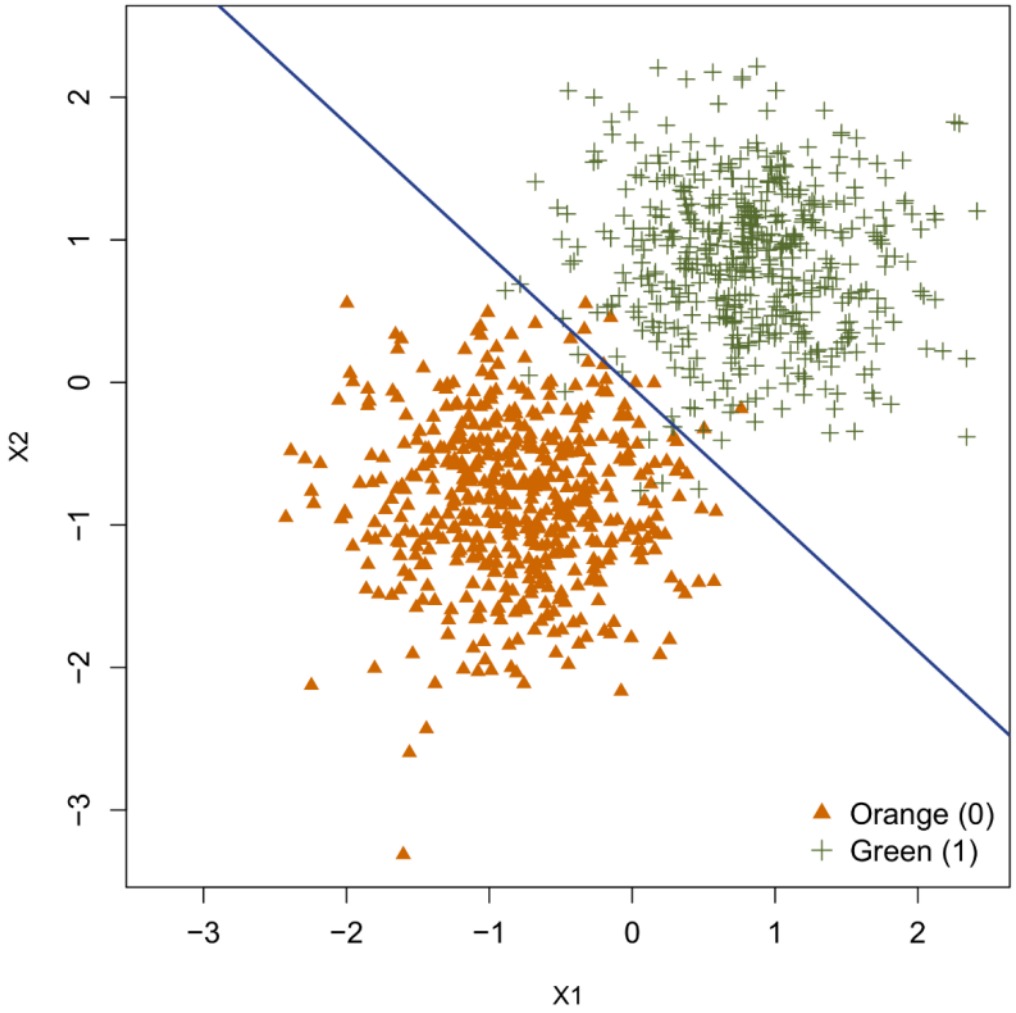
$$\text{logit}(p_i) = \beta_0 + \beta_1 X_{i1} + \ldots + \beta_p X_{ip}$$

- Where p_i is the probability of class 1, and "logit" refers to the logit link function for generalized linear models:

$$\log p\downarrow i \,/ 1 - p\downarrow i$$

- Regression coefficients are estimated such that they maximize the linear separation of the two response classes (i.e., minimize the overlap between the classes).

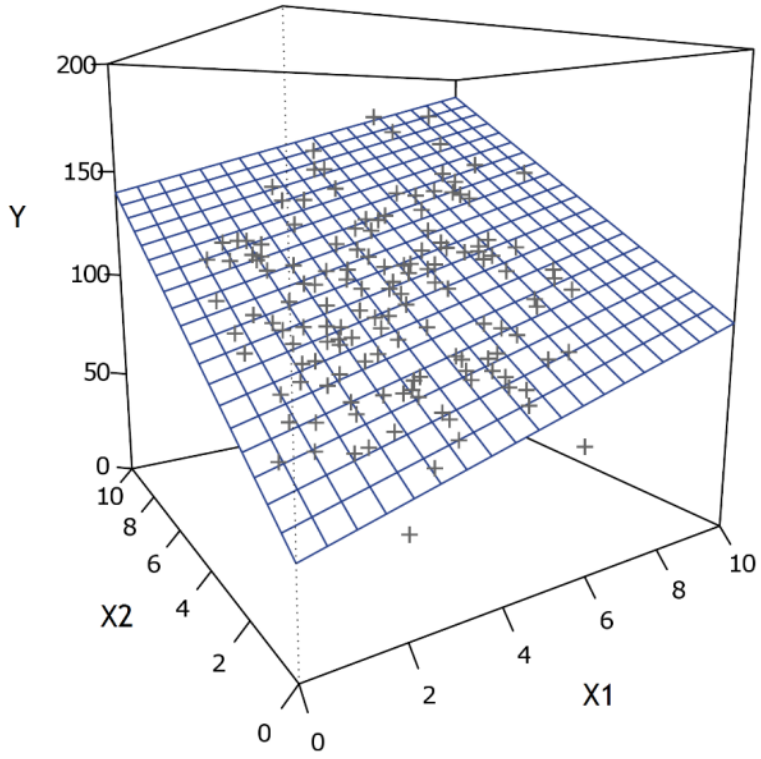
Binary logistic (linear) regression



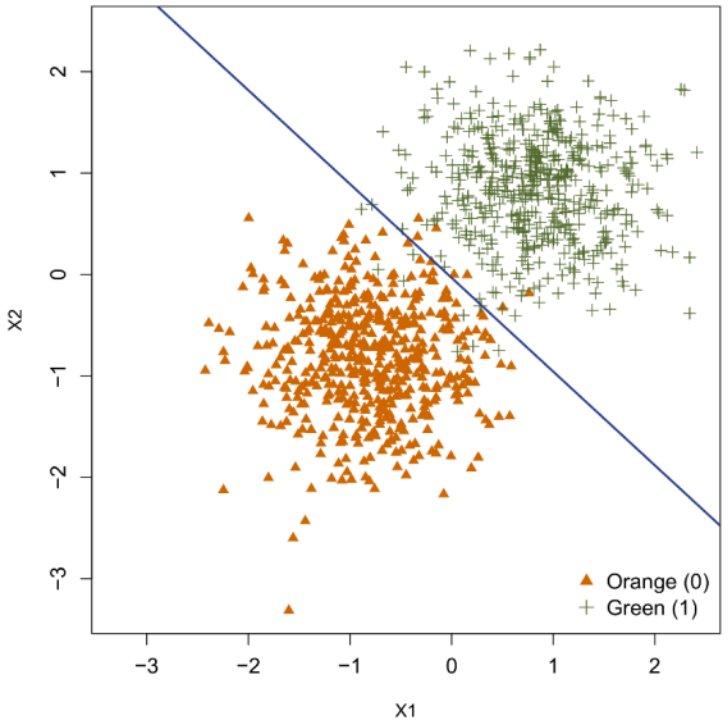
- Logistic regression equation leads to a linear decision boundary between the two response classes
- On the green side of the boundary, the model predicts prob(green) > 0.5
- On the orange side of the boundary, the model predicts prob(green) < 0.5
- Boundary can thus be used to *classify* data points into one class or another

Linearity summary

Continous dependent
"regression model"



Categorical dependent
"classification model"

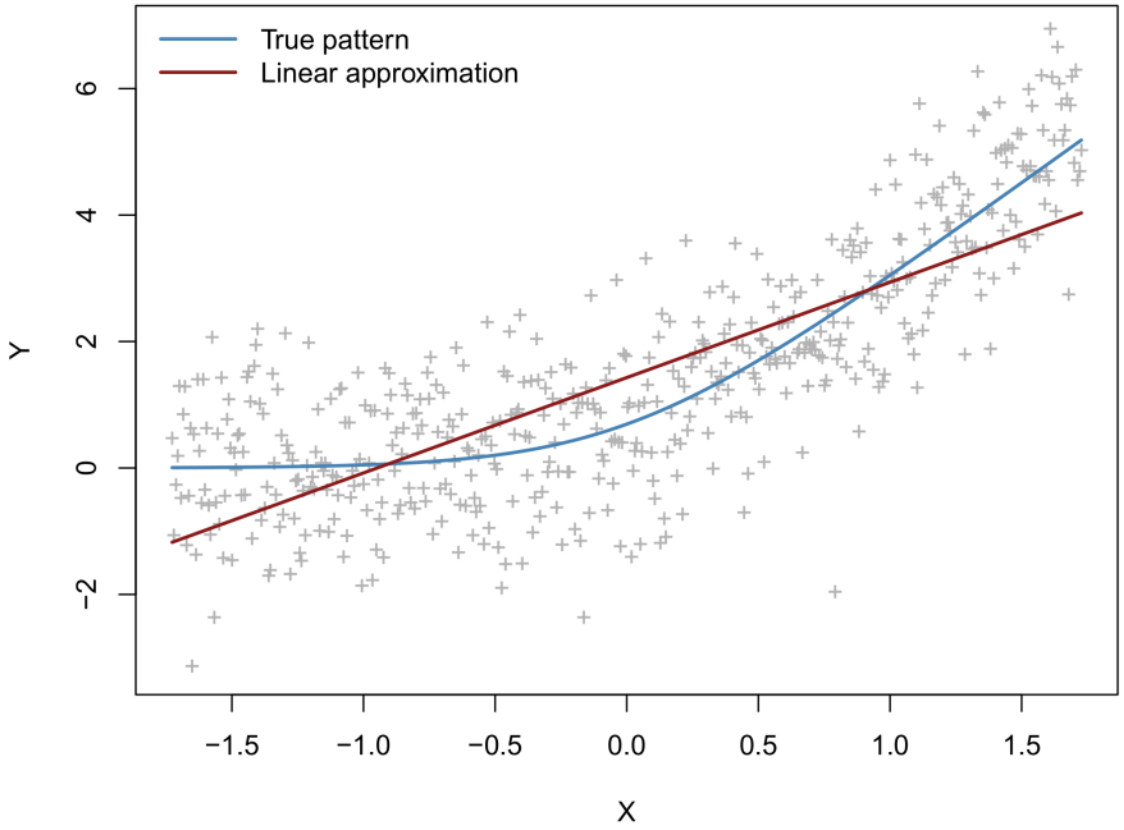


- Basic linear models find an optimal linear equation that minimizes SSE (for continuous dependents) or maximally separates the response classes (for categorical dependents).

3. Needs for machine learning

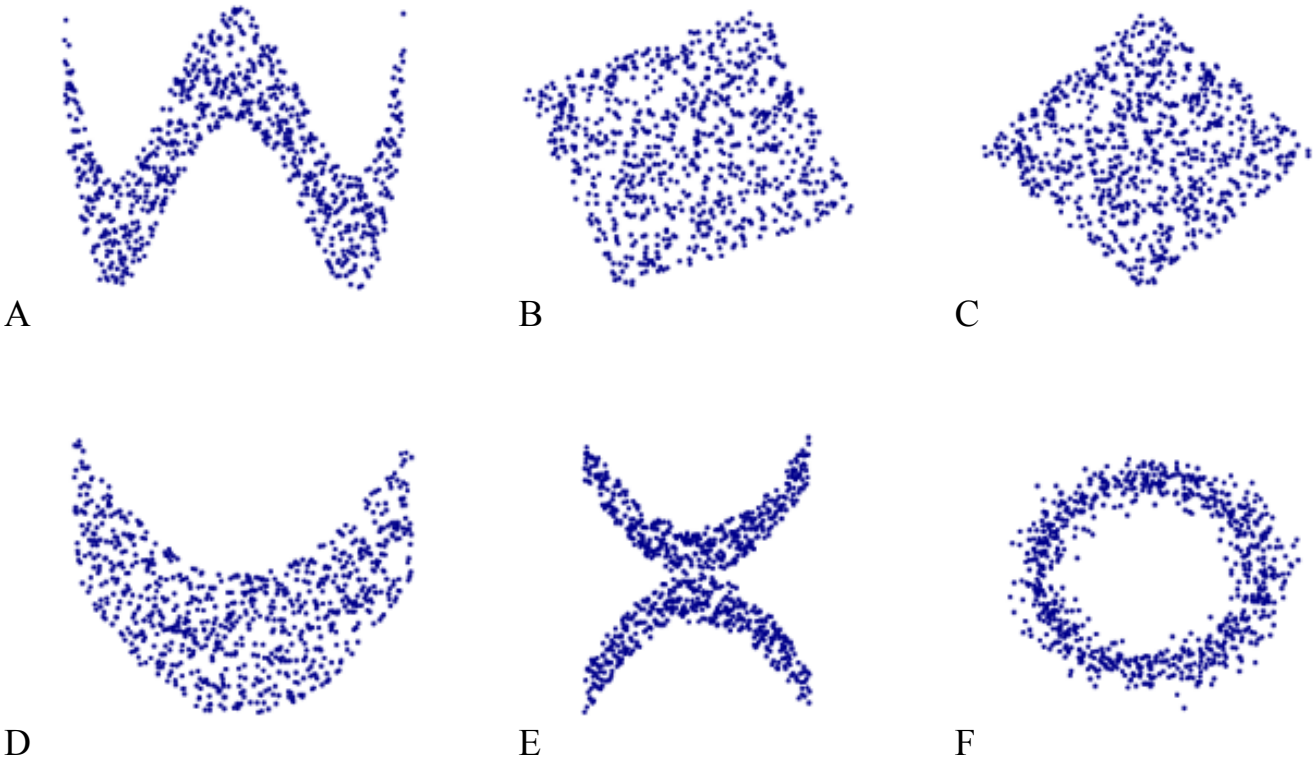
Nonlinearity

- Basic linear models perform remarkably well on a large number of pattern recognition problems, even as approximations to nonlinear patterns, e.g.:



Nonlinearity

- However, nonlinear patterns can also be complex, including curvilinear associations or interactions between multiple features simultaneously. Some relations can be non-functional.

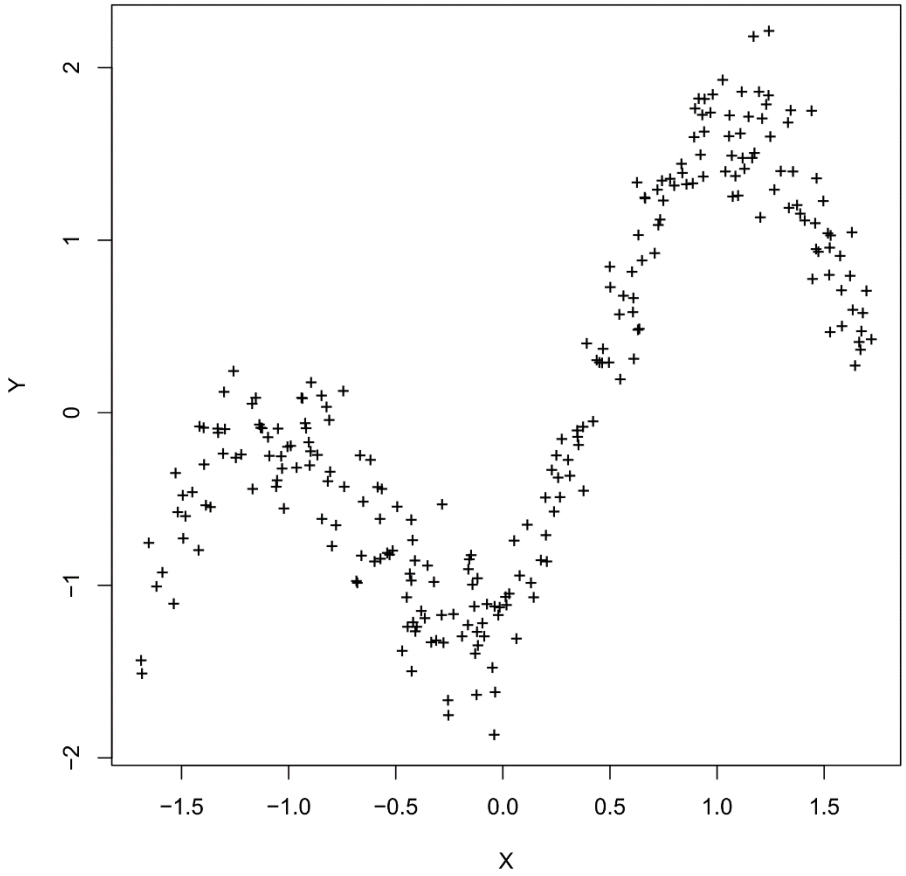


Nonlinearity

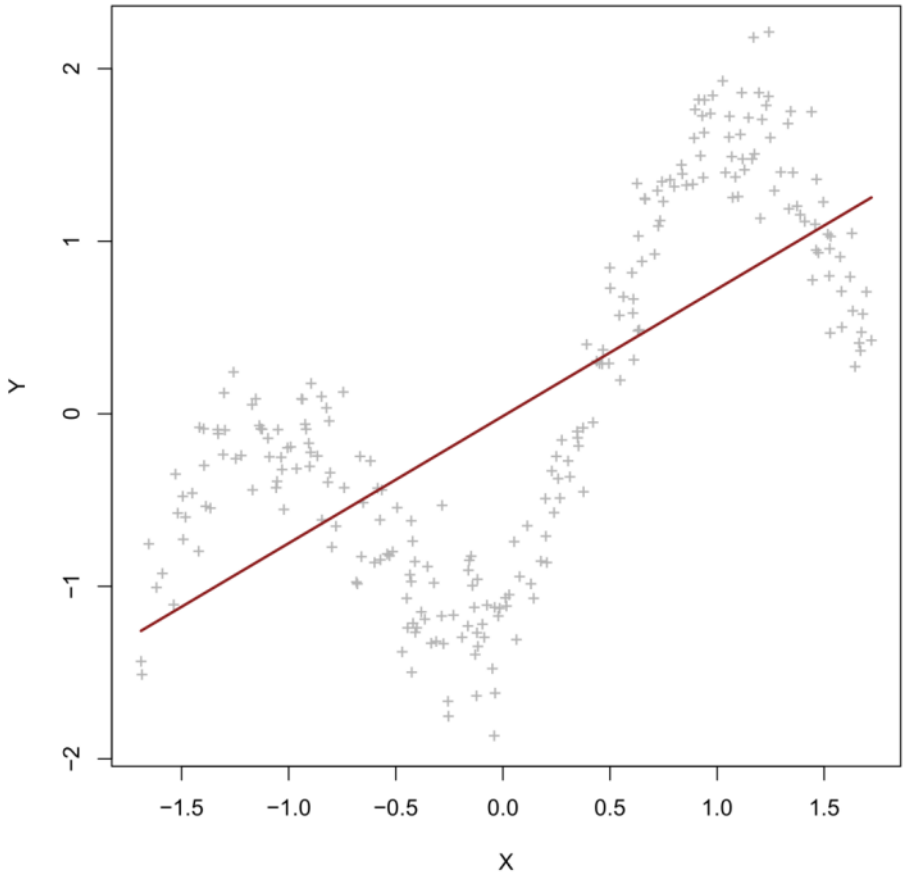
Modelling nonlinearity is the primary reason why most models of machine learning exist

- A simple way to define machine learning is that they are a group of models for nonlinear regression/classification

Nonlinearity in regression



Nonlinearity in regression

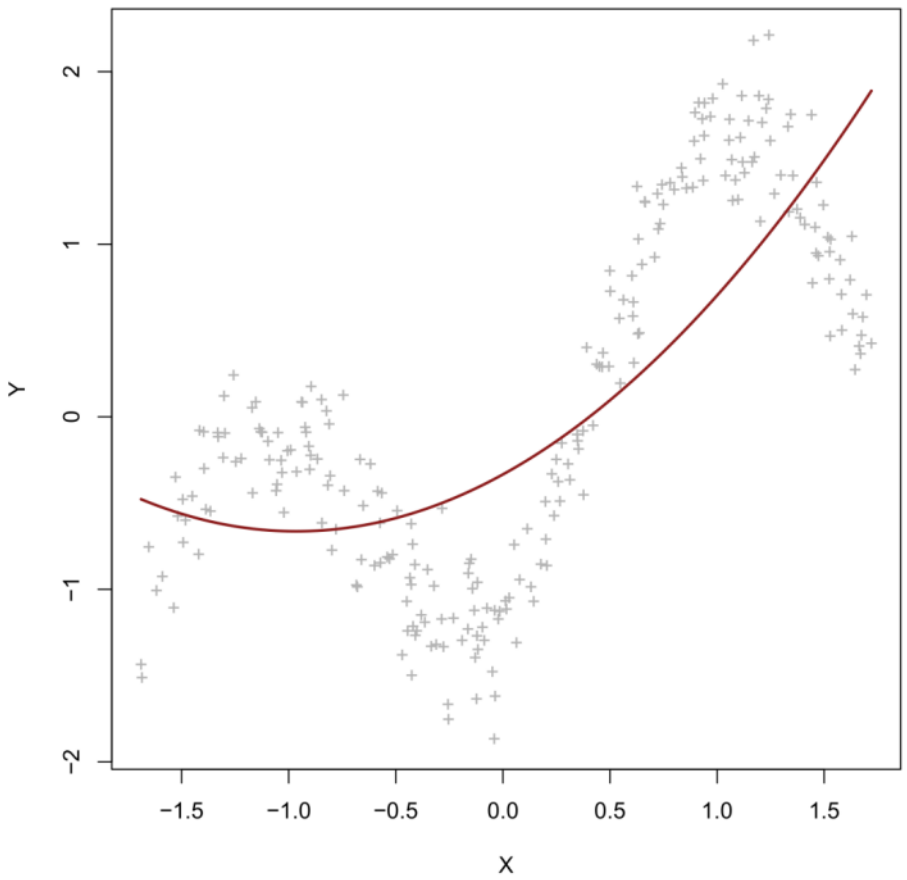


$E(Y) = b_0 + b_1X$

X
-1.52
-1.23
-0.88
-0.57
0.01
0.39
0.55
1.01
...

Dimension of the feature set: 1

Nonlinearity in regression

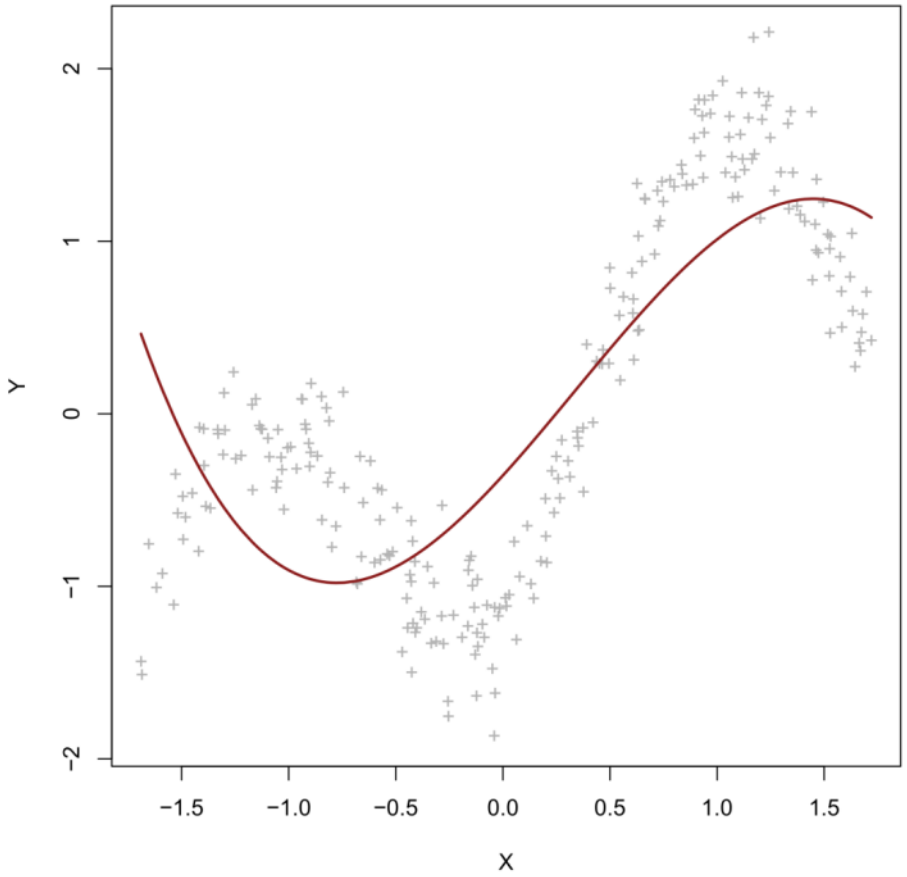


$$E(Y) = b_0 + b_1 X + \boldsymbol{b_2 X^2}$$

X	X^2
-1.52	2.31
-1.23	1.51
-0.88	0.77
-0.57	0.32
0.01	0.00
0.39	0.15
0.55	0.30
1.01	1.02
...	...

Dimension of the feature set: 2

Nonlinearity in regression

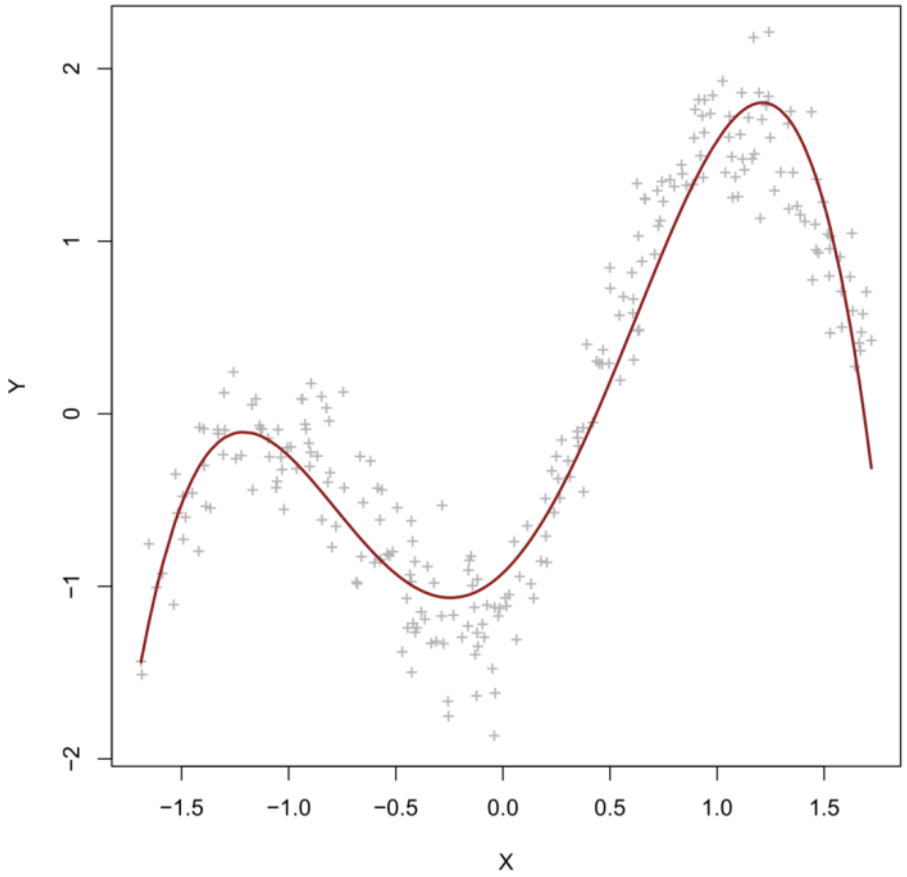


$$E(Y) = b_0 + b_1X + b_2X^2 + \mathbf{b_3X^3}$$

X	X^2	X^3
-1.52	2.31	-3.51
-1.23	1.51	-1.86
-0.88	0.77	-0.68
-0.57	0.32	-0.18
0.01	0.00	0.00
0.39	0.15	0.05
0.55	0.30	0.16
1.01	1.02	1.03
...	...	...

Dimension of the feature set: 3

Nonlinearity in regression



$$E(Y) = b_0 + b_1X + b_2X^2 + b_3X^3 + \mathbf{b_4X^4}$$

X	X^2	X^3	X^4
-1.52	2.31	-3.51	5.33
-1.23	1.51	-1.86	2.28
-0.88	0.77	-0.68	0.59
-0.57	0.32	-0.18	0.10
0.01	0.00	0.00	0.00
0.39	0.15	0.05	0.02
0.55	0.30	0.16	0.09
1.01	1.02	1.03	1.04
...	...	...	...

Dimension of the feature set: 4

Nonlinearity

- Many models of machine learning operate in the manner just shown.

  1. First the model will apply a clever nonlinear transformation to the independent/feature variables
  2. Then the model fits a basic linear model to the transformed data

- The preceding example shows a model that is linear in the parameters, but nonlinear in the features. The transformation used was a polynomial expansion of degree 4.

- A model can also be nonlinear in the parameters, but this kind of model is a minority in machine learning (the most famous example being artificial neural networks).

Parameters

- Most machine learning models have two types of parameters:

  1. Model parameters: coefficients or weights that have to be *estimated* from the sample/training data (e.g., regression weights).

  2. Tuning parameters: also called control parameters. These are options that can be switched on or off, or that are set to a certain value by the user in advance (e.g., the degree of a polynomial expansion). Normally *not* estimated from the data.

- We will see examples throughout this presentation…

4. K nearest neighbors

Intuition prelude

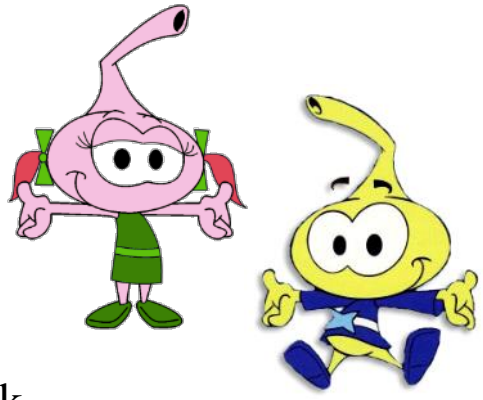
- Let us forget everything that I have talked about so far. We also forget about linear regression for a minute.
- Instead we focus on the goal of machine learning, which is to make accurate predictions based on existing knowledge.
- To achieve this, there are models that are far more simple than linear regression, and that nevertheless achieve a high degree of nonlinearity.
- Let us consider the following classification problem…

Classification problem

Training set
7 observations
3 classes

Smurf

Gummi bear

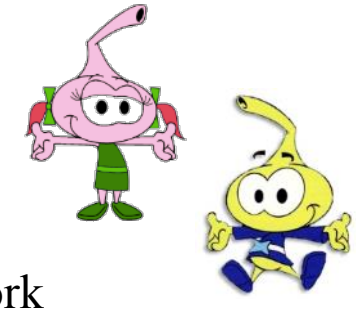


Snork

Classification problem

Which class would you predict?

Smurf



Snork

Gummi bear

Test set

Nearest neighbors

- Congratulations, you have used a nearest neighbor rule to make a prediction!
- The unclassified character resembled most the Smurf characters visually, so an easy decision.
- This reasoning captures the nearest neighbor rule in a nutshell: We classify a (new) observation according to the most similar—or least dissimilar—observation in our known database (i.e., our training set).
- For regression with a continuous dependent variable, the same rule applies. When predicting a response value for a new observation, we predict the same response value that the nearest known observation has.

Features of the characters

Data	ID	Skin color	Hat	Hat color	Pants	Gender	Class
Training	Papa Smurf	Blue	Yes	Red	Yes	Male	Smurf
Training	Brainy Smurf	Blue	Yes	White	Yes	Male	Smurf
Training	Sassette Smurf	Blue	Yes	White	Yes	Female	Smurf
Training	Casey Kelp	Pink	No		No	Female	Snork
Training	Allstar Seaworthy	Yellow	No		No	Male	Snork
Training	Cubbi Gummi	Pink	Yes	Blue	No	Male	Gummi Bear
Training	Tummi Gummi	Blue	Yes	Red	No	Male	Gummi Bear
Test	Smurfette	Blue	Yes	White	No	Female	?

Features/independents

Dependent

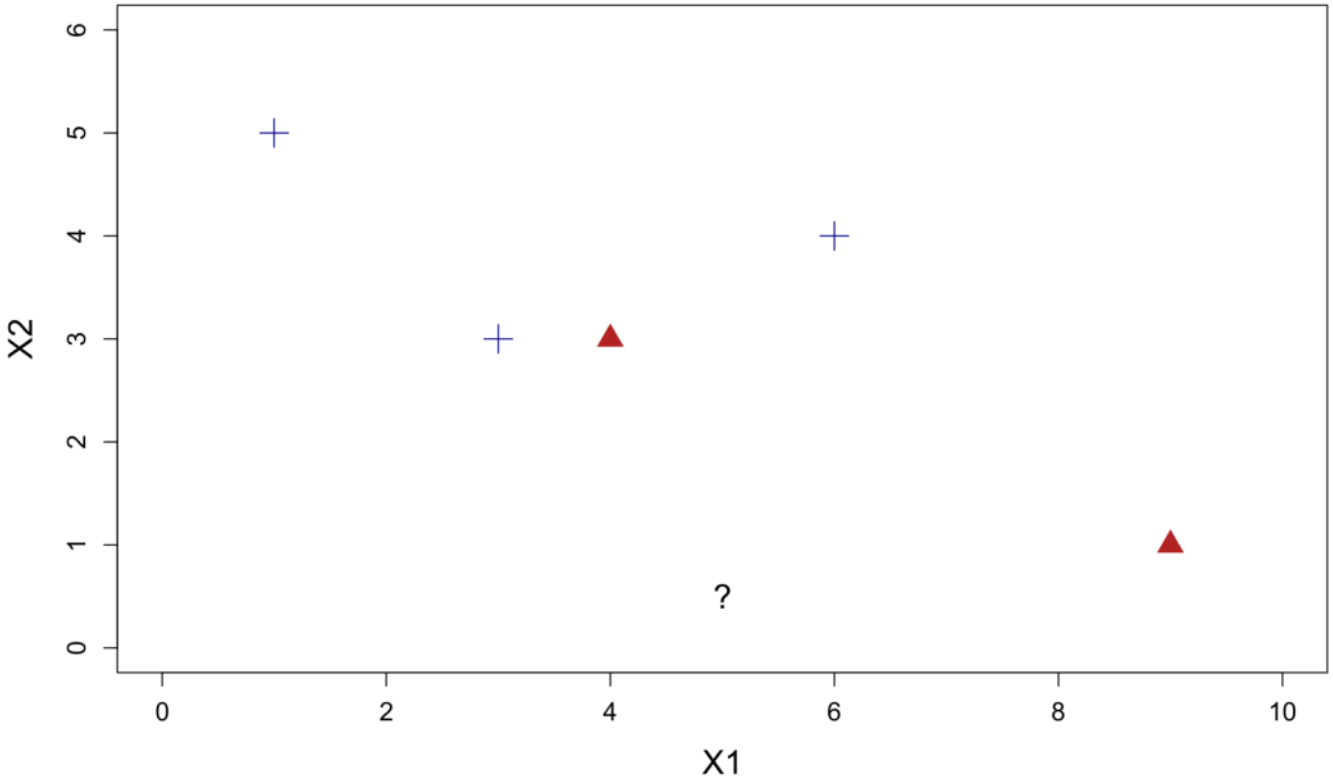
Nearest neighbors

Data	ID	Skin color	Hat	Hat color	Pants	Gender	Class
Training	Papa Smurf	**Blue**	**Yes**	Red	Yes	Male	Smurf
Training	Brainy Smurf	**Blue**	**Yes**	**White**	Yes	Male	Smurf
Training	Sassette Smurf	**Blue**	**Yes**	**White**	Yes	**Female**	Smurf
Training	Casey Kelp	Pink	No		**No**	**Female**	Snork
Training	Allstar Seaworthy	Yellow	No		**No**	Male	Snork
Training	Cubbi Gummi	Pink	**Yes**	Blue	**No**	Male	Gummi Bear
Training	Tummi Gummi	**Blue**	**Yes**	Red	**No**	Male	Gummi Bear
Test	Smurfette	**Blue**	**Yes**	**White**	**No**	**Female**	?

- No perfect match based on these features, but Smurfette is the most similar to Sassette (4 feature matches; class Smurf). She is the most dissimilar to Allstar Seaworthy (1 feature match; class Snork).

Dissimilarity

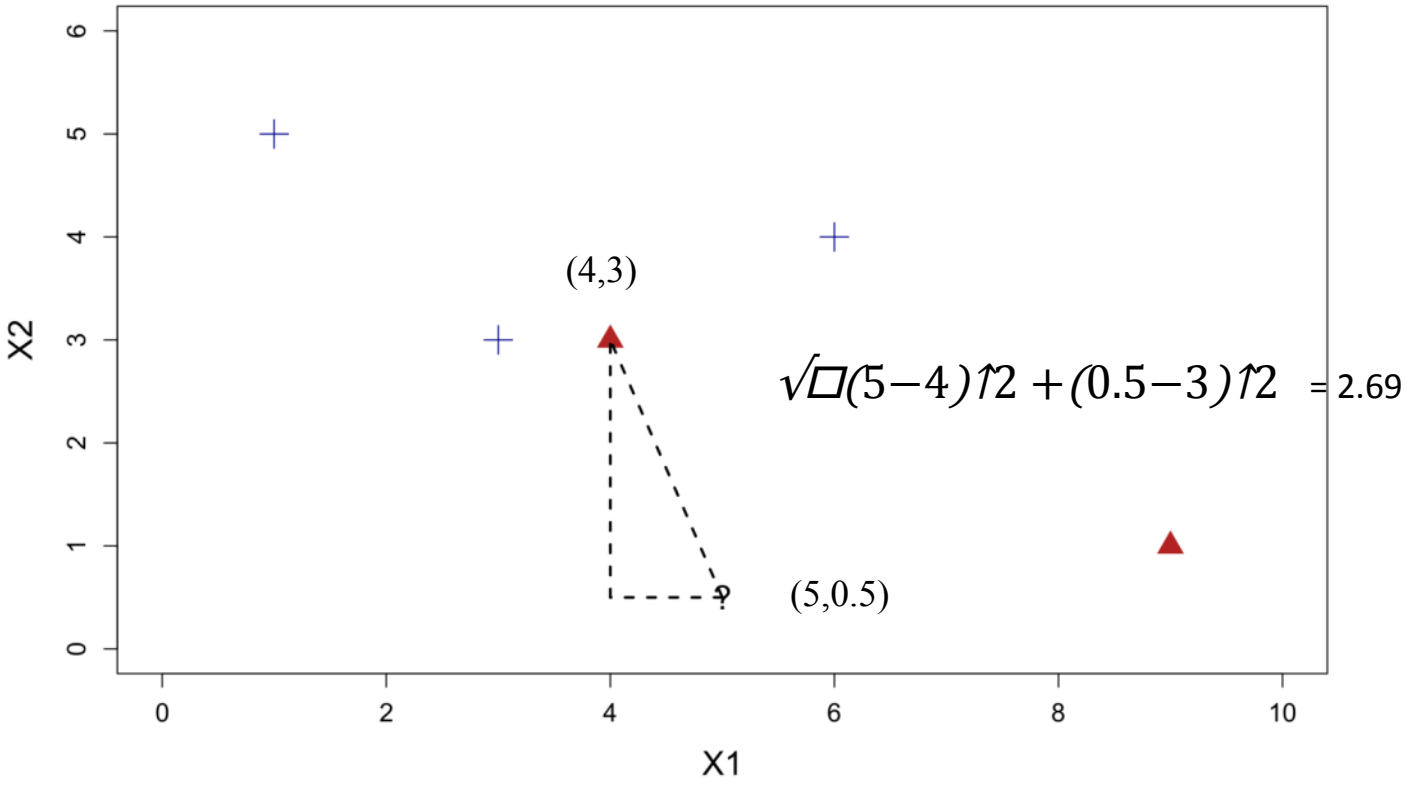
- There are many ways to quantify dissimilarity between two observations in a data set. One way is to measure the distance between the objects in the feature space. The most well-known distance is Euclidean distance:



Euclidean distance matching

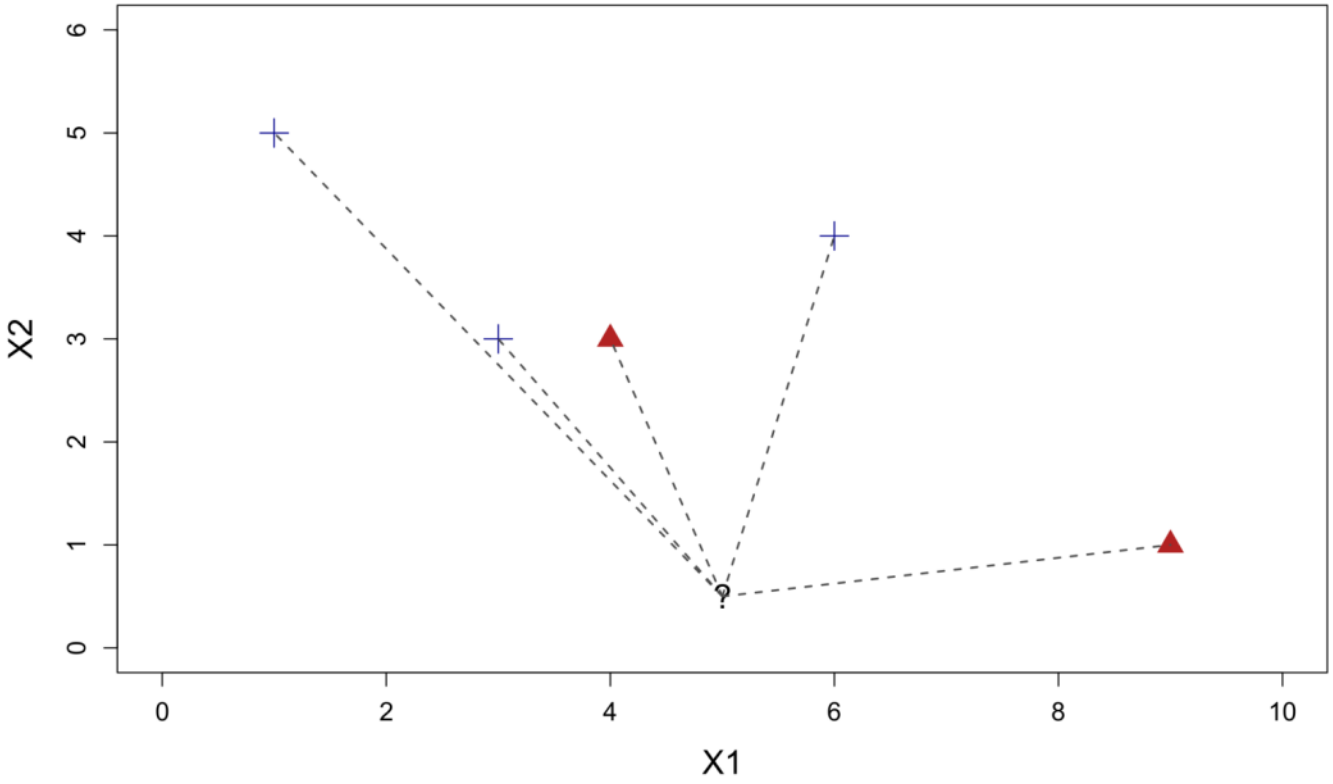
- Euclidean distance between two observations on P features:

$$D_{euc} = \sqrt{(x_{11} - x_{12})^2 + (x_{21} - x_{22})^2 + \ldots + (x_{p1} - x$$



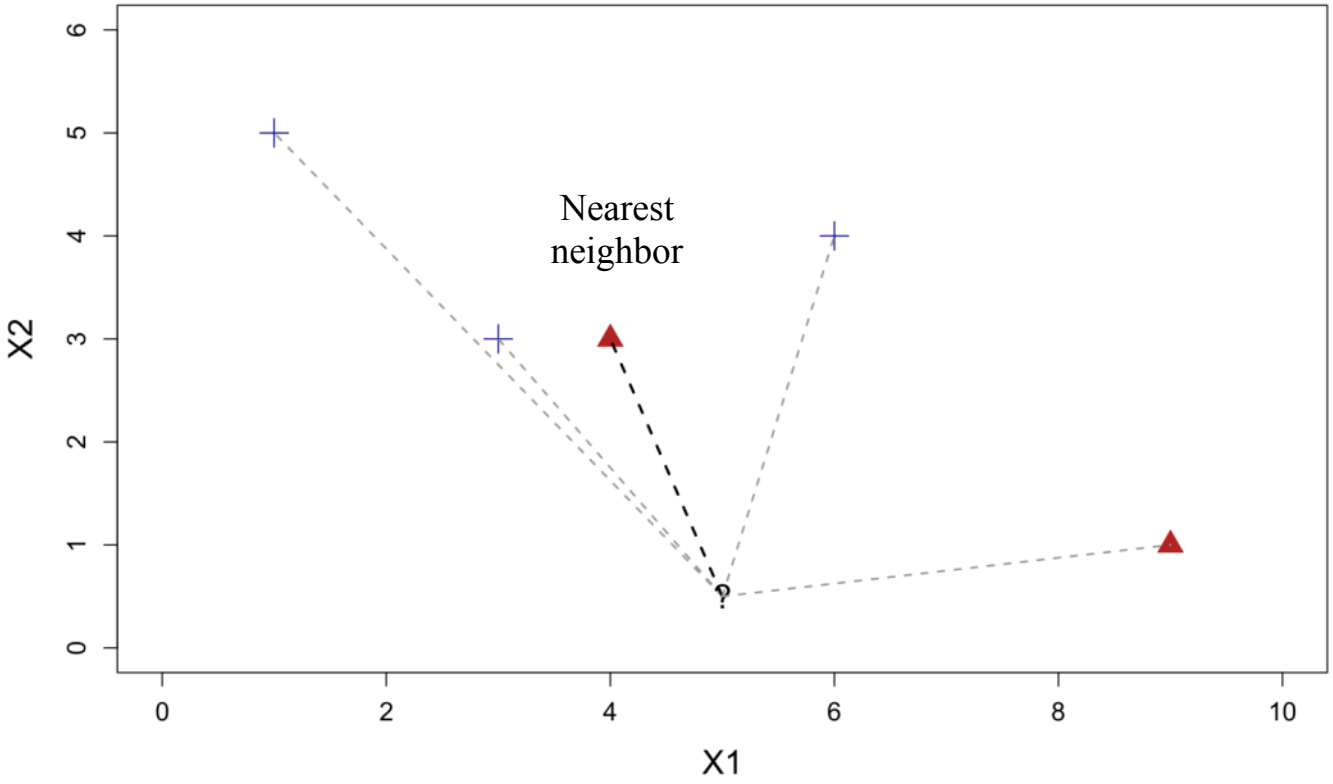
Euclidean distance matching

- Calculating all pairwise distances between observations in this manner will yield a (Euclidean) distance matrix that is symmetric and of size $N \times N$. Diagonal elements have 0 distance by definition (self-distance).



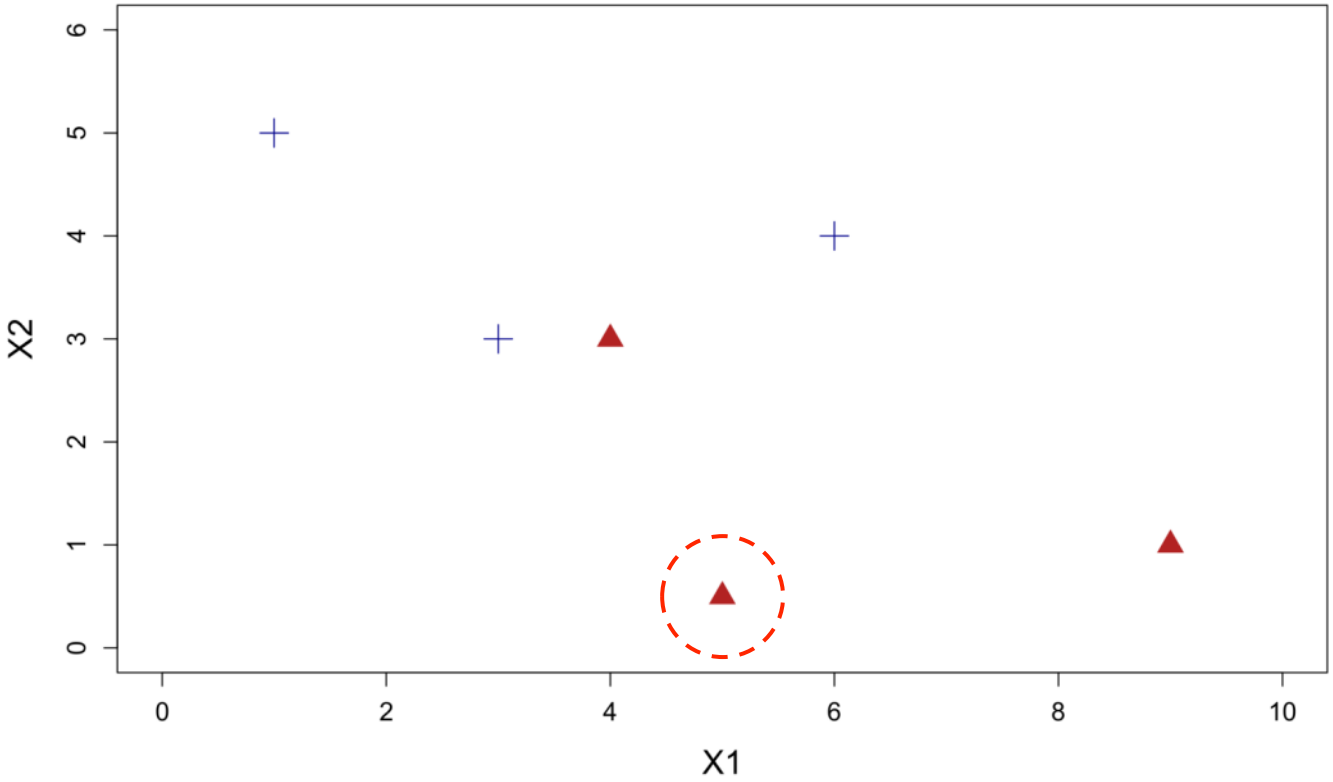
Euclidean distance matching

- The nearest neighbor to an observation is *another* observation that has the shortest distance to the original observation:



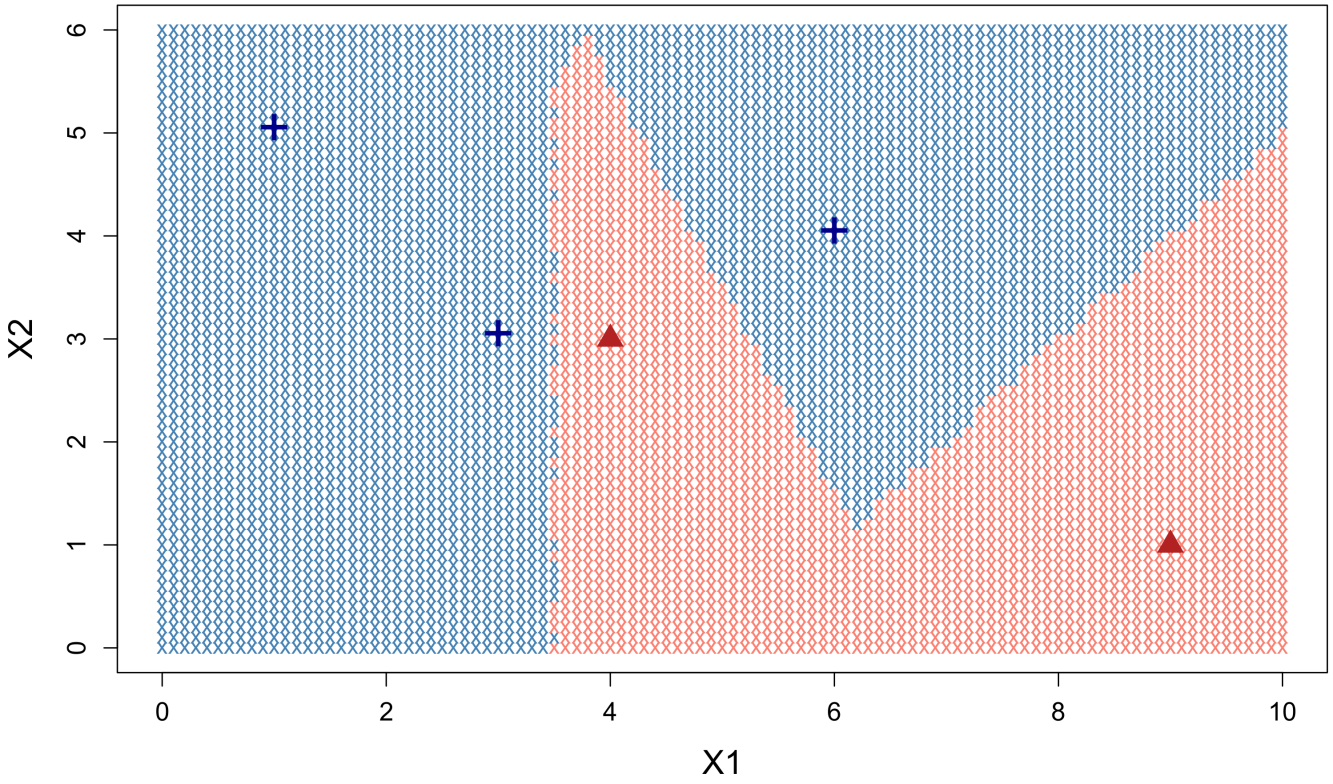
Euclidean distance matching

- The predicted class for this observation is then the same as its nearest neighbor:



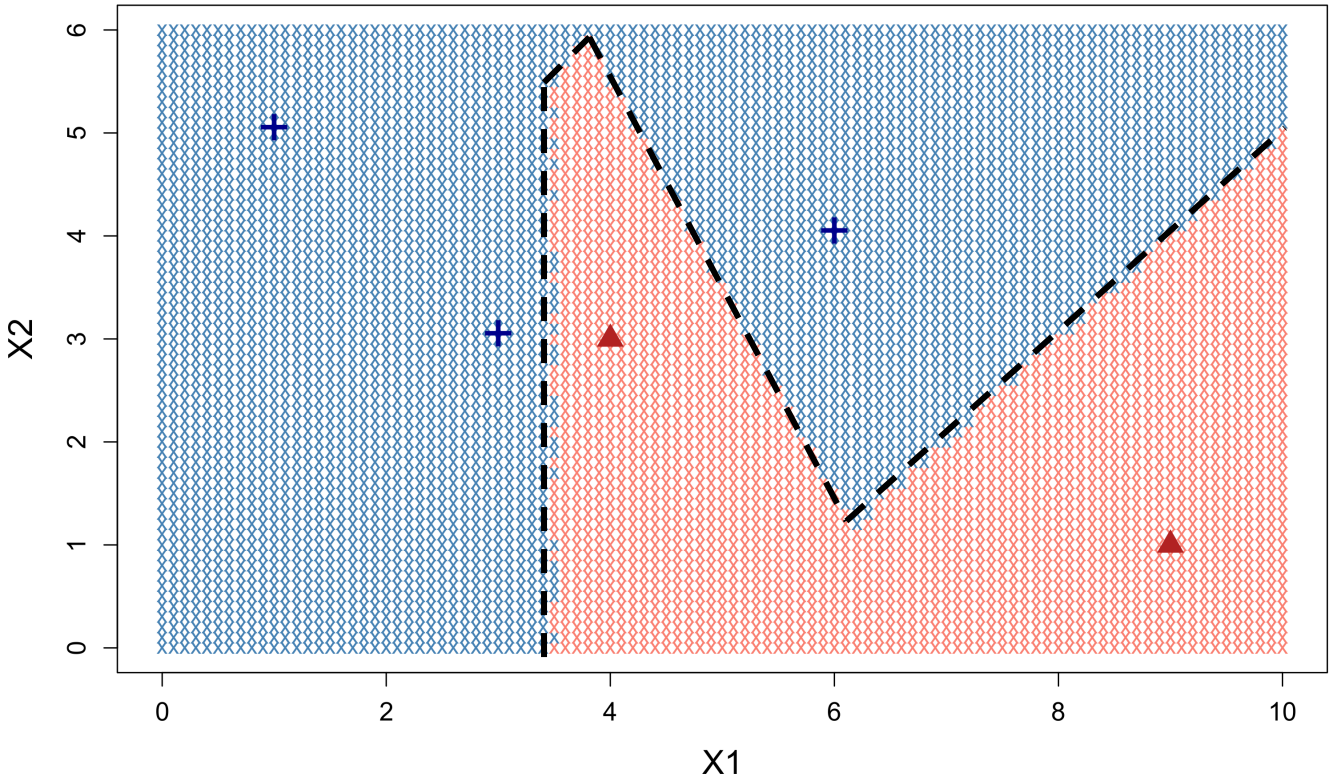
1-nearest neighbor model

- The rule described previously is known as the **1-nearest neighbor model (1NN)**, since prediction is based simply on the nearest neighbor. The decision boundary for this example can be visualized as follows:



1-nearest neighbor model

- In the red part of the feature space, the one nearest neighbor is always a triangle. In the blue part of the feature space, the one nearest neighbor is always a cross.



1-nearest neighbor model

- The decision boundary produced by 1NN is nonlinear. How does this come about? What is the "model" for nearest neighbors?

- It turns out that there is no model in the conventional sense (e.g., weights, assumptions). Instead the model consists of the entire training set, where every training observation functions as its own sub-model.

- That is, the nearest neighbor in a given region of the feature space will act as the *model* for that space (e.g., if the nearest neighbor is a Smurf, then Smurf is the model in that region). More generally, one says that the nearest neighbor acts as a local rule for prediction.

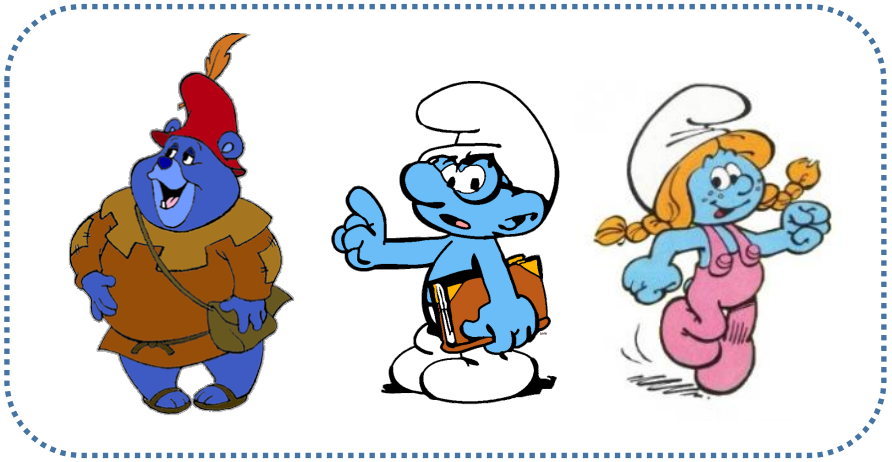
- 1NN with Euclidean distance is locally linear! When adding up all the local models, however, a globally nonlinear decision boundary emerges (e.g., analogous to regression smoothers).

K nearest neighbors

- 1NN is the simplest version of what is generally known as *K* nearest neighbors (KNN).

- Instead of considering just one neighbor to generate a new prediction, we could consider *K* neighbors, and combine their predictions. E.g., for the Smurfette problem:

Test set



3 nearest neighbors

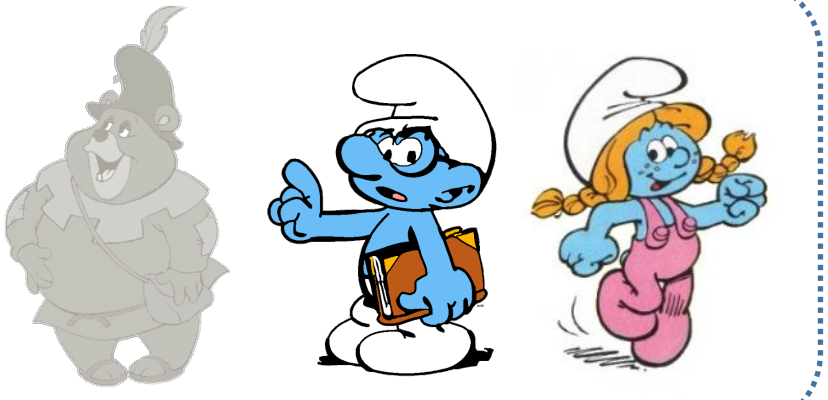
K nearest neighbors

Data	ID	Skin color	Hat	Hat color	Pants	Gender	Class
Training	Papa Smurf	**Blue**	**Yes**	Red	Yes	Male	Smurf
Training	Brainy Smurf	**Blue**	**Yes**	**White**	Yes	Male	Smurf
Training	Sassette Smurf	**Blue**	**Yes**	**White**	Yes	**Female**	Smurf
Training	Casey Kelp	Pink	No		**No**	**Female**	Snork
Training	Allstar Seaworthy	Yellow	No		**No**	Male	Snork
Training	Cubbi Gummi	Pink	**Yes**	Blue	**No**	Male	Gummi Bear
Training	Tummi Gummi	**Blue**	**Yes**	Red	**No**	Male	Gummi Bear
Test	Smurfette	**Blue**	**Yes**	**White**	**No**	**Female**	?

Majority voting

Test set

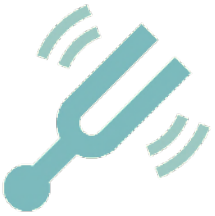
3 nearest neighbors



- The simplest way to combine information from multiple neighbors is to average their predictions. When the dependent is categorical (classification problem) this amounts to a process of majority voting.

- In the example above, 2 out of 3 neighbors predict Smurf. Since this is the majority, the final prediction of this 3NN model for Smurfette is Smurf.

Tuning parameter: *K*



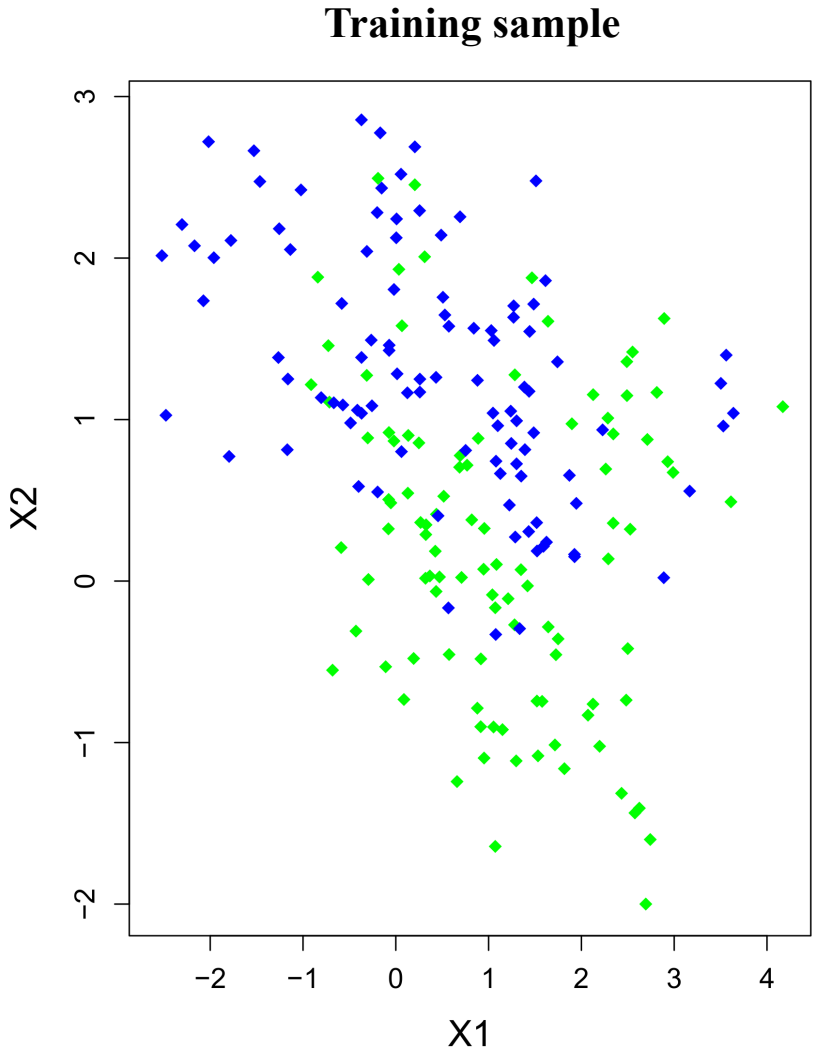
- The primary tuning parameter of KNN is the number of neighbors, *K*. Unless there is a theoretical reason to fix this number in advance, *K* is usually selected by (internal) cross-validation.

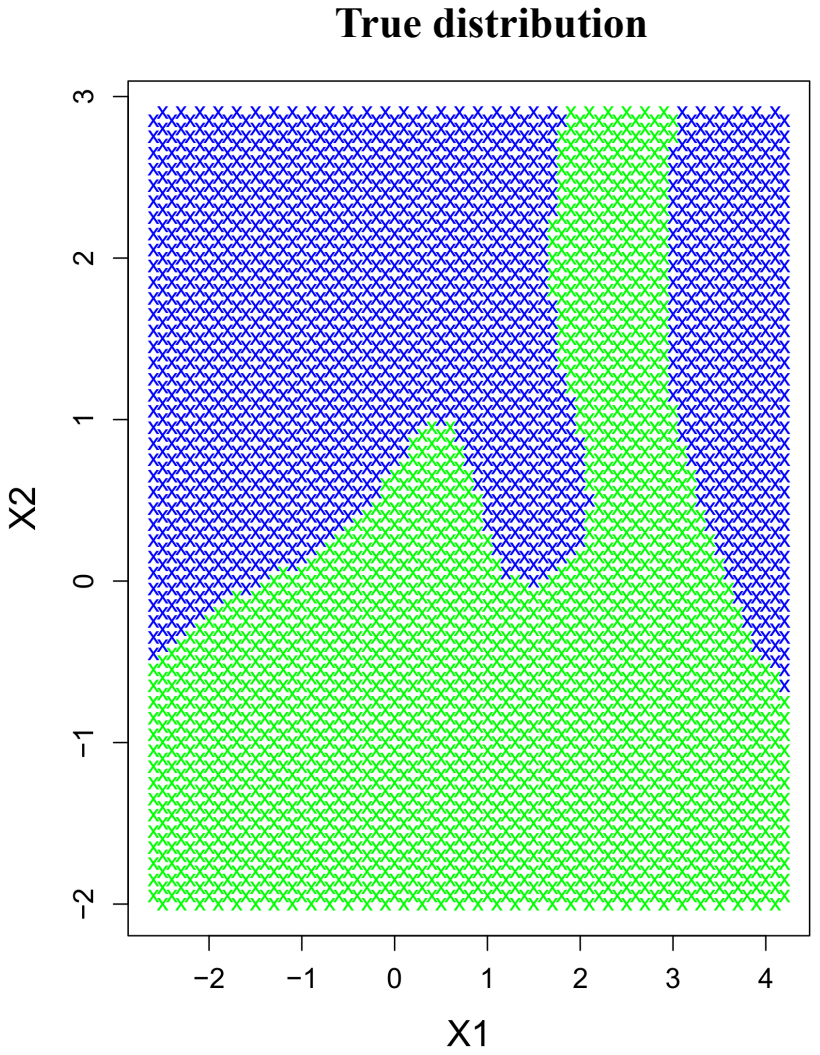
- Fortunately, there are many packages in R that automate this selection process, and will return the optimal value for *K* (see later).

- It turns out that the value of *K* acts as a kind of smoothing parameter. When *K*=1, the decision boundary produced by the model can be highly variable/unstable. Setting *K* to higher values will produce smoother decision boundaries.

- For *K*=*N*, the model simply defaults to predicting the majority class in the training set. This type of bias will already become evident when *K* approaches *N*. For this reason, optimal values of *K* tend to be relatively small.

Two-class data problem





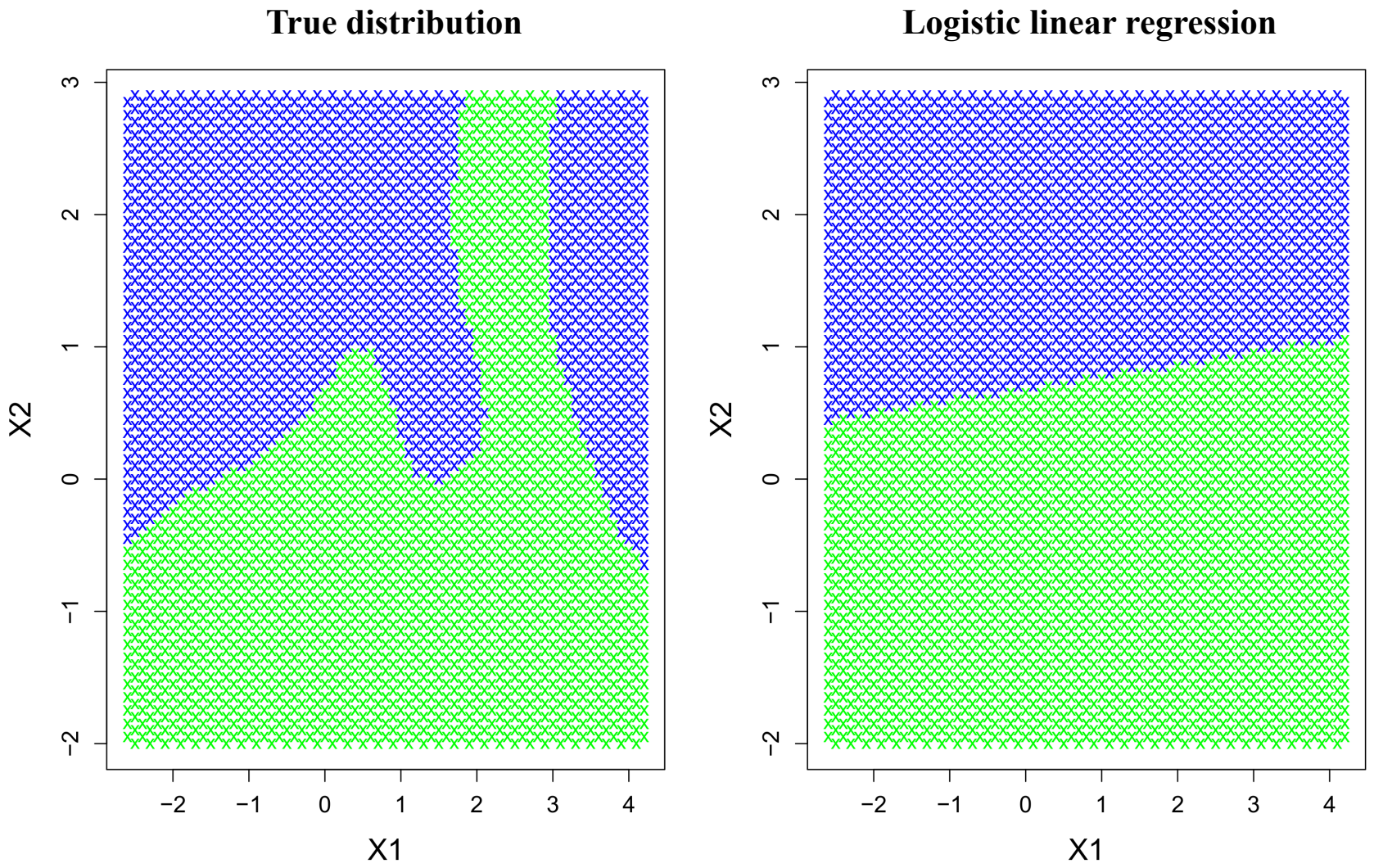
Two-class data problem

ID	color	X1	X2
1	Blue	1.432	0.306
2	Green	2.489	1.358
3	Green	0.320	0.017
4	Blue	3.503	1.224
5	Blue	0.259	1.250
6	Green	-0.079	0.323
…	…	…	…
N	Green	2.499	-0.418

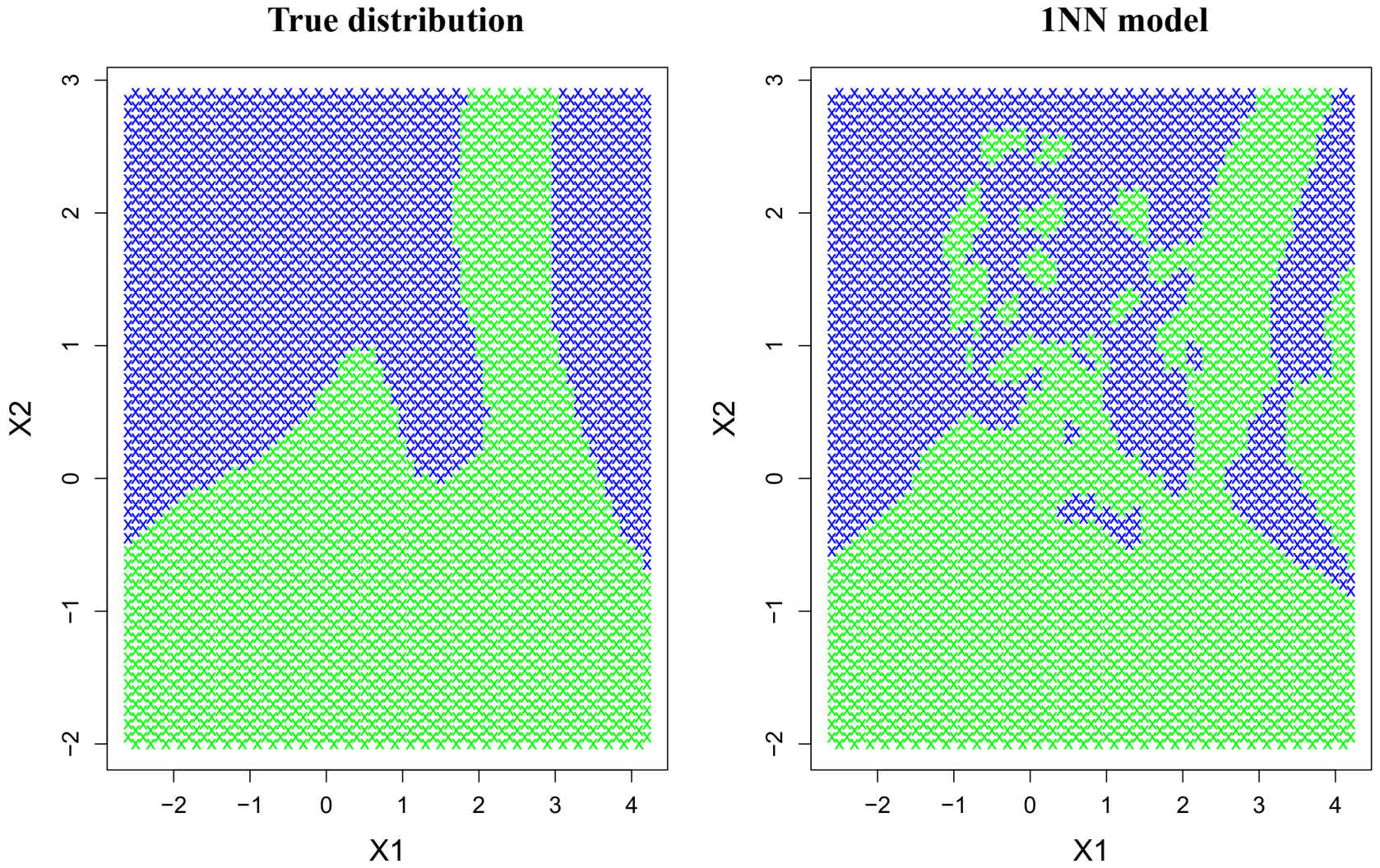
Two-class data problem

- Some sample/training data (left panel) were generated from an underlying distribution (right panel).
- There are two features/independents, X1 and X2, and there is one response/ dependent, color (blue or green).
- A good model should be able to learn the true distribution of these two classes, based on the sample data.
- The plot shows clearly that the boundary between the blue and green classes is highly nonlinear. A linear model is expected to perform very poorly for this problem…

Linear model generates this



1 nearest neighbor



K nearest neighbors in R

- The error of the 1NN model is approximately 20% (missclassification). Perhaps setting the number of neighbors (*K*) higher could improve this…

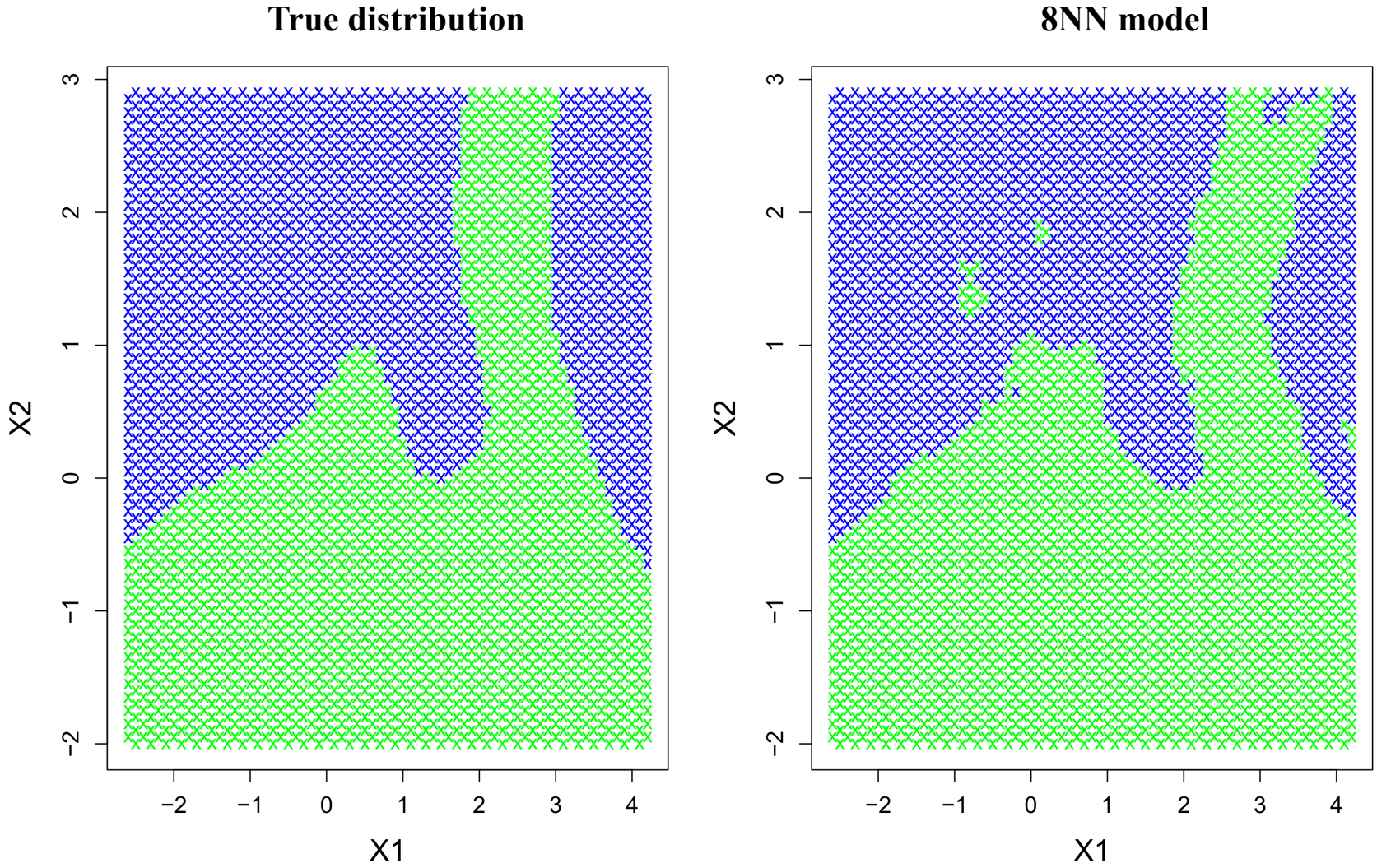
- The `kknn` package in R includes a handy function for estimating the optimal value of *K* by leave-one-out cross-validation:

```
train.kknn(color~X1+X2,data=training,kmax=101)

> Call:
> train.kknn(formula=color~X1+X2,data=training,kmax=101)
>
> Type of response variable: nominal
> Minimal misclassification: 0.155
> Best kernel: optimal
> Best k: 8
```

- According to this cross-validation, *K*=8 is the optimal number of nearest neighbors for prediction in this data problem.

8 nearest neighbors



K nearest neighbors in R

- For the purpose of generating predictions, `kknn` is not always so useful. Instead the `ipred` package is helpful:

```
library(ipred)

knn <- ipredknn(color~X1+X2,data=training,k=8)
predictions <- predict(knn,newdata=validation,type="class")

table(True=validation$color,Predicted=predictions)
>       Predicted
> True   blue green
>  blue  1179   408
>  green  263  1565

mean(predictions!=validation$color)
> 0.09122
```

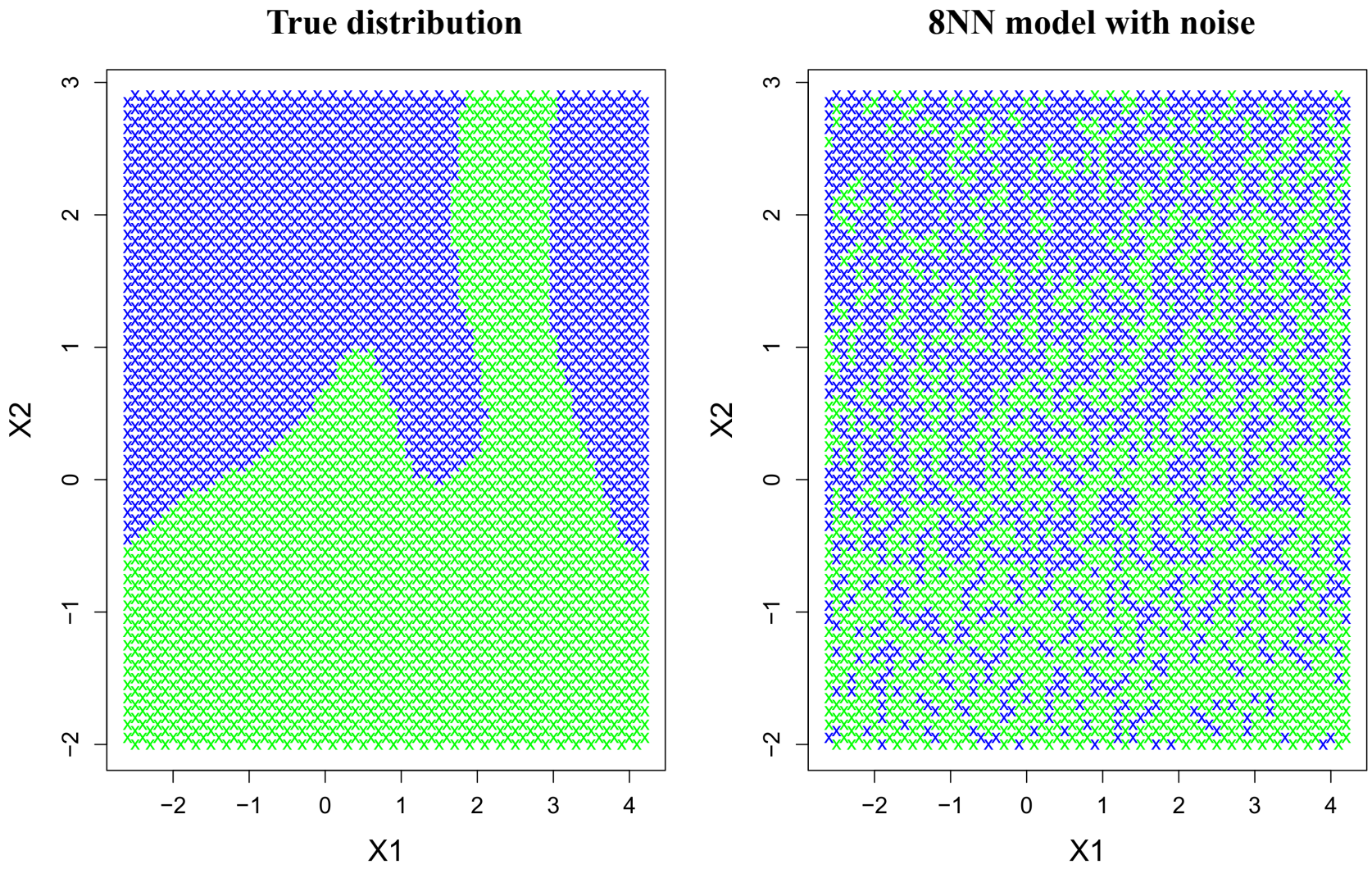
1 nearest neighbor

- The error of the 8NN is only 9%. A huge improvement! This is fantastic, so we can stop modelling and go home, right?

- No! What happens if we add a lot of irrelevant/random features to the data set, and refit the model…

ID	color	X1	X2	R1	R2	R3	…	R20
1	Blue	1.432	0.306	0.110	0.676	0.706	0.513	0.604
2	Green	2.489	1.358	0.606	0.394	0.197	0.834	0.222
3	Green	0.320	0.017	0.524	0.941	0.205	0.694	0.603
4	Blue	3.503	1.224	0.461	0.143	0.613	0.104	0.485
5	Blue	0.259	1.250	0.296	0.625	0.616	0.289	0.675
6	Green	-0.079	0.323	0.038	0.334	0.781	0.237	0.759
…	…	…	…	…	…	…	…	…
N	Green	2.499	-0.418	0.884	0.769	0.062	0.833	0.033

KNN sensitivity to noise



Feature relevance in KNN

- In the previous example I added 20 extra features to the two-class data, consisting of random noise. As a result the performance of the 8NN model reduced substantially (37% error).

- A notable disadvantage of KNN is that it cannot detect automatically which features are relevant and irrelevant . All features receive equal weight when calculating the nearest neighbor distances.

- One solution to this problem is to conduct univariate pre-screening of the features in advance, e.g., with *t*-tests or correlation tests, and keep only significant features for the ensuing KNN.

- Disadvantages of this procedure are that **(a)** we again introduce parametric assumptions with the pre-screening, **(b)** make linear assumptions, and **(c)** use hypothesis testing. P-values are flawed in many situations and we may miss out substantial nonlinear effects of certain features.

Why should I use KNN (in social science)?

- Nearest neighbor rules may mimic certain cognitive processes of categorization (e.g., semantic, social). The psychological equivalent of KNN is often referred to as exemplar-based categorization, and has been a core topic in the work of Robert Nosofsky (see Nosofsky & Palmeri, 1997).

- Thus, KNN could be used to validate exemplar-based models in psychological categorization.

- KNN is a relatively non-parametric method for nonlinear modelling. As such, it could be a basic nonlinear benchmark against linear alternatives. This is comparable to using a LOWESS smoother as an alternative against a standard linear model.

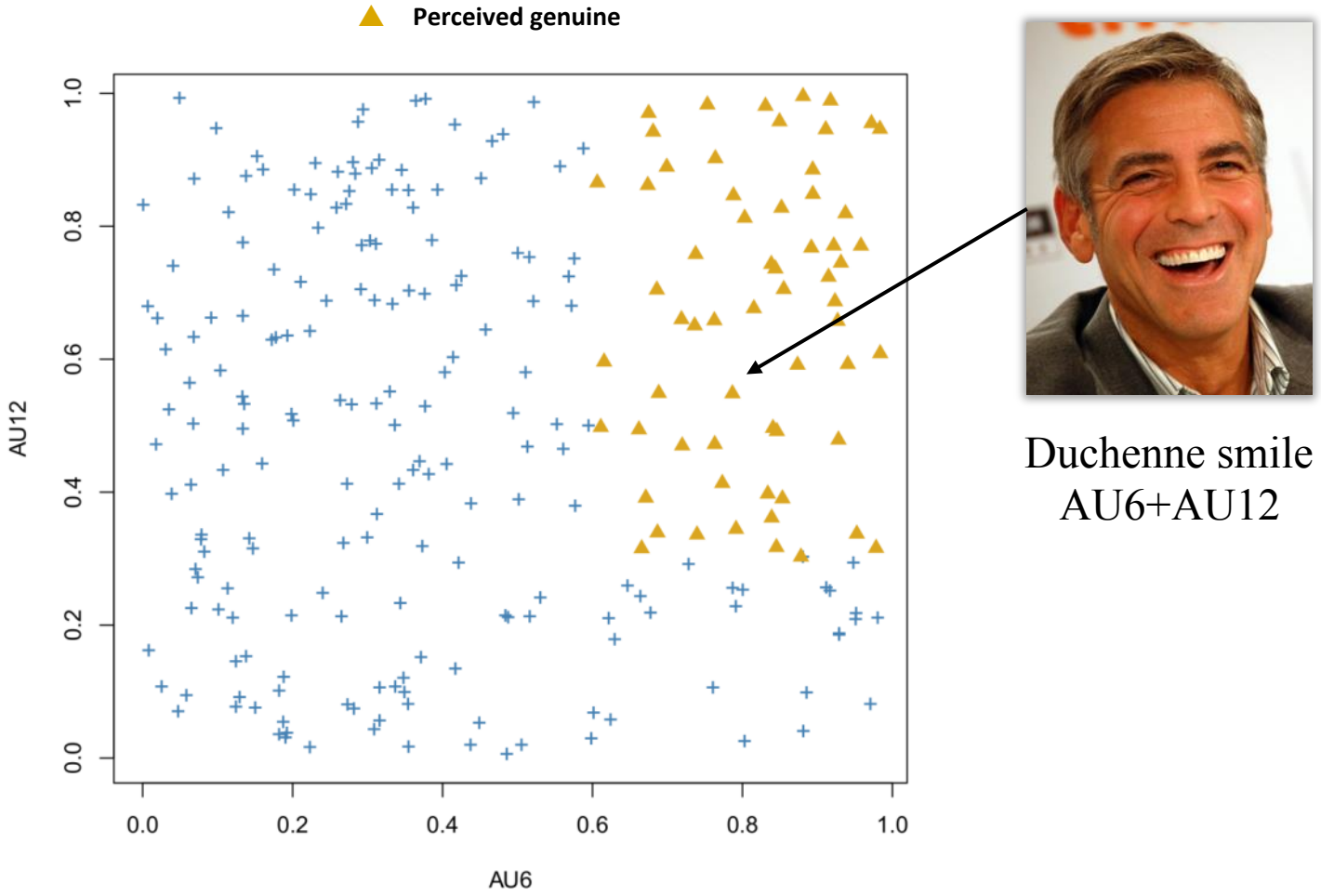
- For small samples with relatively few features it can be quite powerful and interpretable!

KNN evaluation

- Advantages:
  - \+ Intuitive logic of the nearest neighbor rule
  - \+ Relatively non-parametric
  - \+ Highly adaptive to nonlinear patterns
  - \+ Only one real tuning parameter, *K,* the number of neighbors
- Disadvantages
  - – High variance in the 1-nearest neighbor model
  - – No automatic evaluation of feature relevance
  - – Presence of irrelevant features add noise and severely harm performance
  - – Computationally expensive

5. Decision trees

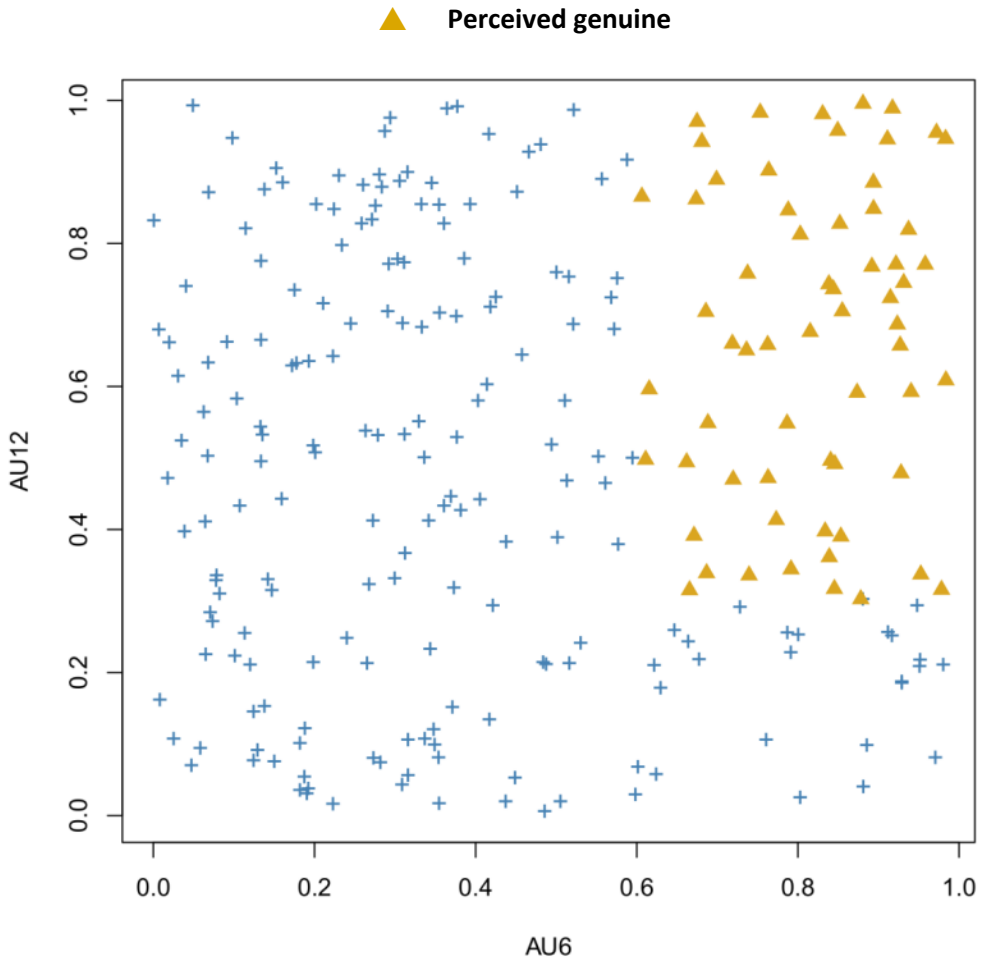
Time for intuition again…



Duchenne smile
AU6+AU12

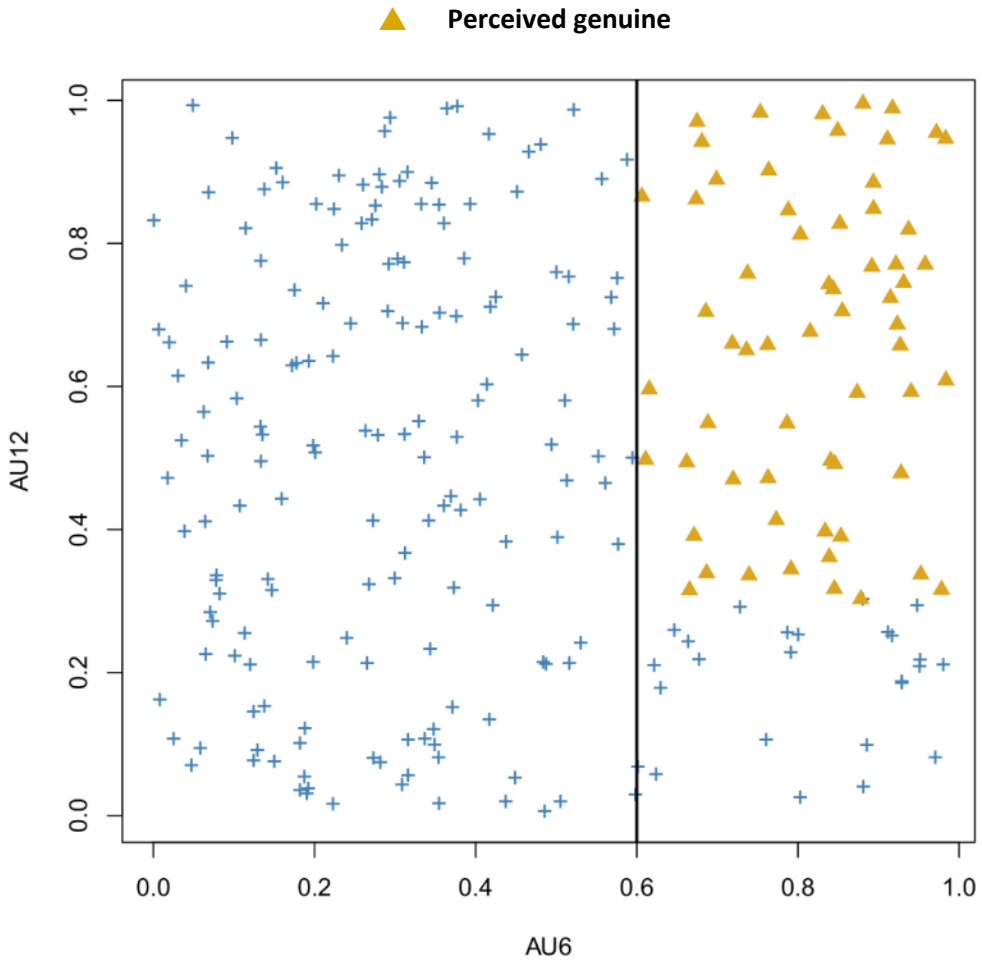
- Subjects rated 250 photos of smiles. For each photo, they rated whether they thought the smile looked fake or genuine. Independently, researchers coded the photos for facial action unit (AU) activity.

Time for intuition again…

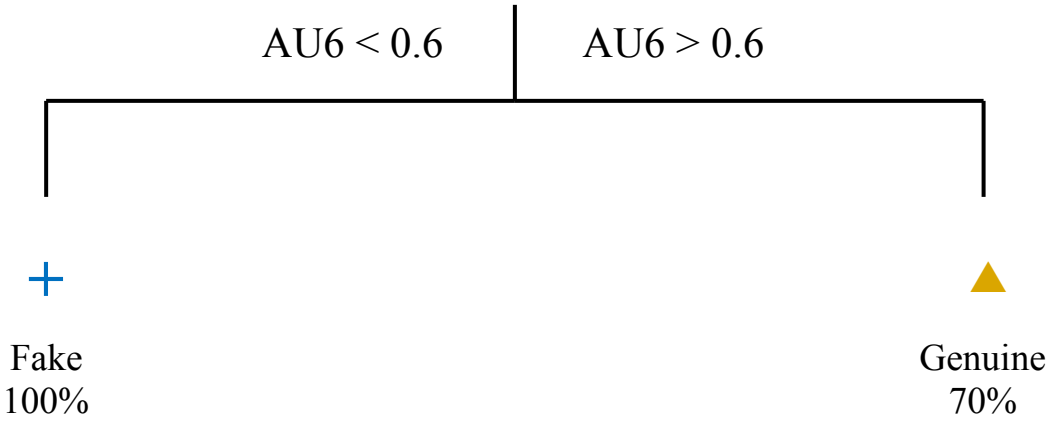


- Based on this plot, can you come up with a good rule for predicting genuine smiles…?

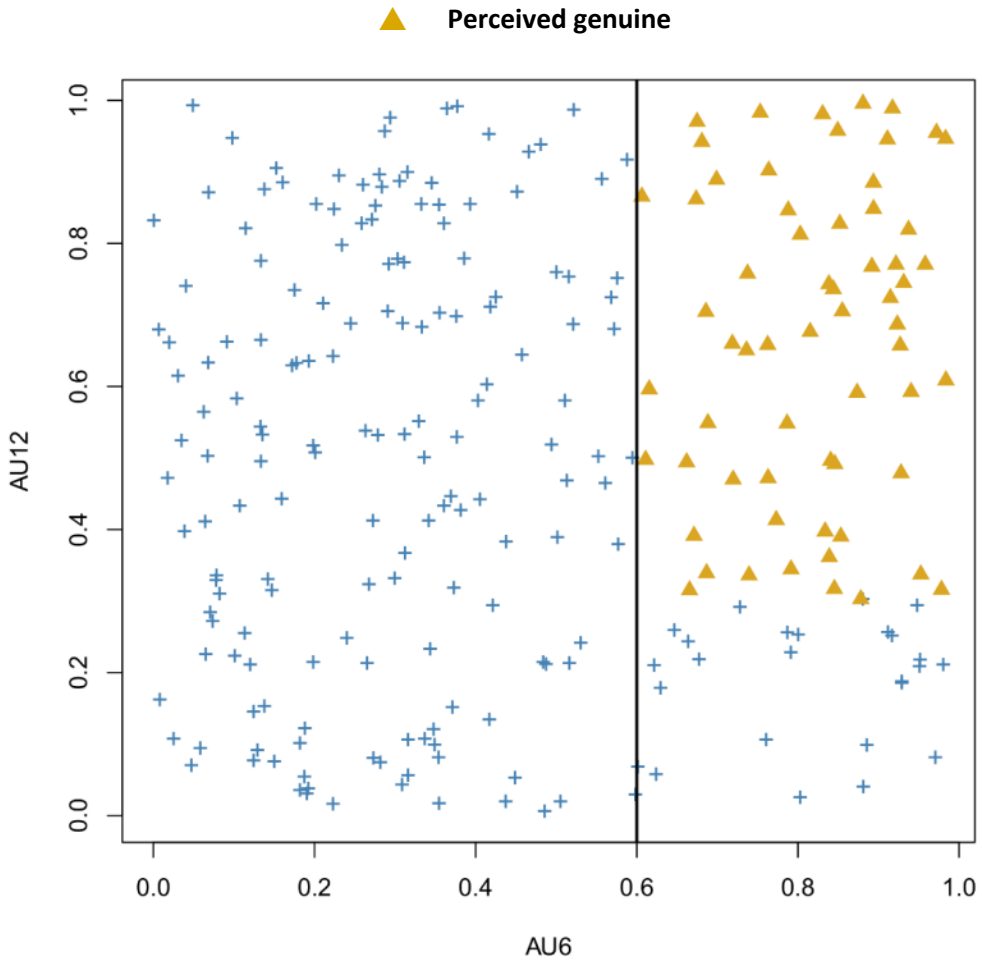
Time for intuition again…



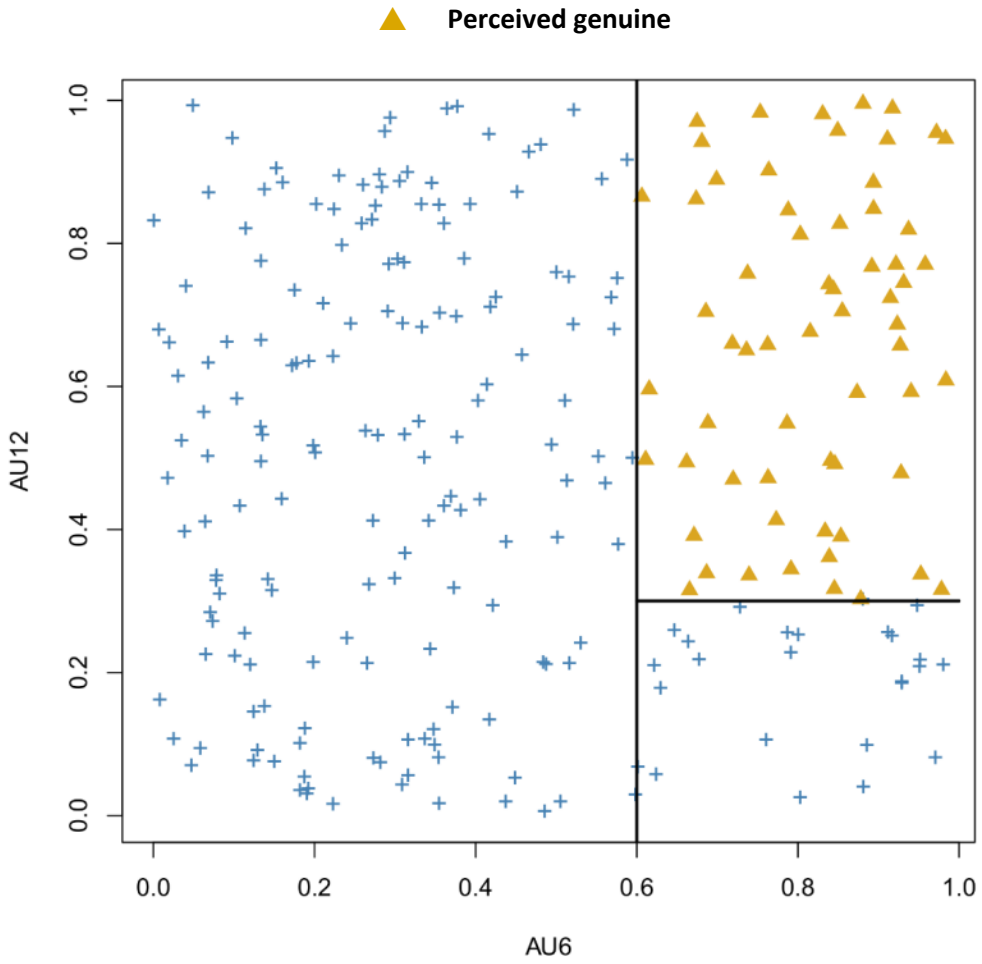
Time for intuition again…



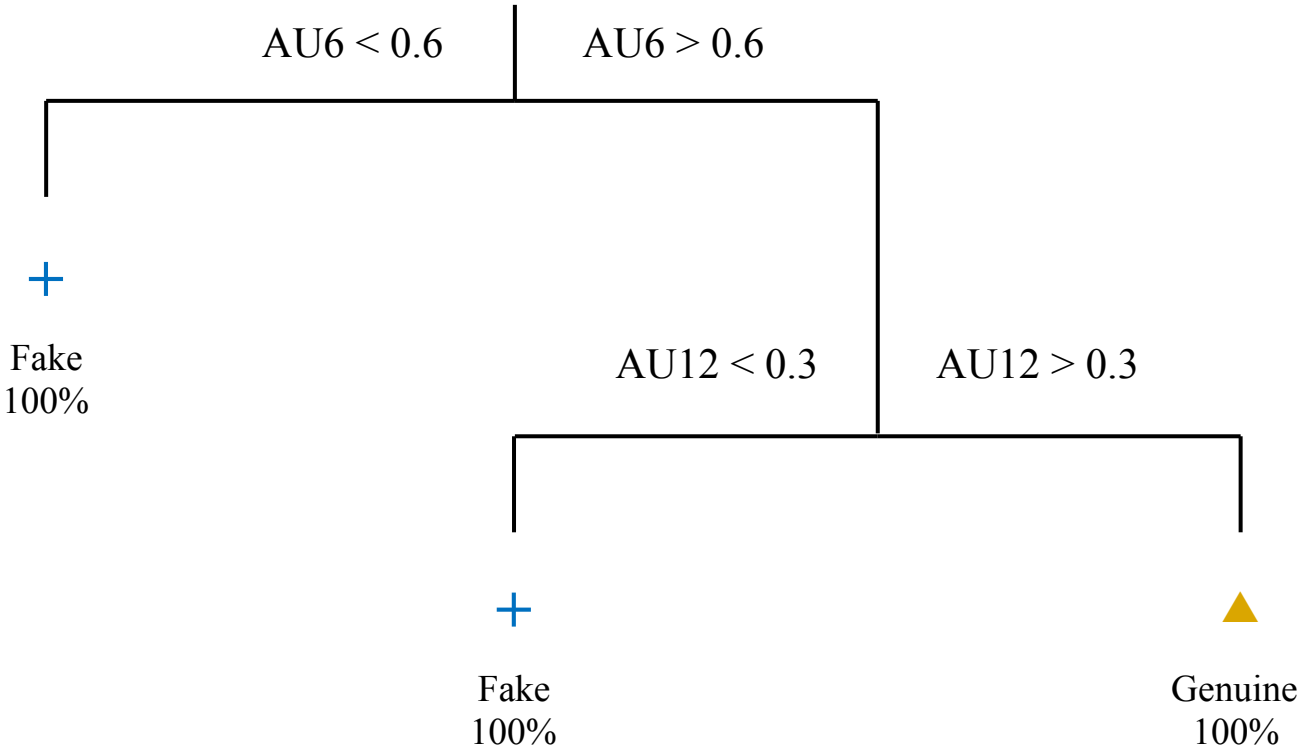
Time for intuition again…



Time for intuition again…



Time for intuition again…



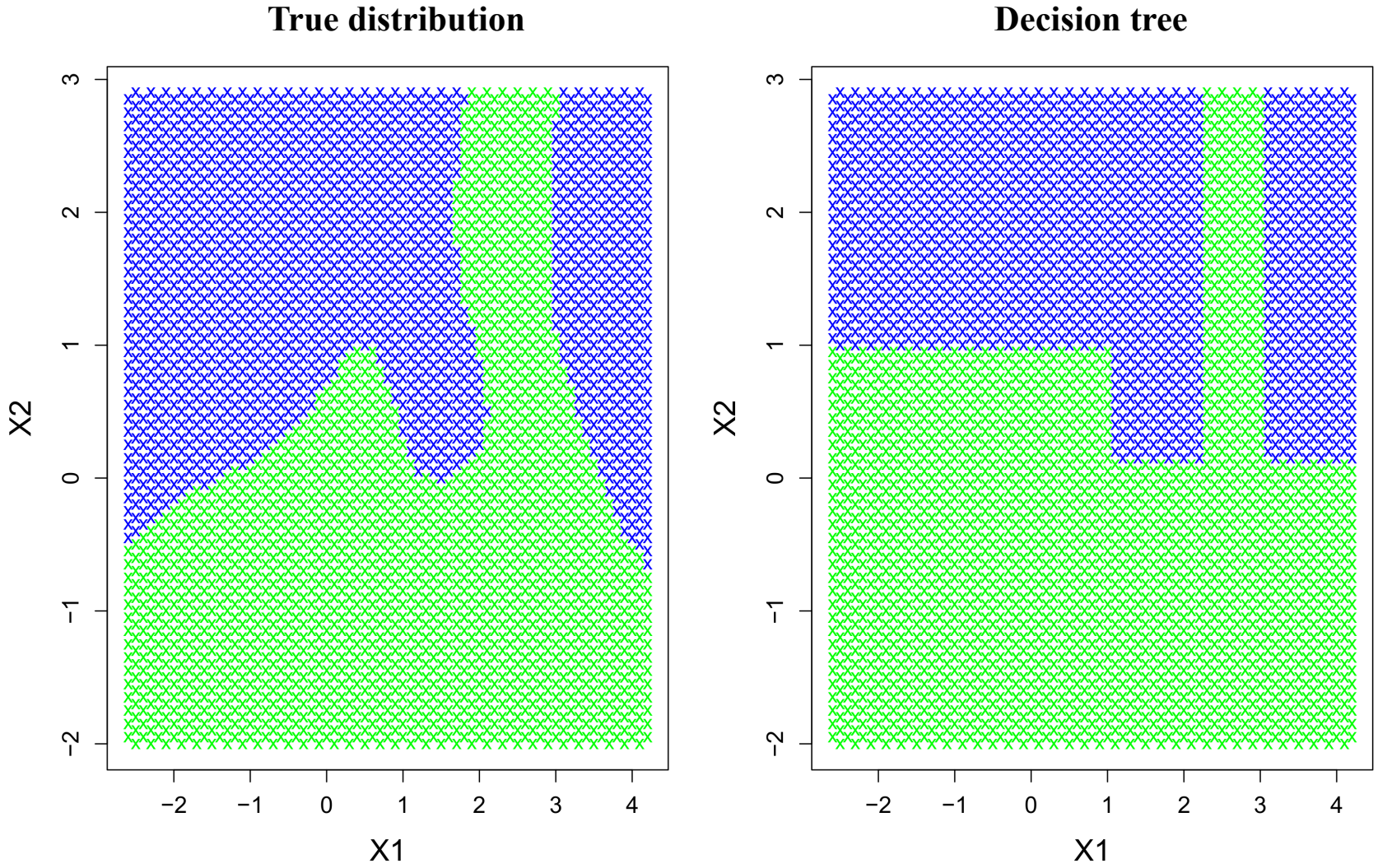
Decision trees

- We have just "fitted" a basic decision tree to these data. All we have done is construct a sequence of if-else rules to predict the response classes. The statistical version of this logic is known formally as recursive partioning.

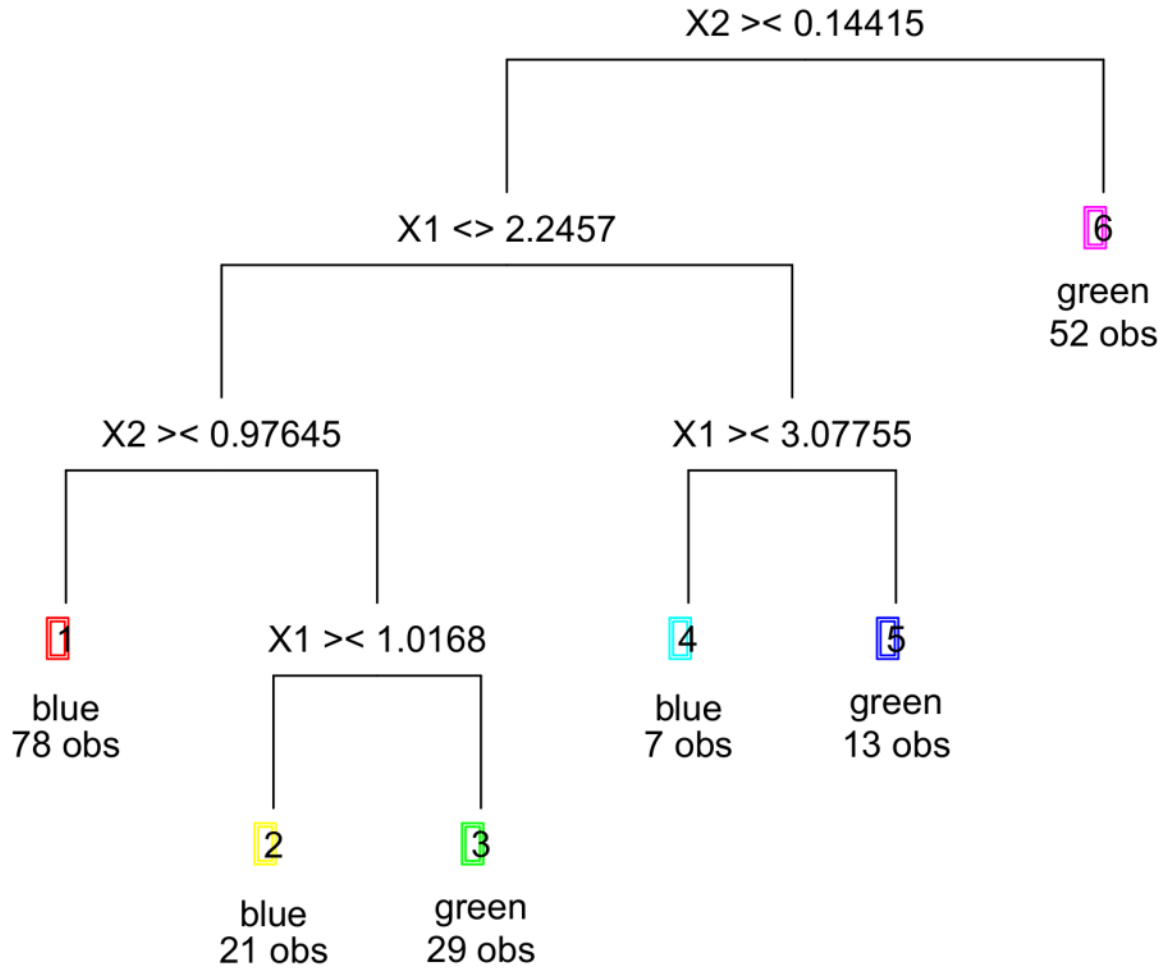
- At each step of a partitioning algorithm, the tree will find the variable (e.g., AU6) and the split-point (e.g., 0.6) that best separate the response classes (fake versus genuine smiles) into rectangular regions.

- Decision trees can be used both for categorical response variables (classification) and continuous response variables (regression).

Trees for the two-class data problem



Trees for the two-class data problem



Trees in R

- The standard package for trees in R is `rpart`. In addition, the `maptree` package is convenient for plotting:

```
library(rpart)
library(maptree)

tree <- rpart(color~X1+X2,data=training)
draw.tree(tree)
```

- As before, we can generate predictions, generate a confusion table, and calculate the error:

```
predictions <- predict(tree,newdata=validation,type="class")
table(predictions,validation$color)
mean(predictions!=validation$color)
> 0.1341142
```

Stopping rules

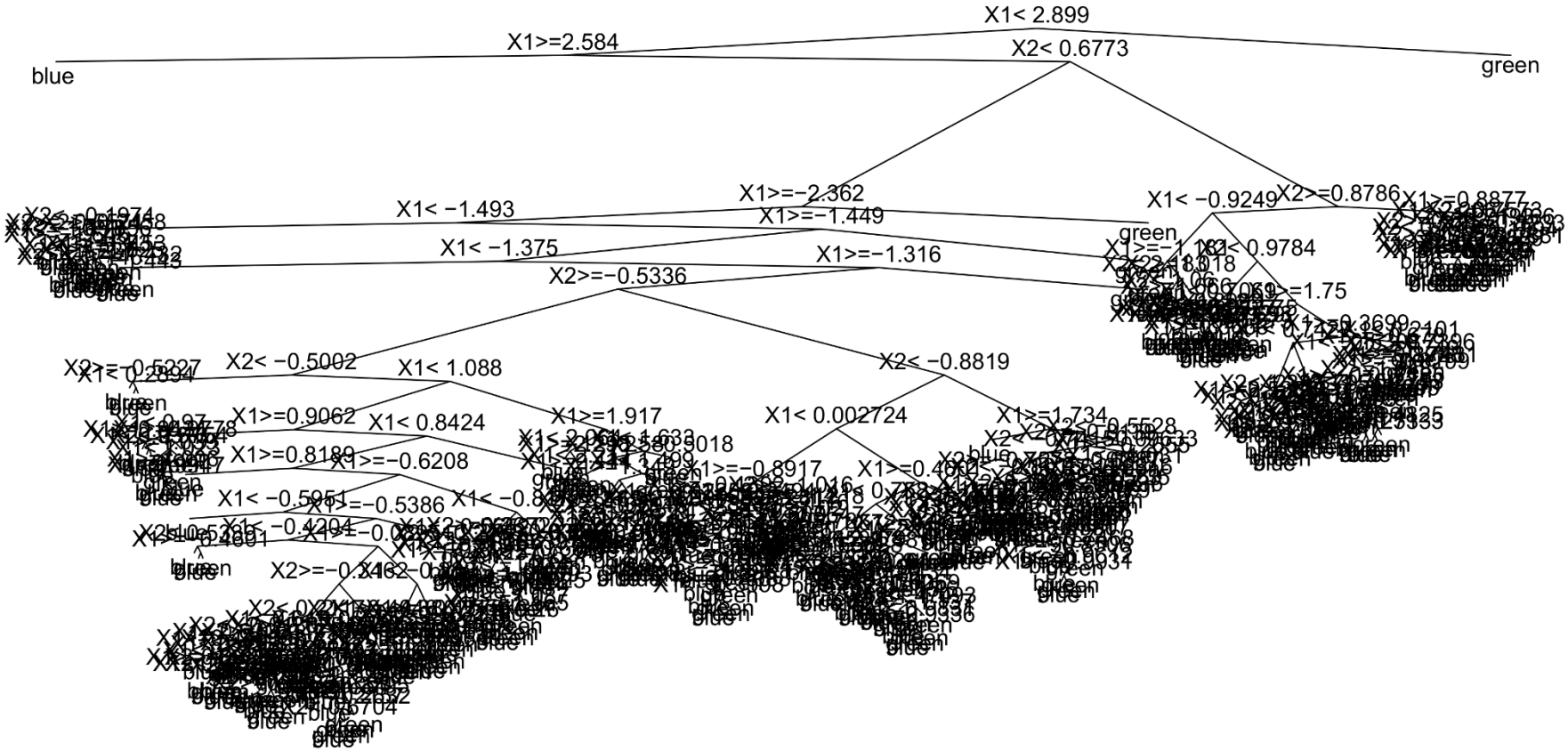
- For the smile example it was easy to decide when to stop splitting the tree: at some point all leaves of the tree were pure (i.e., containing observations of only one class). For other data this may not be so easy. A leaf could contain mixes of different classes. Is it worthwhile to keep splitting in this case?

- A number of rules exist to guide this decision:

  - Split until all leaves are pure
  - A minimum number of observations must remain in a leaf before a new split is attempted (e.g., 10 observations)
  - Split until a maximal depth of the tree is reached
  - Split until the degree of class-separation falls below a certain threshold (cost-complexity criterion)

- In the extreme case, it would be possible to keep splitting until every observation in the data was in its own leaf…

Stopping rules



Overfitting

- Most data are assumed to contain systematic patterns + random noise. A good model should be able to find the patterns, and *not* confuse noise for a pattern.
- A model that treats noise as patterns is said to be overfitting the data.
- Overfitting turns out to be a major problem for many nonlinear models. Because of their flexibility to adapt to complex data patterns, many would be able to adapt perfectly to each individual data point.
- Although this would yield perfect fit to the data the model was trained on, the model would likely generalize very poorly to independent data that it had not "seen".
- This is the reason why ML makes the distinction between training and validation data, and why cross-validation is an important aspect in optimizing a model.

Tuning parameter: CP

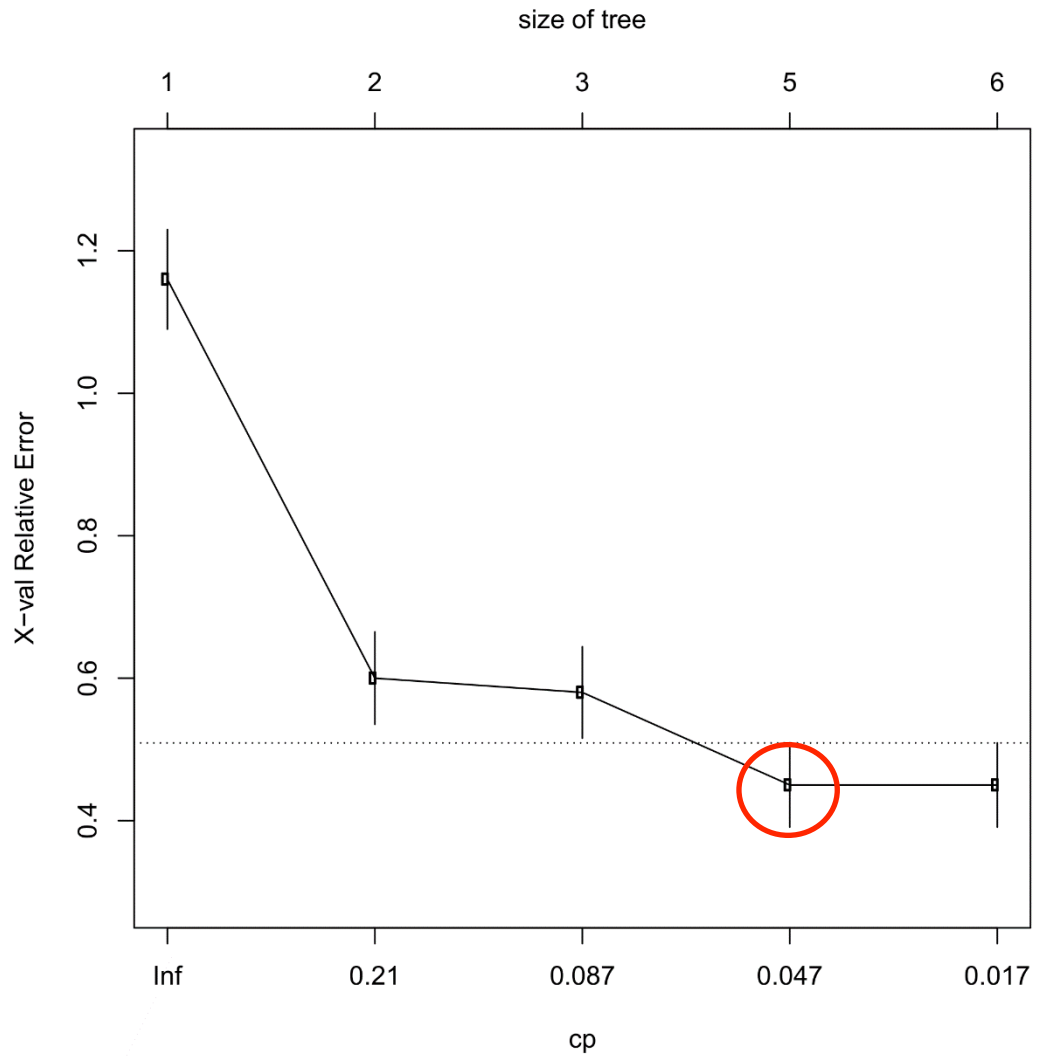
- The most useful control parameter in a decision tree is the cost-complexity parameter (CP). This parameter sets a threshold on the goodness-of-fit increase that is considered meaningful for a new split to be added to the tree.

- CP can be considered a penalty on the number of split points. Larger values lead to less complex trees (strong penalty) while smaller values lead to more complex trees

- In R, the `rpart` package comes with a plotting function that automatically estimates a good cutoff for CP, meaning that manual cross-validation is unnecessary.

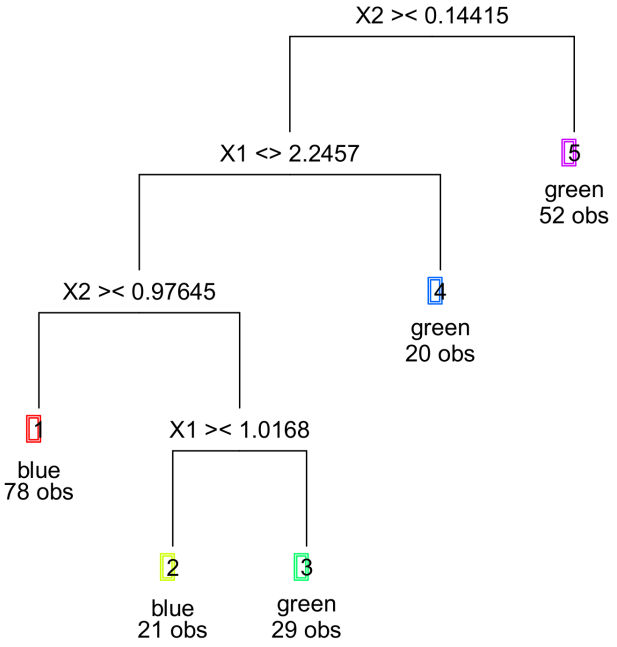
```
tree <- rpart(color~X1+X2,data=training)
plotcp(tree)
```

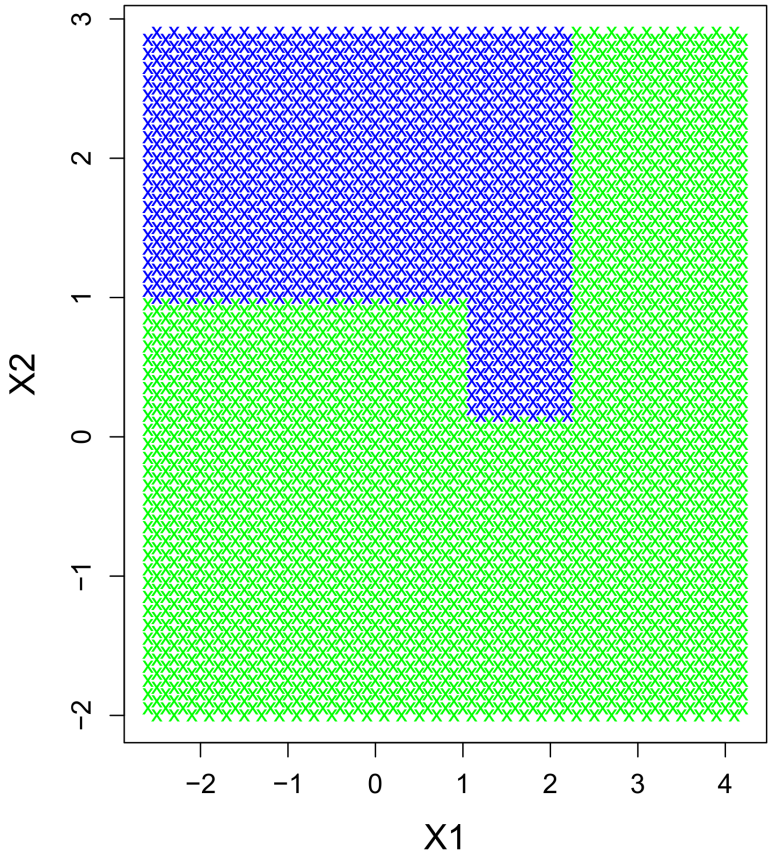
Tuning parameter: CP



Tuning parameter: CP

```
tree <- rpart(color~X1+X2,data=training,control=list(cp=0.047))
```





Feature relevance in trees

- Feature relevance in a decision tree is simple. Any variable from the feature set that was selected for the tree construction is relevant for predicting the response classes (≈ stepwise linear regression).

- Conversely, features that do not show up in the final decision tree are irrelevant for prediction.*

- For the selected features, more detailed rankings of importance can be calculated:
  1. By inspecting the hierarchy of the feature split points (e.g., the primary split point is usually the most important)
  2. By calculating the relative increase in class-separation that each feature split point achieves

- However, since decision trees are highly interactive/nonlinear models, it should be noted that measures of variable importance are also nonlinear!

* Or they may be correlated with a better split variable

Nonlinearity of trees

- As we have seen, a characteristic property of decision trees is that they divide the feature space into rectangular sections.

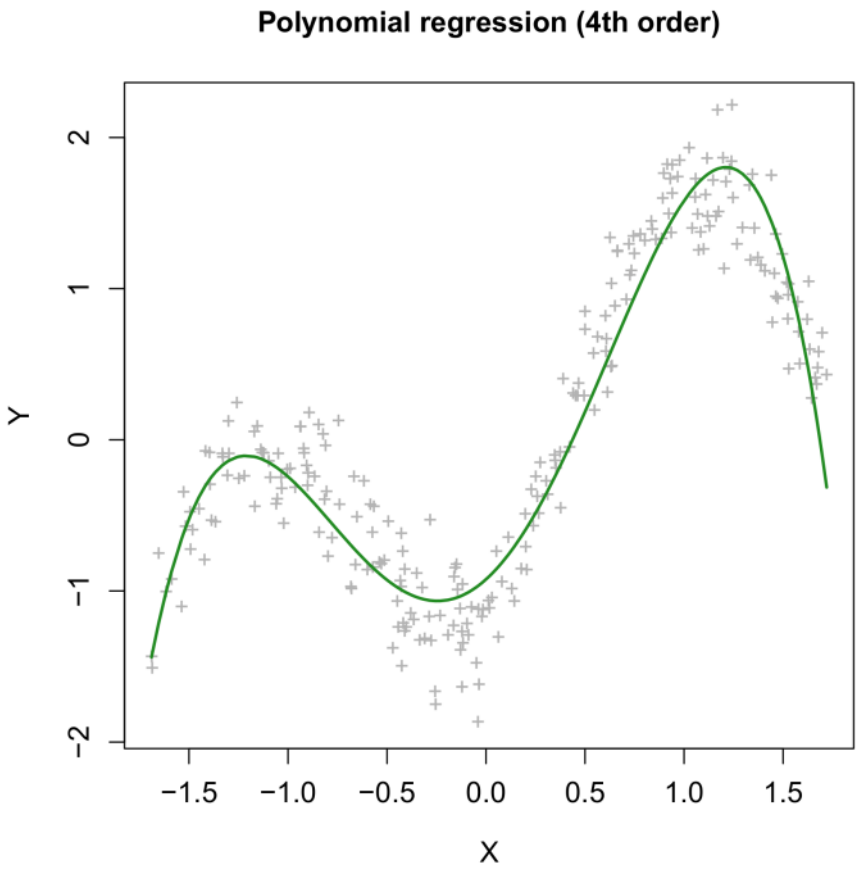
- The model is heavily *interactive* in that each split point further down the tree depends upon a hierarchy of past split points. In fact, a decision tree contains only one genuine main effect, which is the top level split point. Every split rule after that interacts with a previous split.

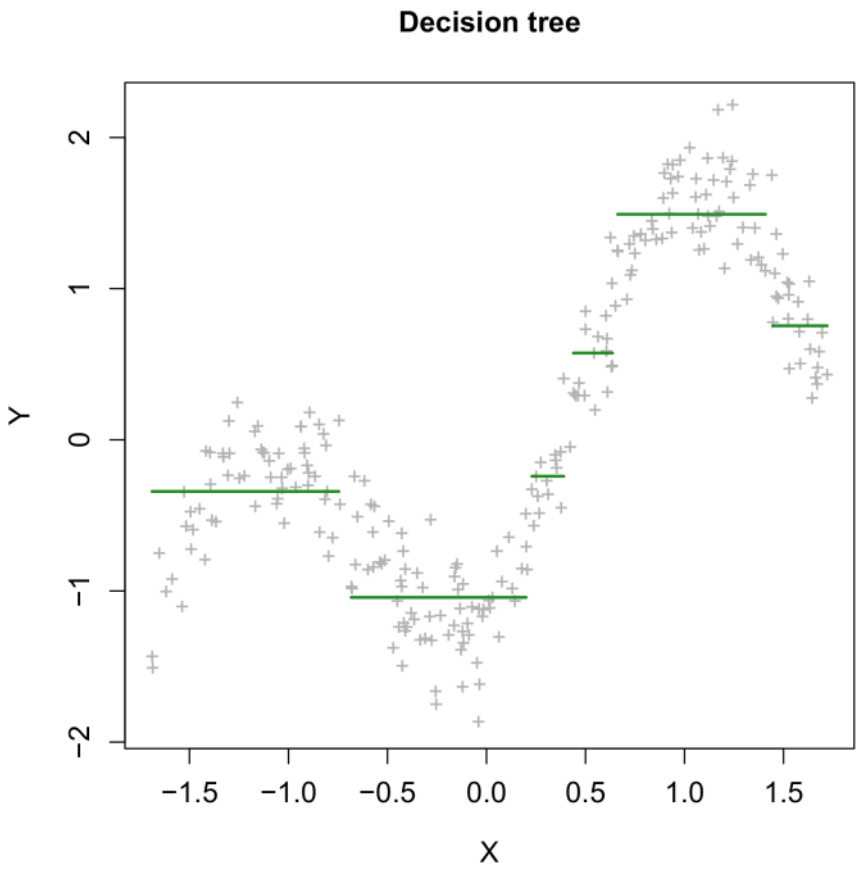
- Trees have some paradoxical properties. On the one hand they can adapt to nonlinear patterns by modelling complex interactions. On the other hand they are limited to model constant values as their output.

- For this reason decision trees have particular difficulty to adapt to basic linear patterns or smooth curves…

Nonlinearity of trees

- One could say that a decision tree model is piecewise constant:





Why should I use decision trees?

- Decision trees are among the simplest statistical models, due to their logic-type format of representation. To a layperson, a decision tree is more understandable than a regression model. This would make it an obvious choice for a basic analysis.

- On the other hand, complex trees are highly nonlinear. Most data are unlikely to contain, e.g., 6th degree interactions.

- Decision trees have sometimes been used as a follow-up to a classical MANOVA analysis (i.e., categorical predictor, multivariate dependents). When the test comes up significant, the tree is used to find out how the groups differ. However, linear discriminant analysis (LDA) should be considered as the more logical choice of follow-up since its distributional assumptions are identical to MANOVA.

- In general, decision trees are best employed as tools for exploratory analysis (e.g., automatic detection of interactions).

Decision trees evaluation

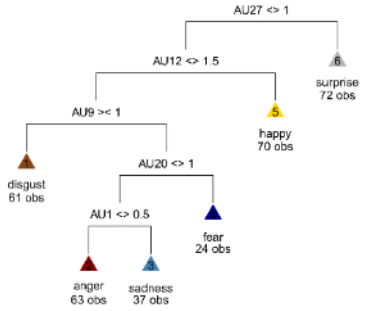
- Advantages:
  - \+ Tree structure is intuitive and highly interpretable
  - \+ Hard selection/elimination of features
  - \+ Well-suited to detecting interaction patterns among features
  - \+ Can handle both continuous and categorical features
  - \+ Can handle missing data
- Disadvantages
  - – Cannot handle simple additive data patterns (such as in ordinary linear regression)
  - – Hierarchical structure makes the model too nonlinear
  - – Poor model for smooth curves
  - – Instability of split points at higher interaction depth
- However, it turns out that some of these weaknesses can be exploited to create a better model…

6. Random forest

Instability of trees

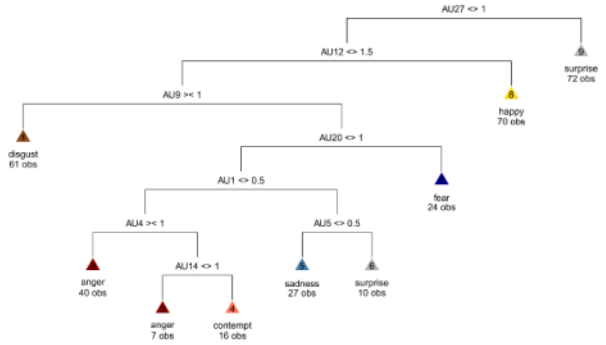
- Decision trees tend to be unstable when too complex. Small perturbations in the data (e.g., different sample, different noise) can lead to very different split points at high interaction levels, and thus different predictions in the leaves of the tree.

- If we had more than one data sample available we could fit multiple trees and compare their structure or predictions to see how bad this problem is:



Sample 1

Sample 2



Ensemble learning

- In a group of decision makers, it is often true that the average decision achieves a better performance than any individual member's decision (e.g., betting on sports results). We have seen this phenomenon before with KNN. Majority voting over multiple neighbors typically performs better than having just 1 neighbor.

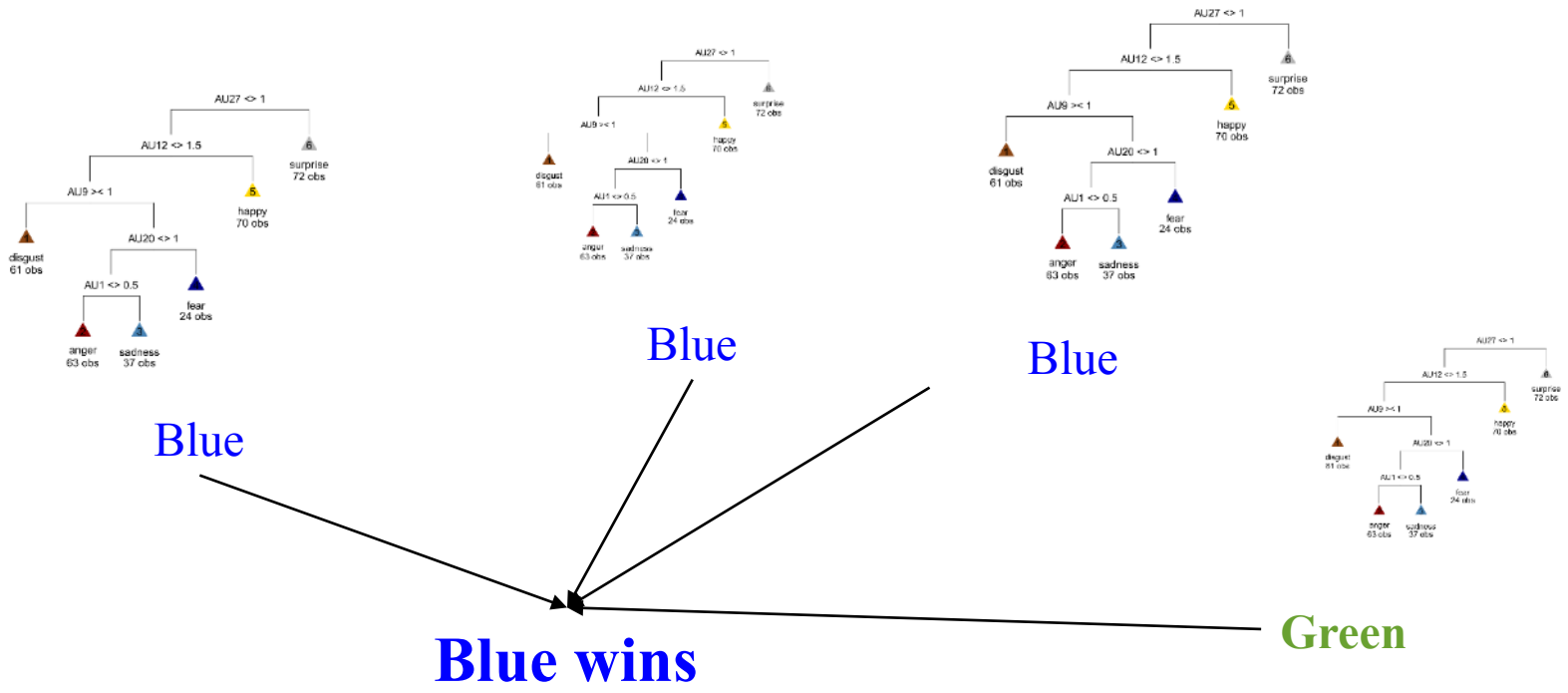
- In machine learning, this principle is more generally known as ensemble learning. That is, the average prediction of multiple competing models (for the same data) will often outperform the prediction of any individual model.

- The latter is especially true for models that are unstable due to high nonlinearity or sensitive optimization algorithms, such as decision trees and neural networks.

- The average prediction of a group of decision trees will almost always outperform an individual tree in the group. This is because each individual tree's instabilities are smoothed out by the aggregation of predictions.

Decision tree committee

- In machine learning, an ensemble of models that generate predictions together is often called a committee. When the response variable is categorical, the aggregated prediction of the committee usually occurs by majority vote:



Bootstrap aggregation

- But how do we obtain all the data that is needed to fit multiple trees…? Collecting data is often costly and we may not have enough observations to create a sufficient amount of subsamples…

- I previously noted that decision trees are sensitive to perturbations in the data. Such perturbations can be simulated by bootstrapping. Bootstrapping a data set means to resample the data with replacement. All cases are put into a bag and drawn at random (with replacement) until we reach a sample equal to the size of the original one.

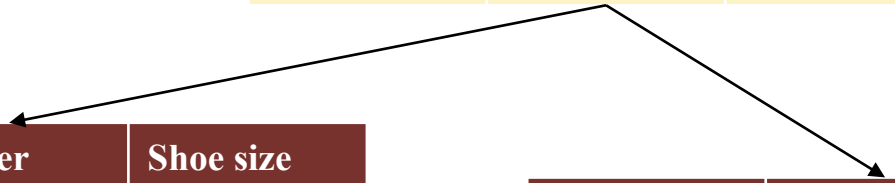
- This process simulates natural variation from bootstrap sample to bootstrap sample, and gives a approximate idea of the variability of certain estimates (e.g., averages, parameters, predictions).

- Ensemble learning that is derived from bootstrapped data sets and models is called bootstrap aggregation.

Bootstrap aggregation

Original sample

ID	Gender	Shoe size
1	Male	42
2	Female	39
3	Female	37
4	Male	43



ID	Gender	Shoe size
4	Male	43
2	Female	39
4	Male	43
4	Male	43

Bootstrap sample 1

ID	Gender	Shoe size
3	Female	37
1	Male	42
2	Female	39
2	Female	39

Bootstrap sample 2

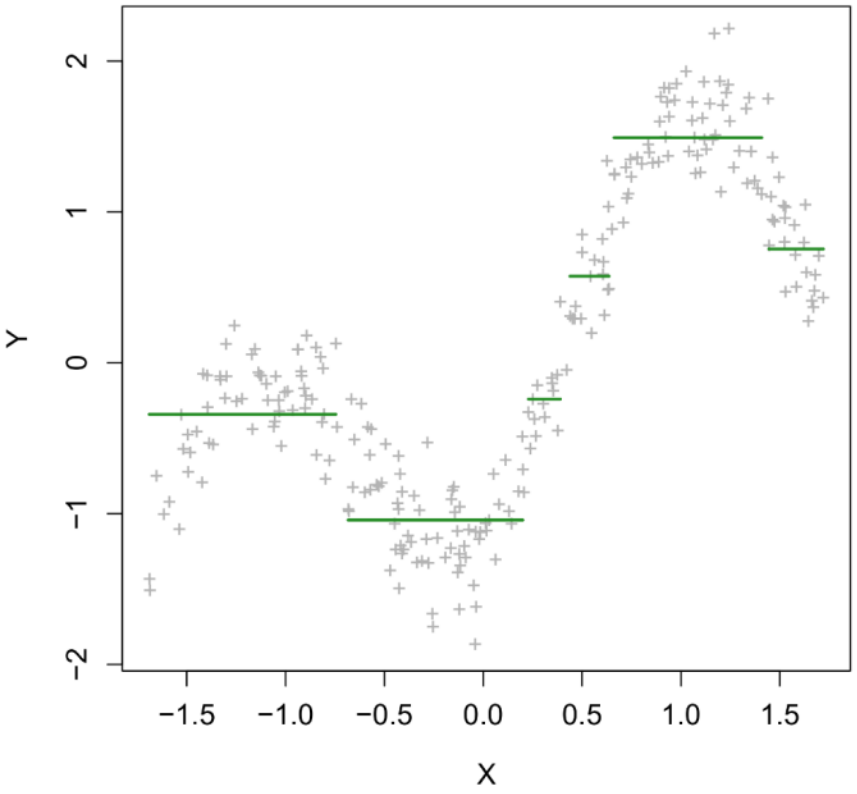
Bootstrap aggregation

- Original data
- Bootstrap data 1
- Bootstrap data 2
- …
- Bootstrap data *B*

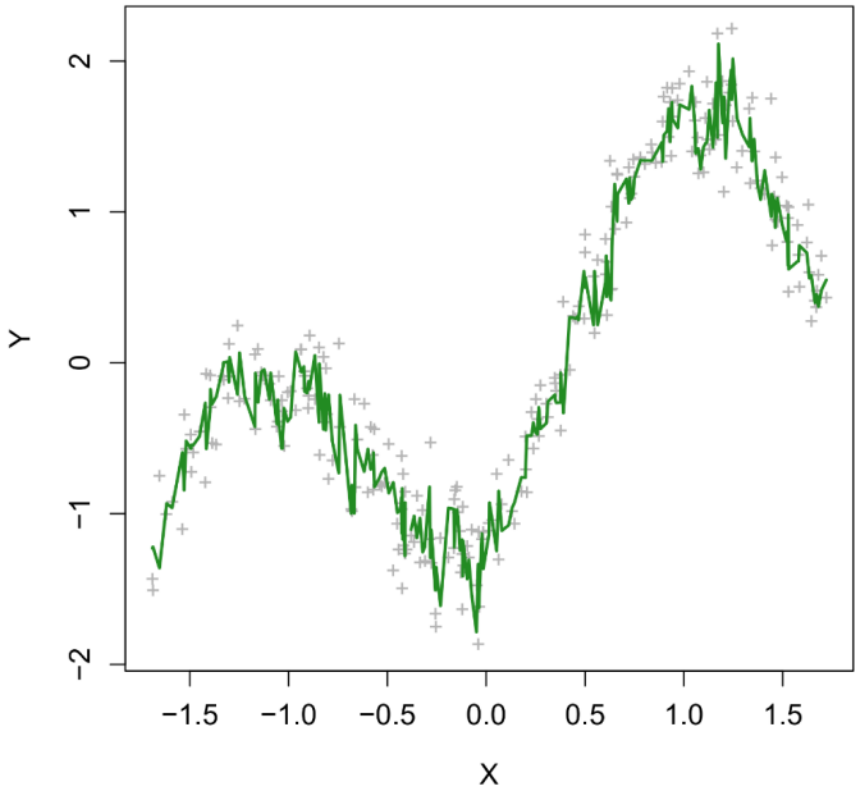
Final averaged prediction

Bootstrap aggregation

One decision tree

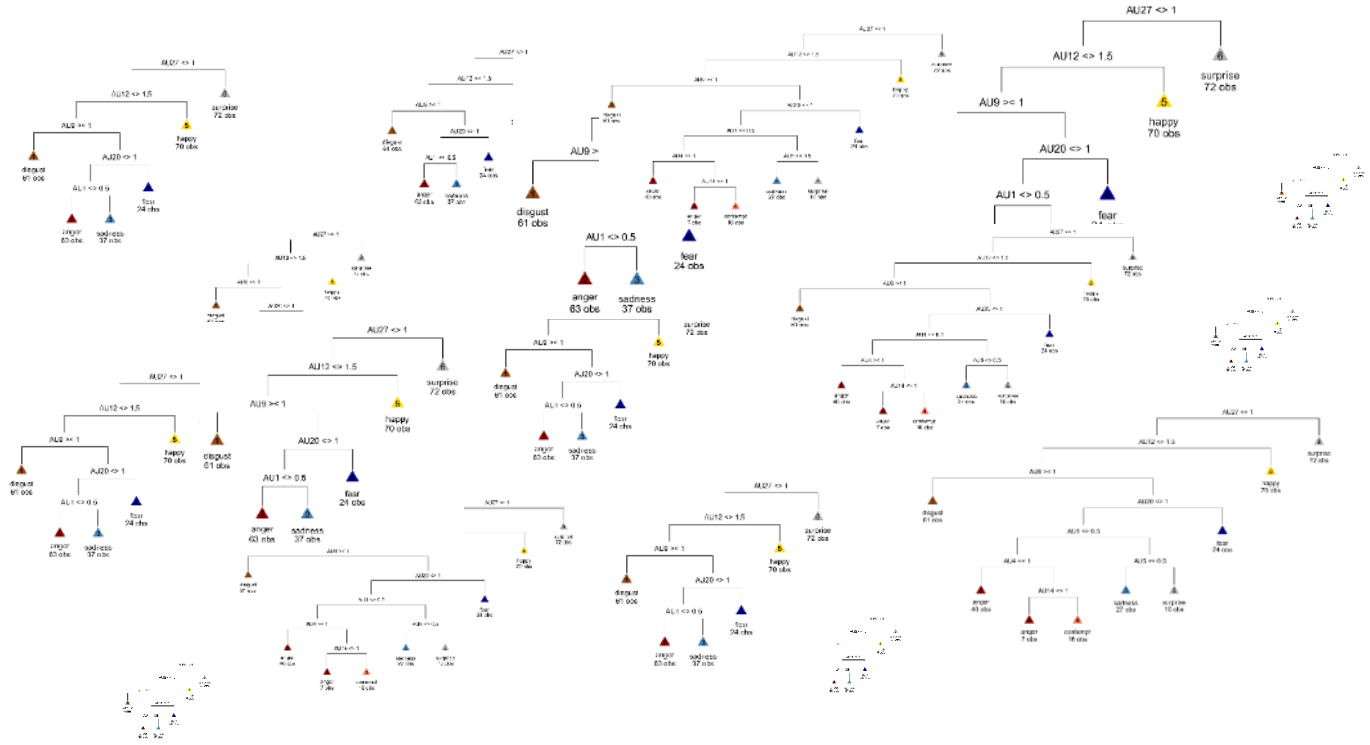


Ten decision trees



Random forest

- The random forest model (Breiman, 2001) uses bootstrap aggregation of decision trees for prediction. Typically, the number of trees in a forest numbers into the hundreds or even thousands.



Random forest

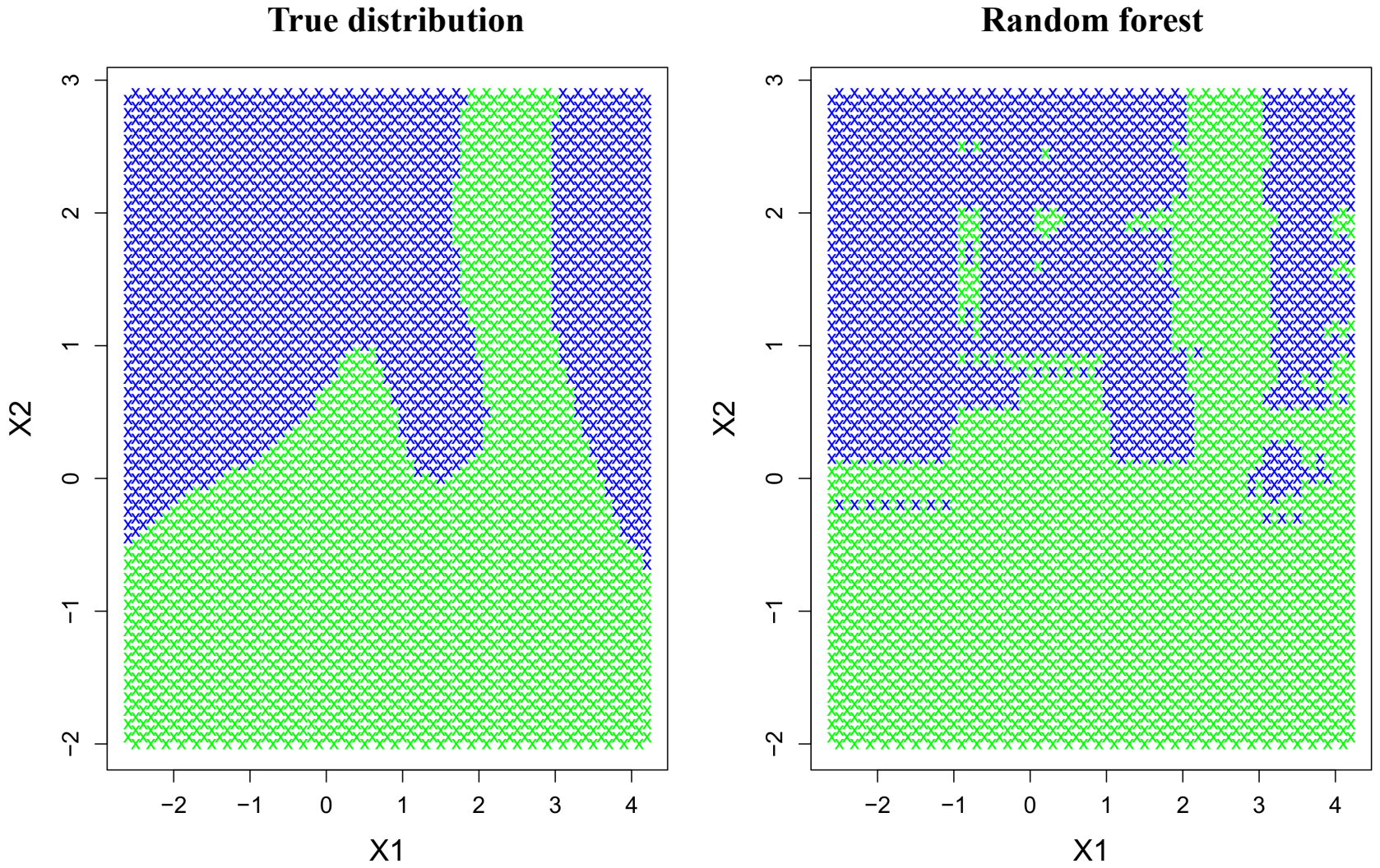
- For each split point in an individual tree, the random forest also randomizes the subset of features that it uses to select the split points. This ensures that different features have an equal chance of being considered for prediction.

- The random forest performs extremely well on many data sets and on many nonlinear problems, sometimes competing with far more sophisticated models such as deep neural networks or complicated support vector machines.

- As of today, it remains one of the best off-the-shelf methods of machine learning. Fitting the model requires the optimization of only one tuning parameter, which is the number of trees in the forest, although the exact number does not seem to affect model fit dramatically. Diagnostic plots can help guide this decision.

- Random forests have been developed for regression, classification, survival analysis, quantile regression, unsupervised clustering, and multivariate prediction.

Random forest for two-class data problem



Random forest for two-class data problem

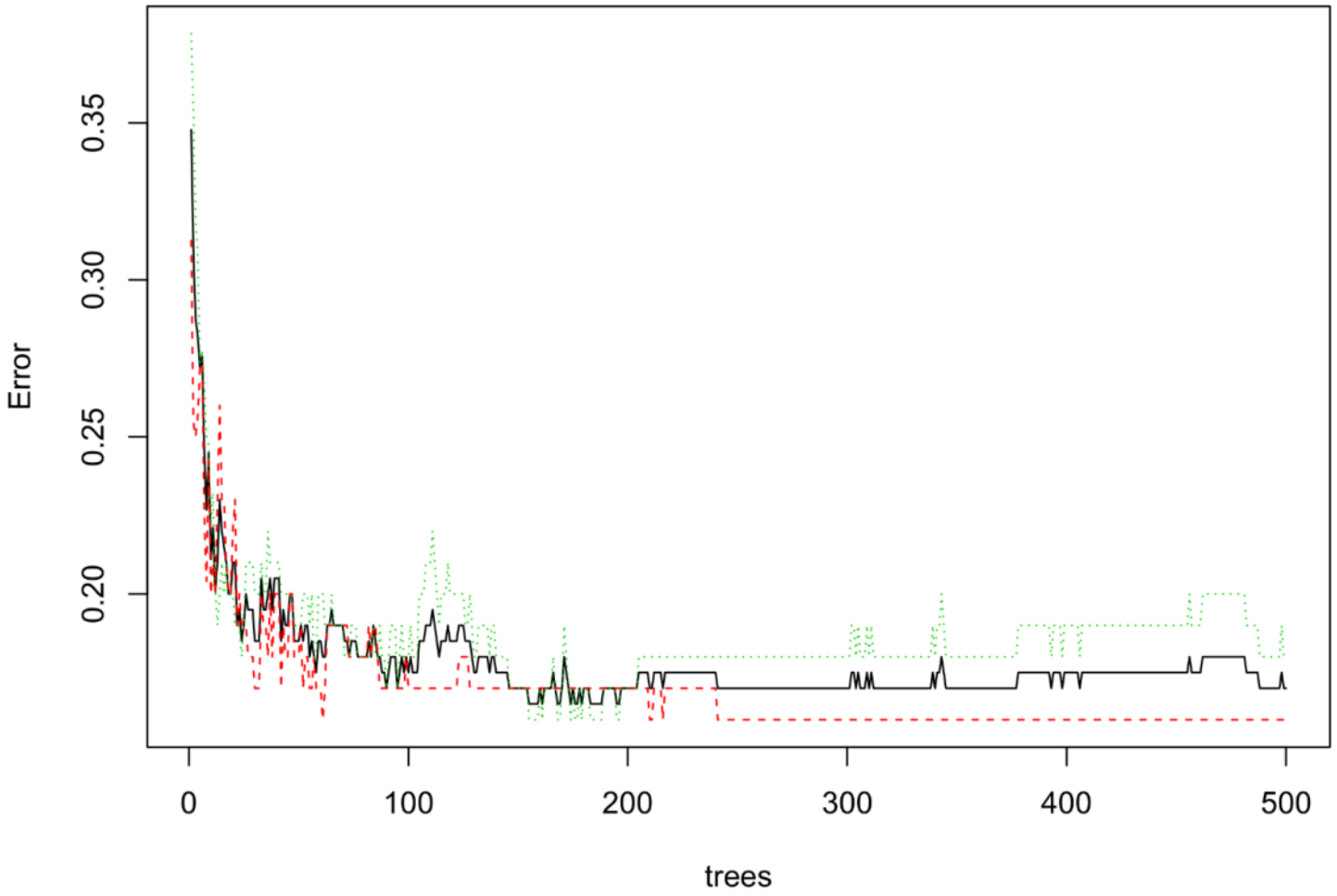
- The approximation looks a bit noisy but otherwise the random forest achieves a very low error for this problem, 9.7% (missclassification). Almost as good as the 8-nearest neighbors model.

- Due to the popularity of the `randomForest` package it has recently become part of R's core packages that are pre-installed with the program. Fitting a random forest proceeds as follows, e.g., for the two-class problem:

```
library(randomForest)
set.seed(88661)
rf <- randomForest(color~X1+X2,data=training,ntree=5000)
plot(rf)
rf
mean(predict(rf,newdata=validation,type="class")!=validation$color)
> 0.09604685
```

- The package includes a diagnostic plot that allows the user the choose the optimal number of trees (*ntree* parameter). It seems that having around 200 trees for this problem is sufficient. Afterwards the fit stabilizes:

Random forest for two-class data problem



Is it a black box?

- It is not possible to represent a random forest as a single set of decision rules. Although we can average the predictions of the trees, averaging their decision structures would not be appropriate.

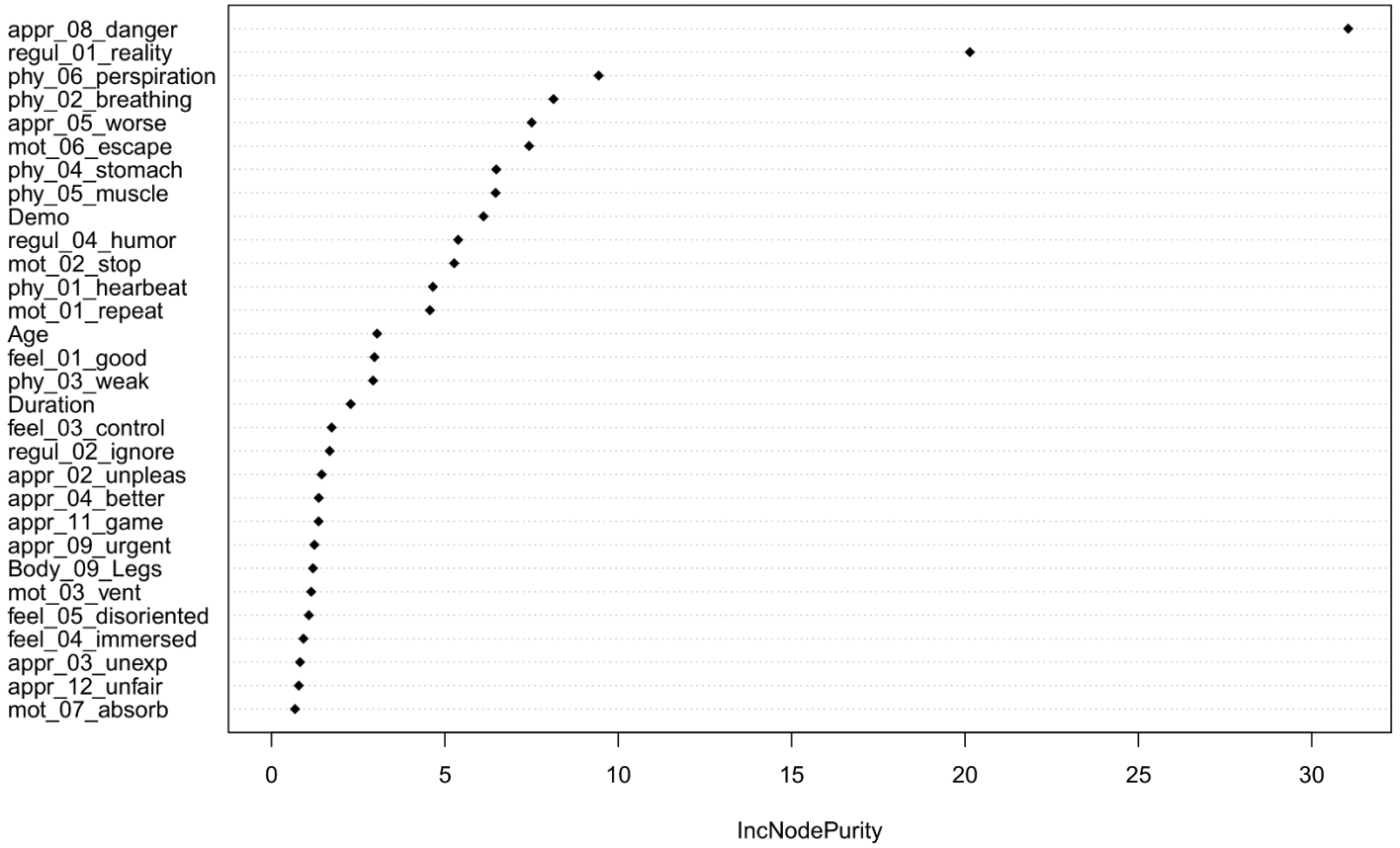
- Individual trees in the forest could be inspected, in theory, but when the forest counts between 200–5000 trees potentially this is not practical.

- This makes the random forest somewhat of a black box. Its internal structure cannot be interpreted. For researchers interested in explanation—which social scientists usually are—this is undesirable.

- However, information about effects could still be obtained by variable importance rankings…

Feature relevance in random forest

- Measures of nonlinear variable importance are based on how often a predictor shows up in the individual decision trees, and their relative success at predicting the target data.



Why should I use random forest?

- Despite its popularity in machine learning, the random forest is not widely known in the social sciences. It has gained recognition in neurosciences, however, where it is used for the analysis of high-dimensional brain data (e.g., multivoxel pattern analysis).

- Given that the model is virtually a black box, it would be hard to motivate the random forest on domain-specific grounds.

- In a paper, it could be used as a general powerful nonlinear machine learner. Non-parametric methods like KNN are also appealing for this purpose but the random forest often outperforms these methods, and has more flexibility.

- Thus, if you seek to compare a linear analysis in a study to a general nonlinear one, the random forest could be a good choice.

Random forest evaluation

- Advantages:
  - \+ Extremely powerful learner
  - \+ Easy to use with practically no complex tuning parameters
  - \+ Can be used for nonlinear clustering (not discussed here)
  - \+ Good general nonlinear benchmark against linear alternatives
- Disadvantages
  - – Has black box characteristics
  - – Simple interpretation of individual decision trees is lost
  - – No hard pruning of irrelevant variables
  - – Some sensitivity to the presence of noise features

7. Concluding remarks

Conclusions

- The amount of information may be overwhelming, but I have only given you a surface overview of machine learning…
- With the two-class data problem we could inspect visually how well our fitted model approximated the true distribution of the blue and green classes.
- In practice we will not have information about the "truth" directly available. This means we must find other ways of quantifying how accuractely our model has predicted the dependent variable (e.g., by cross-validation). Today I have not really discussed this important topic.
- In addition, I have not discussed many other famous models in machine learning, such as discriminants, regularized regression models, generalized additive models, support vector machines, and artificial neural networks.

Conclusions

- Hopefully this presentation showed you that statistical modelling is not limited to linear regression.
- Once you walk outside of this box, it turns out there are other approaches, which are sometimes simpler and more intuitive than linear regresion (KNN, trees).
- Complex ≠ better. For example, nearest neighbors have been very successful for a long time in the area of handwriting recognition!
- In addition, I have introduced to you some more advanced concepts in machine learning:
  - Overfitting
  - Decision boundaries
  - Ensemble learning
  - Bootstrap aggregation

Thank you for your attention