GENERALI

MHPC Master in High Performance Computing SISSA ICTP

Data mining and Machine Learning

Valerio Consorti
mail: valerio.consorti@generali.com

GENERALI

Master in High Performance Computing

Day 2: Categorical variables and classification problems

Cook-book for categorical variables

- Categorical variables are those that express the membership of a record to a specific group or category
  - Male, female
  - Cat, dog, horse, bear, …
- There is not any predefined sorting order between different categories (cat<dog<horse<bear???)
- There is no obvious encoding of a categorical label into a number

- How to use this kind of variables in a model?
- Can they be used together to other categorical and ordinary variables in a model?

Cook-book for categorical variables: encoding

Problem: predict the amount of meat that you need to buy for a pet knowing the type of pet (dog or cat) and the group of age (puppy, young, old)

1. Models normally uses only numerical variable so the first step is to translate these 2 categorical variables into numbers
   - Dummy encoding:

Type	encoding
Dog	0
Cat	1

Age group	encoding
Puppy	0
Young	1
Old	2

$$Amount\ of\ meat = \alpha_0 + \alpha_{type} Type + \alpha_{age} AgeGroup$$

   - Do you think this would ever work?

Cook-book for categorical variables: encoding

Thinking about...

Type	encoding
Dog	0
Cat	1

Age group	encoding
Puppy	0
Young	1
Old	2

$$Amount\ of\ meat = \alpha_0 + \alpha_{type} Type + \alpha_{age} AgeGroup$$

- The reference of the model is the amount of meat eaten by a puppy dog
  - Puppy dogs drink milk → $\alpha_0 = 0$
- Since cats are in average smaller than dogs α_{type} is probably < 0
  - Than puppy cats produces meat (eat a negative amount of meat)
- Depending on the sign of α_{age} either an old pet eat twice as a young one either the older a dog is the more meat it produces
- …

Cook-book for categorical variables: encoding

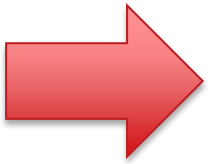
What caused this epic failure?

- There is not any predefined sorting order between different categories!
- In our encoding: dog<cat and puppy<young<old

Better encoding:

1. Transform each category into a different variable
2. Assign 0 if the record doesn't belong to a category and 1 if it does belong

id	Type	Age
1	Dog	Puppy
2	Dog	Old
3	Cat	Young
4	Dog	Young
5	Cat	Puppy



id	Dog	Cat	Puppy	Young	Old
1	1	0	1	0	0
2	1	0	0	0	1
3	0	1	0	1	0
4	1	0	0	1	0
5	0	1	1	0	0

$$Amout\ of\ meat = \alpha_0 + \alpha_{dog} Dog + \alpha_{cat} Cat + \alpha_{puppy} Puppy + \alpha_{young} Young + \alpha_{old} Old$$

Classification problems: categorical target variable

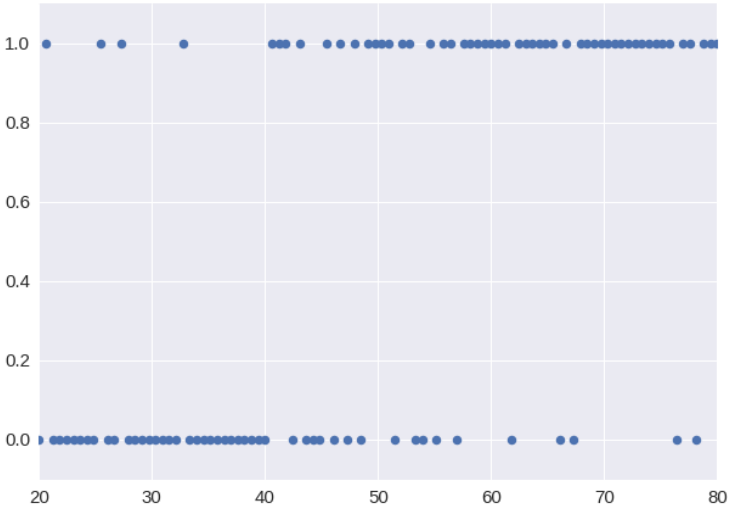
What if the target variable is categorical?

Business problem: predict if a person own an health insurance depending on the age

- Analytical problem: …?
  - Independent variable:
    - Age: continuous ordinal variable
  - Target variable:
    - Dichotomous variable: $y = \begin{cases} 0 & doesn't\ have\ an\ insurance \\ 1 & have\ an\ insurance \end{cases}$
    - The problem is formally analogous to determine if a person of a given age is likely to belong to the group of the policy owners or to the group of people not owning any policy

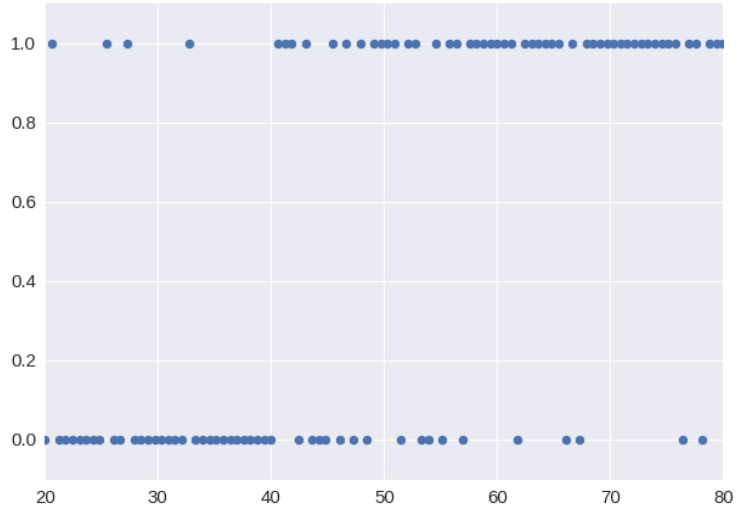
Classification problems: categorical target variable

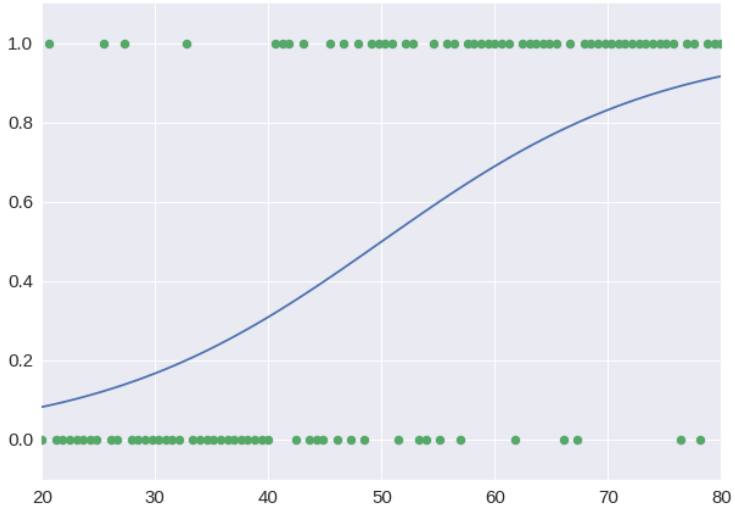
- Target variable:
  - Dichotomous variable: $y = \begin{cases} 0 & doesn't\ have\ an\ insurance \\ 1 & have\ an\ insurance \end{cases}$
  - This variable behaves like a Binomial variable



Classification problems: categorical target variable

- Target variable:
  - Dichotomous variable: $y = \begin{cases} 0 & doesn't\ have\ an\ insurance \\ 1 & have\ an\ insurance \end{cases}$
  - This variable behaves like a Binomial variable

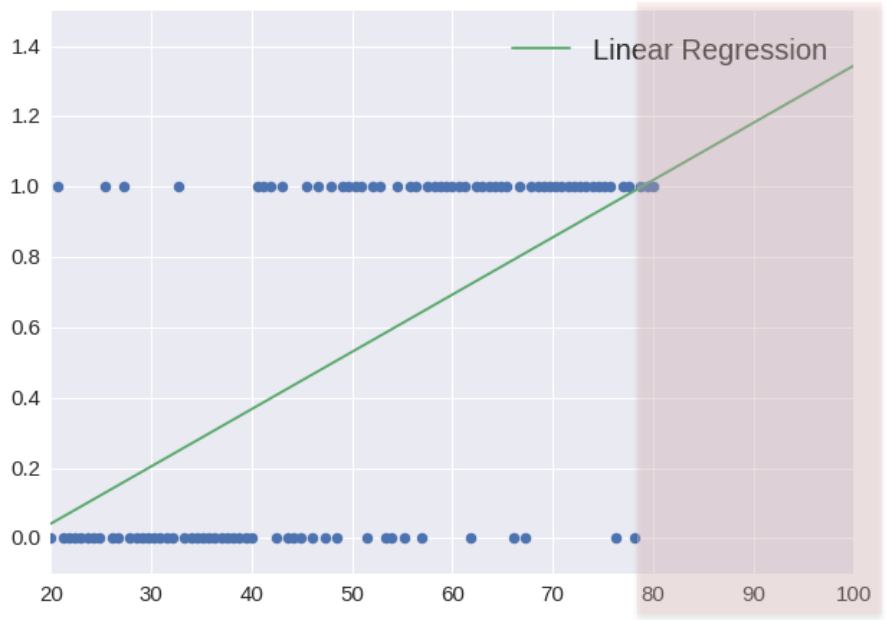




  - The actual target variable to predict than is the probability to have an insurance
  - To predict the probability moves back the classification problem to a regression problem with a continuous target variable which represents the probability to belong to a class

Classification problems: modelling probabilities

- Probabilities are limited between 0 and 1
- Often the p distribution actually is a cumulative distribution along the variable $\int_{x0}^{x1} pdf(x)\,dx$ which typically is a an “S” shaped function
- The linear regression is not really adequate to describe the problem since:
  - It is not inf. nor sup. limited
  - It is not “S” shaped so it can only represent a first order approximation valid in a limited range

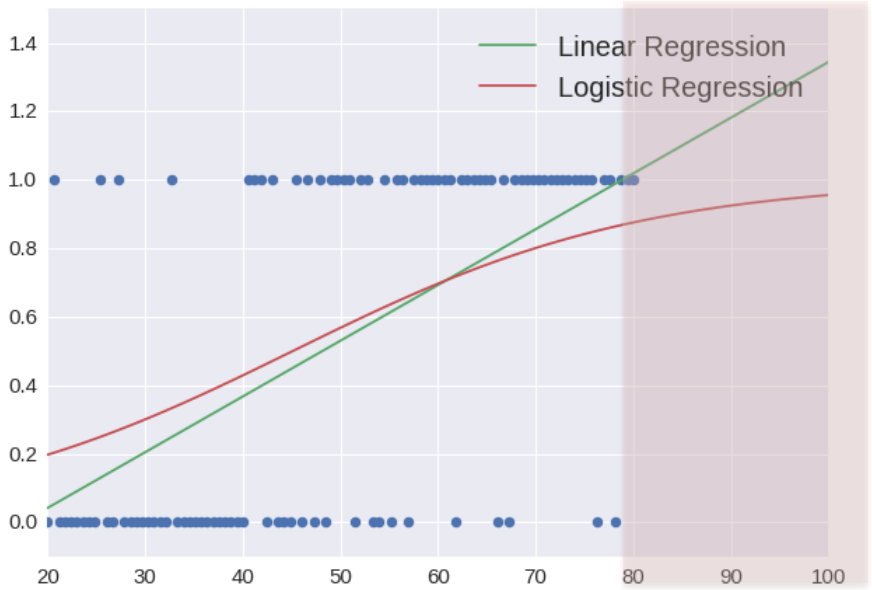


Classification problems: the logistic regression

- A possible good fit kernel function is:

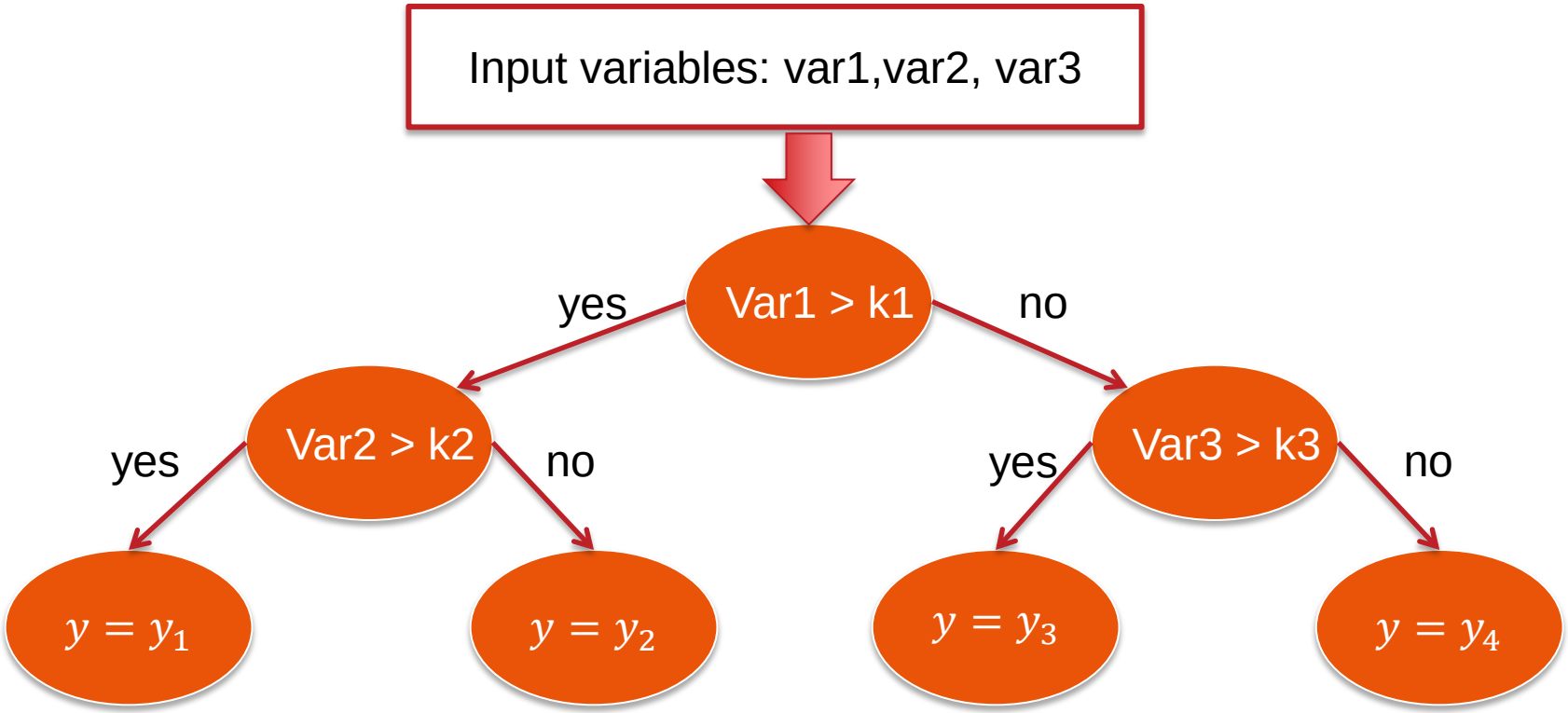
$$f\left(\beta_0, \vec{\beta}, \vec{x}\right) = \frac{e^{\beta_0 + \vec{\beta}\cdot\vec{x}}}{1 + e^{\beta_0 + \vec{\beta}\cdot\vec{x}}}$$

  - It is limited between 0 and 1
  - It is “S” shaped
  - We can interpret $f\left(\beta_0, \vec{\beta}, \vec{x}\right)$ as $p(y|x)$
  - By transforming the target variable from $p(y|x) \rightarrow ln\left(\frac{p}{1-p}\right) = \beta_0 + \vec{\beta}\cdot\vec{x}$ it is possible to approach the problem as a simple linear regression



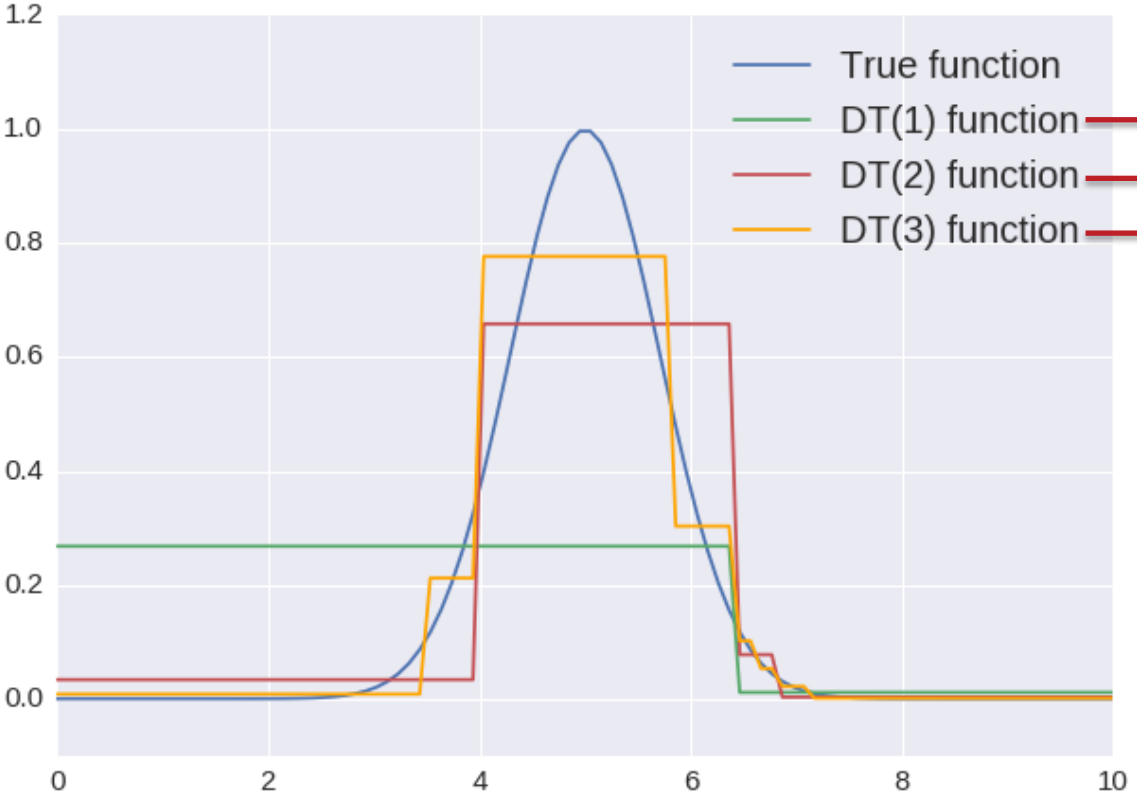
Classification problems: Decision Trees

- Decision tree is a very simple and efficient linear model
- The output depends on the occurrence of a set of condition on the input variables
- It can be used both for regression and classification problems



Classification problems: Decision Trees

- Decision tree algorithm provides an approximation any function which maps the observations to the phenomenon as multi-step function
- The number of steps it can learn depends on the number of the leafs (leafs are the last level nodes)
- The deeper is the tree the more accurate the approximation is

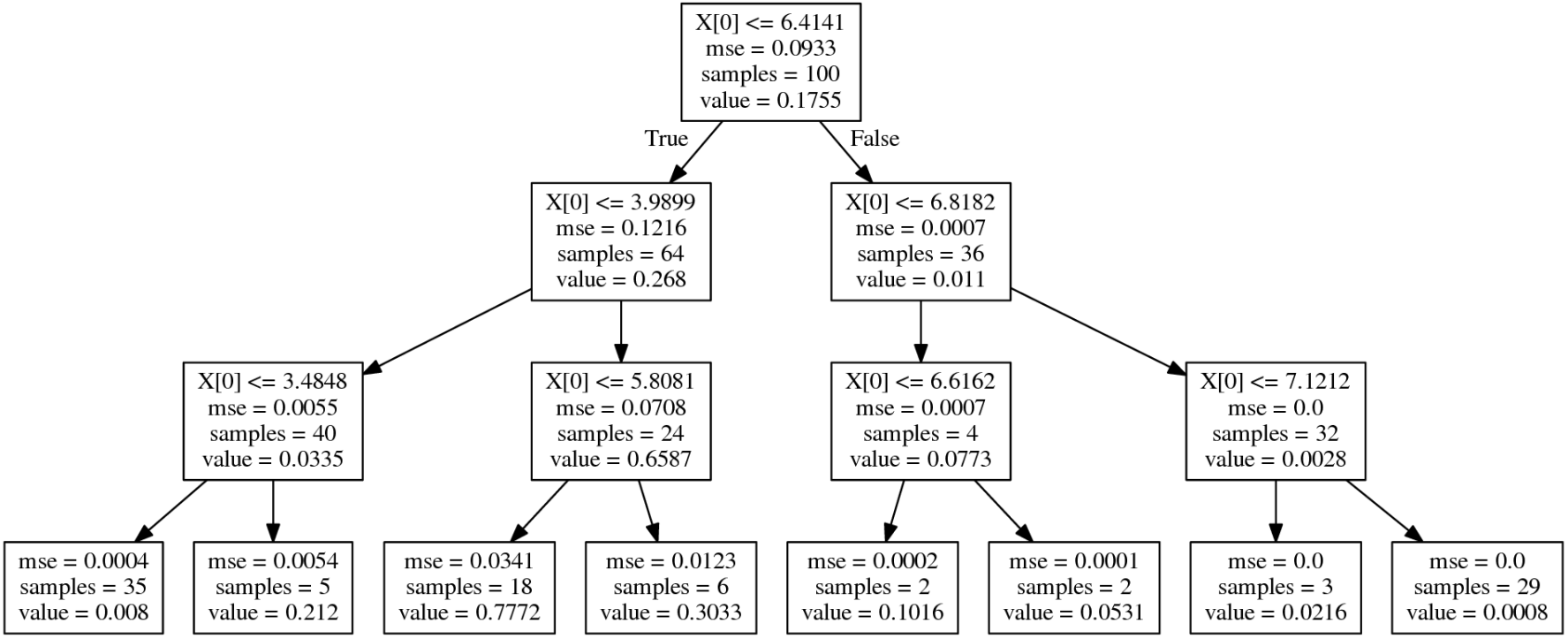


1 lv. split = 2 leafs
2 lv. split = 4 leafs
3 lv. split = 8 leafs
…
n lv. split = 2^n leafs

Classification problems: Decision Trees

Pros:

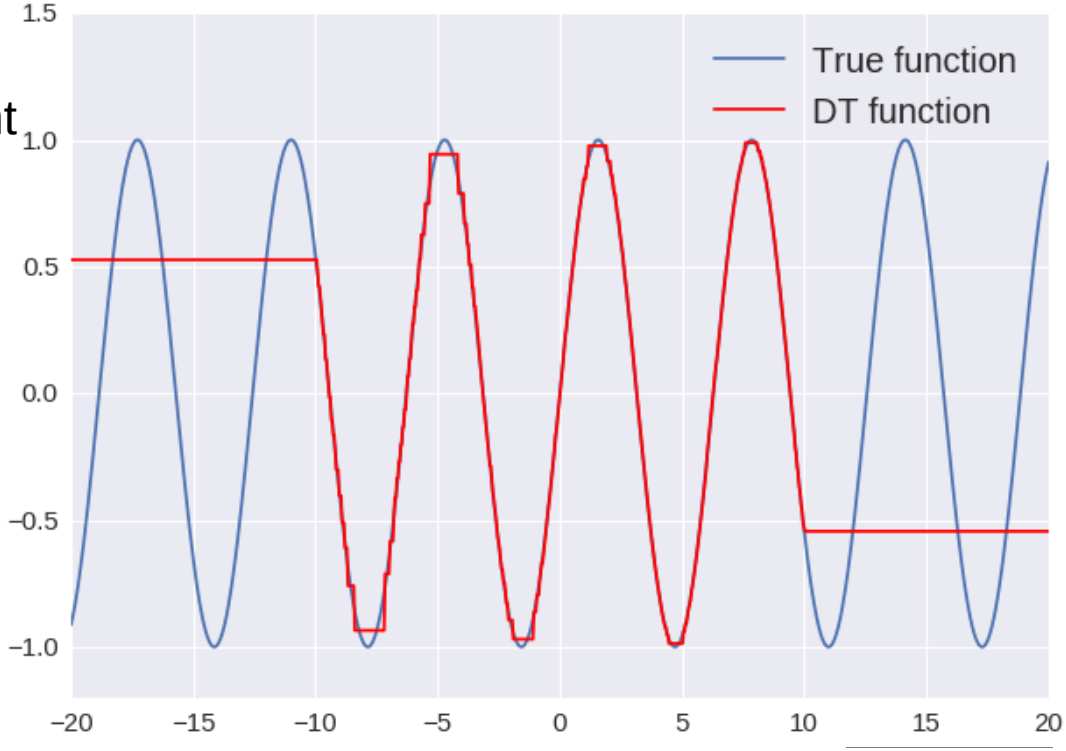
- Very simple and intuitive model
- The tree can be visualized and the phenomenon can be interpreted in terms of the different splits present in the tree
- It can approximate almost any function
- It is a suitable model both for regression and classification



Classification problems: Decision Trees

Cons:

- Very easy to run in overfitting
- In presence of correlated variables the training could fall in a “local” optimal solution not which might be very different from the “global”
- It can describe the behavior of the phenomenon only when the variables are within the range they had in the training set. Anything outside that range will be approximated with the same value (there is no underlying functional form)
- It is not very stable: often small variations in the training set might result in big differences in the structure of the tree



Classification problems: Model selection and quality measurements

- From an algorithmic point of view a classification problem is equivalent to a regression problem where the target variable is the probability of being part of a class

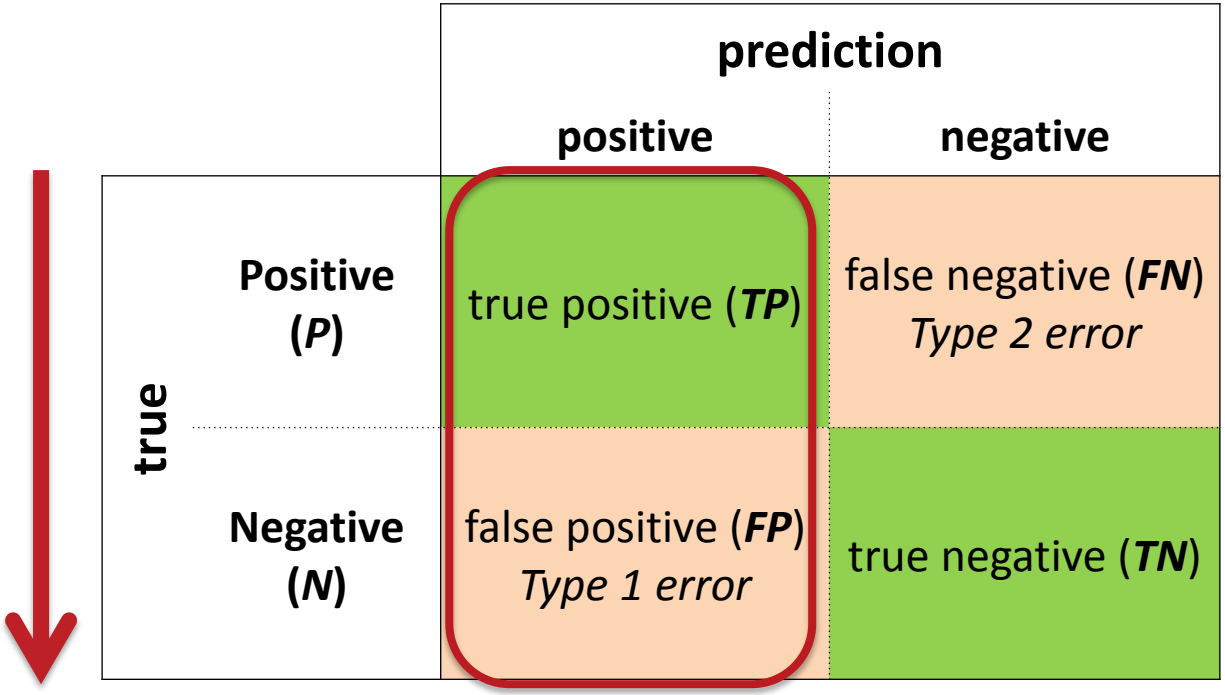
- To use the SS or R^2 to evaluate the performance of the model and to compare different models is senseless
  1. The goal is to associate each event to a specific group not just to describe the probability to belong to the group
  2. The probability *true* function is normally unknown and can be only eventually empirically computed from the training set

- We need to define and develop a different strategy to evaluate the performance of the model

Quality measurements: confusion matrix

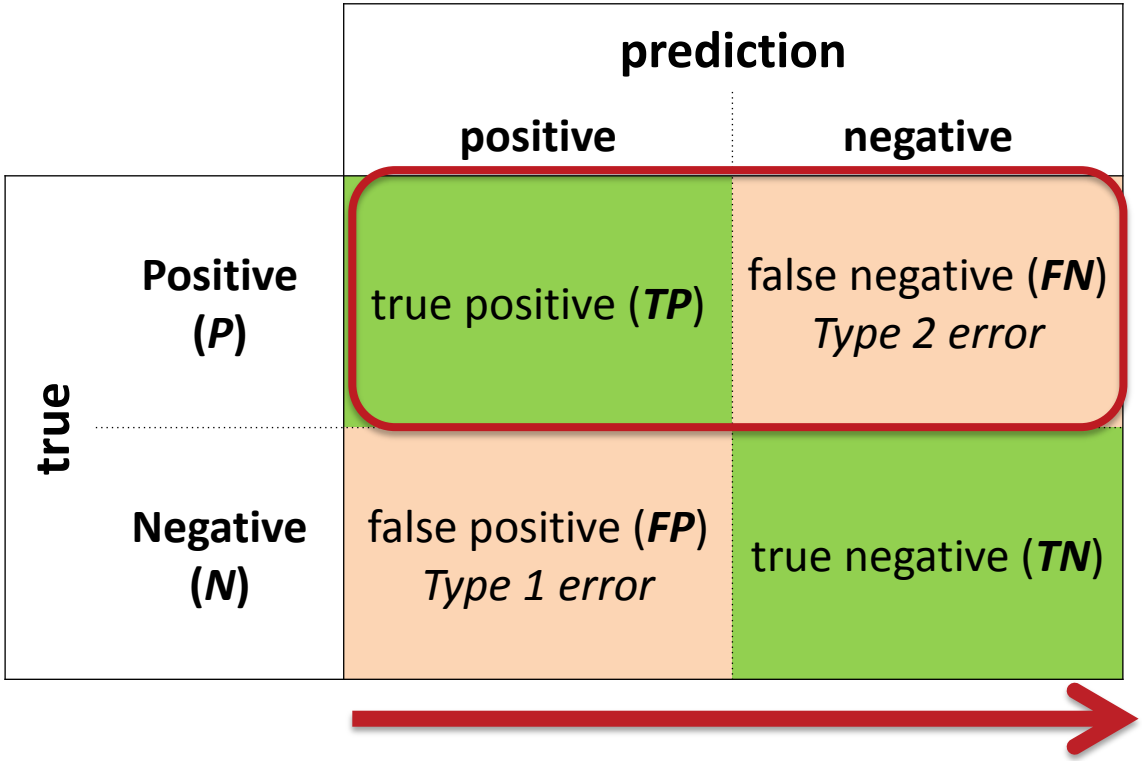
		prediction	
		positive	**negative**
true	**Positive (*P*)**	true positive (***TP***)	false negative (***FN***) *Type 2 error*
	Negative (*N*)	false positive (***FP***) *Type 1 error*	true negative (***TN***)

Quality measurements: precision



$$precision = \frac{TP}{TP+FP} : \qquad P\left(positive_{true} | positive_{predicted}\right)$$

Quality measurements: recall



$$precision = \frac{TP}{TP+FP} : \quad P\left(positive_{true} | positive_{predicted}\right)$$

$$recall = \frac{TP}{TP+FN} : \quad P\left(positive_{predicted} | positive_{true}\right)$$

Quality measurements: accuracy

		prediction: positive	prediction: negative
true	Positive (*P*)	true positive (***TP***)	false negative (***FN***) *Type 2 error*
true	Negative (*N*)	false positive (***FP***) *Type 1 error*	true negative (***TN***)

$$precision = \frac{TP}{TP+FP}: \quad P(positive_{true}|positive_{predicted})$$

$$recall = \frac{TP}{TP+FN}: \quad P(positive_{predicted}|positive_{true})$$

$$accuracy = \frac{TP+TN}{All}: \quad P(correct\ class)$$

Quality measurements: F1

		prediction: positive	prediction: negative
true	Positive (*P*)	true positive (***TP***)	false negative (***FN***) *Type 2 error*
true	Negative (*N*)	false positive (***FP***) *Type 1 error*	true negative (***TN***)

$$precision = \frac{TP}{TP+FP}: \quad P(positive_{true}|positive_{predicted})$$

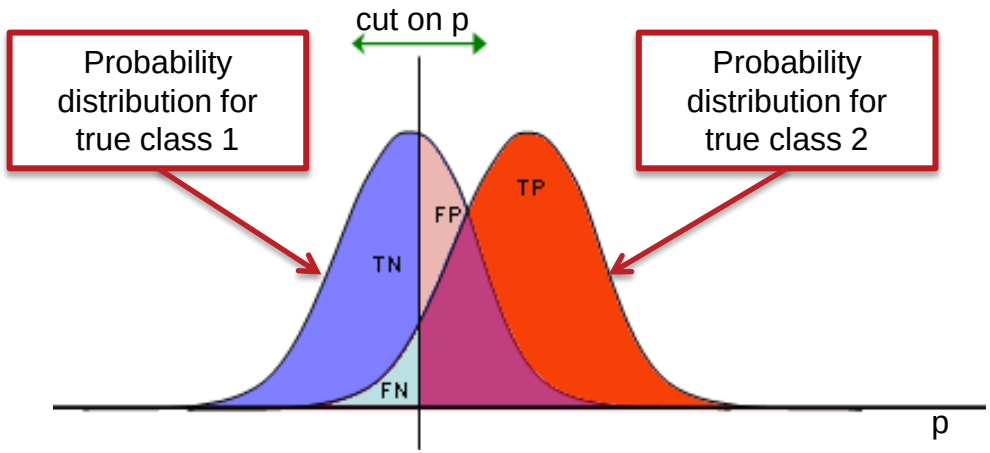
$$recall = \frac{TP}{TP+FN}: \quad P(positive_{predicted}|positive_{true})$$

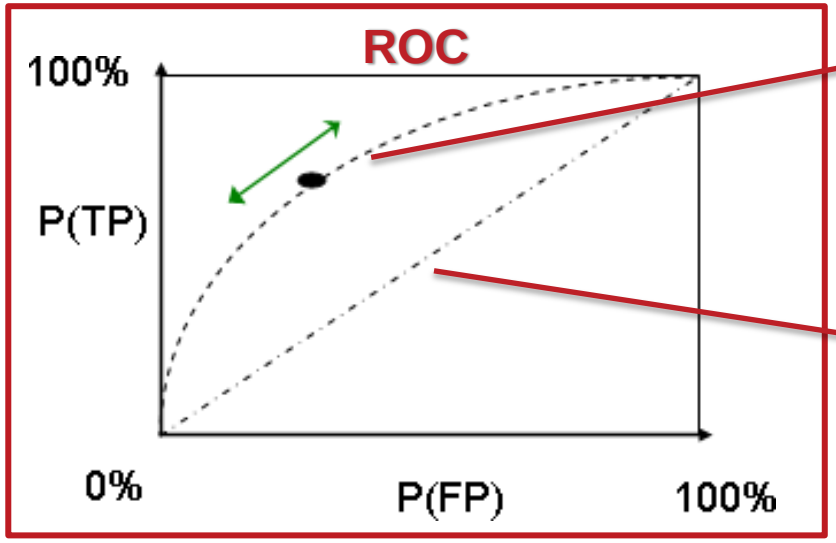
$$accuracy = \frac{TP+TN}{All}: \quad P(correct\ class)$$

$$F_1 = 2\frac{precision \cdot recall}{precision+recall} = \frac{2 \cdot TP}{2 \cdot TP+FP+FN}$$: harmonic average between precision and recall

Quality measurements: Receiver Operating Characteristic (ROC)

- The output of the binary classification is the probability to belong to a class
  - Which is the optimal cut on p?



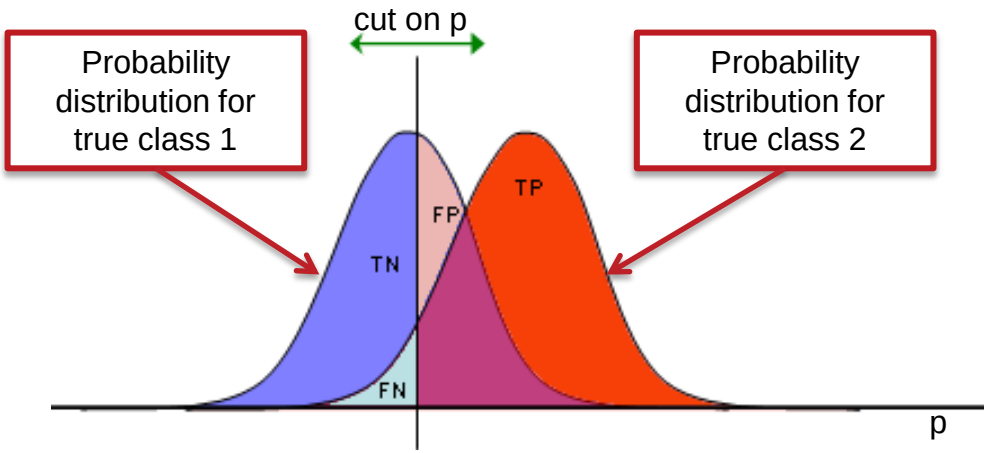


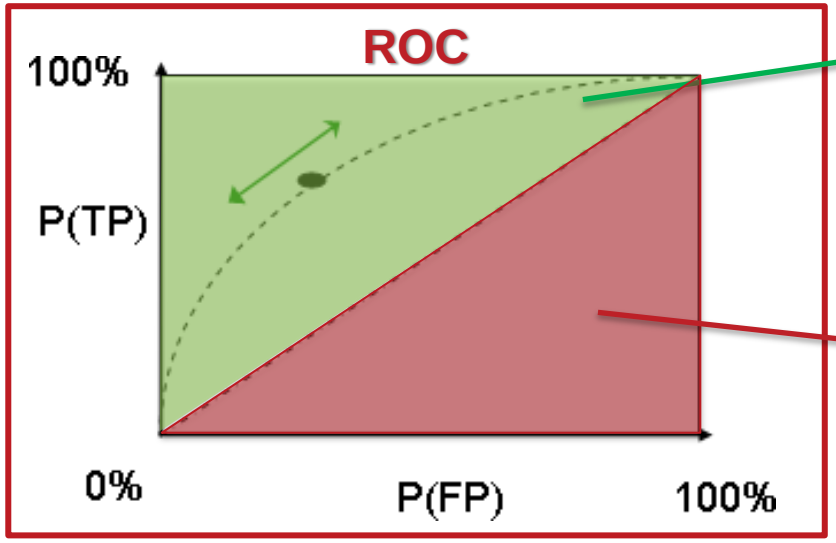
Curve obtained by varying the cut on the output *p of the model*

Random model: extract a random number *n* from a uniform distribution
- if *n*<*p*: class 1
- else: class 2

Quality measurements: Receiver Operating Characteristic (ROC)

- The output of the binary classification is the probability to belong to a class
  - Which is the optimal cut on p?



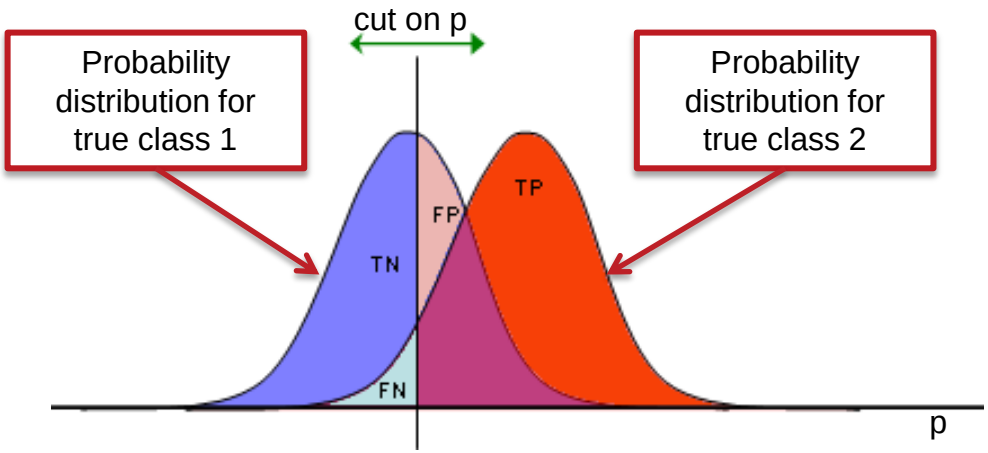


Models whose ROC curve is in the upper triangle are good models

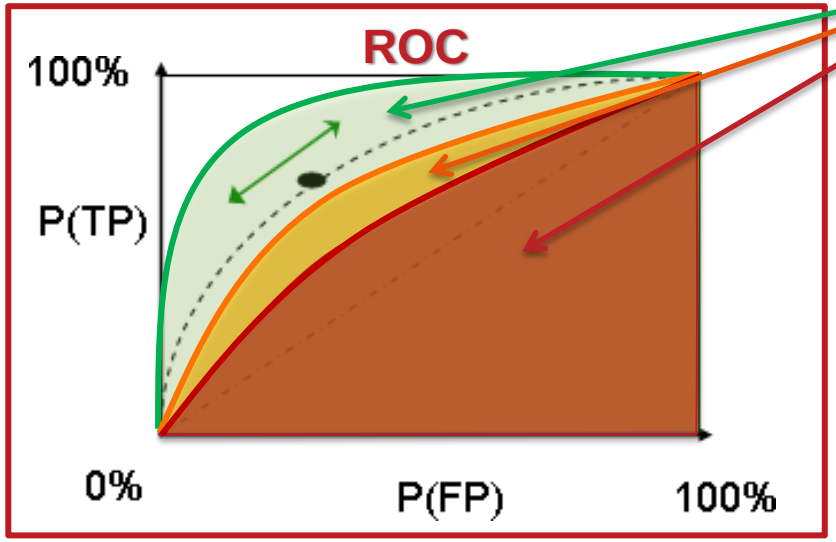
Models whose ROC curve is in the lower triangle are worst than a toss of a coin

Quality measurements: Receiver Operating Characteristic (ROC)

- The output of the binary classification is the probability to belong to a class
  - Which is the optimal cut on p?



The larger is the area under the ROC curve the better is the model



Quality measurements: Lift

$$Lift = \frac{P\left(positive_{predicted} \mid positive_{true}\right)}{P\left(positive_{predicted}\right)} = \frac{\left(\frac{TP}{TP + FN}\right)}{\left(\frac{TP + FP}{All}\right)}$$

- The *lift* represents the expected improvement of the model with respect to a random model

Example problem: A company wants to promote a new product across its existing customers. Due to budget constraints they can sand advertisement only to a fraction of its customers

Solution:

- Implement a model that classify customers in those that are likely to buy the new product and not
- The best model is the one that has the higher lift with respect to a random selection of the customers

Quality measurements: Lift

		prediction	
		positive	negative
true	Positive (*P*)	3	1
	Negative (*N*)	1	2

$$Lift = \frac{P\left(positive_{predicted} \mid positive_{true}\right)}{P\left(positive_{predicted}\right)}$$

- $P\left(positive_{predicted}\right) = 4/7$
- $P\left(positive_{predicted} \mid positive_{true}\right) = 3/4$
- $Lift(positive) = 21/16 \sim 1.31$

Class unbalancing

- Class unbalancing occurs when in a classification problem there is a class that outnumbers the other
  - i.e. in a binary classification problem one class represent the 98% of the overall population and the other the remaining 2%

- Class unbalancing introduces biases in the training
  - the accuracy measurement is biased (an algorithm predicting only the majority class will be however 98% accurate)
  - this will bias the model trained towards the majority class

- The main available methods to deal with this problem are:
  1. Undersampling
  2. Oversampling
  3. Synthetic Data Generation
  4. Cost Sensitive Learning

Class unbalancing: under- and over-sampling

Under-sampling:

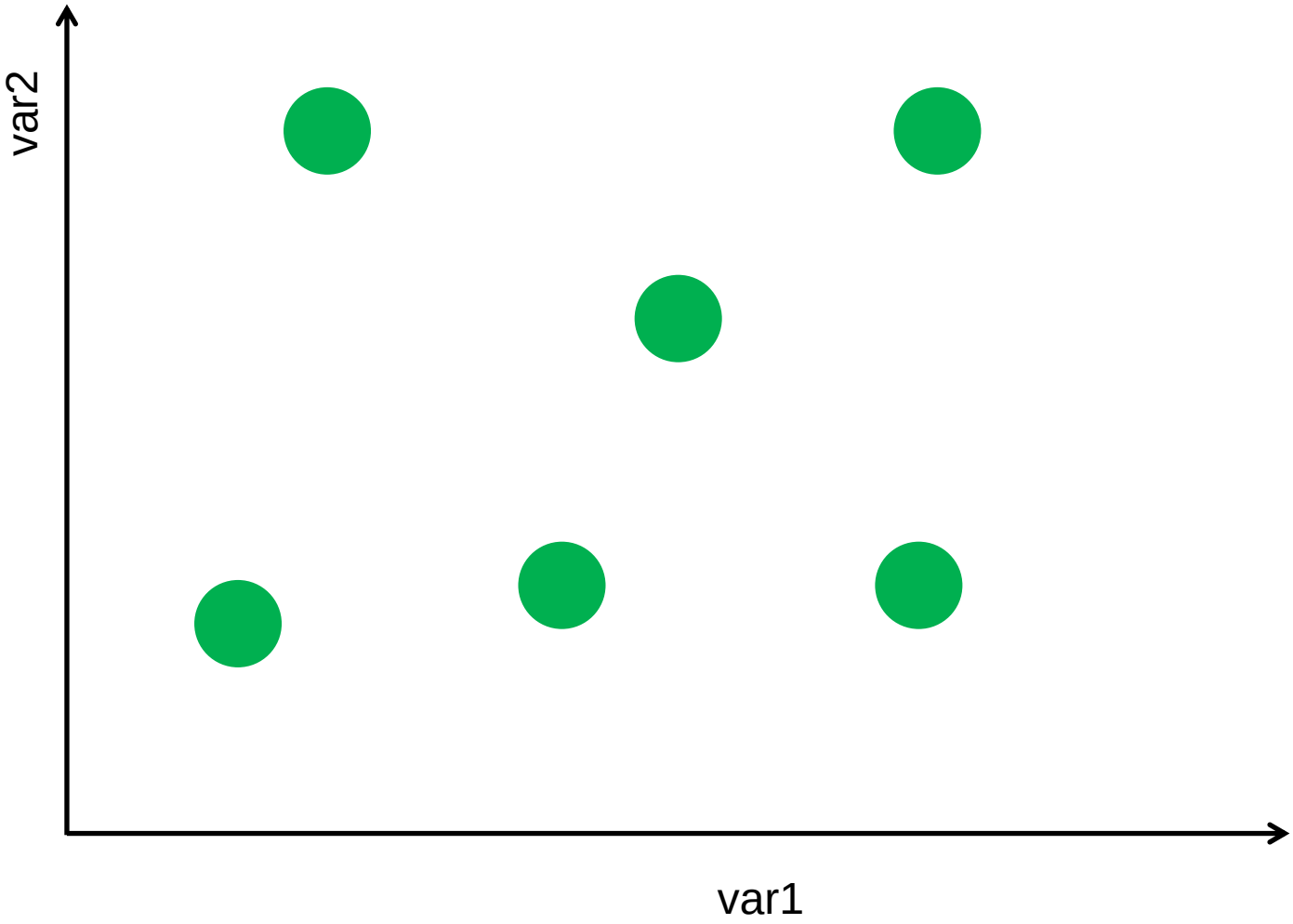
- Reduce the number of observations from majority class to make the data set balanced
- Best method with large datasets
- *Random method*:
  - It randomly remove records in the majority class to re-establish the balancing across all classes
- *Informative method*:
  - It follows a pre-specified selection criterion to remove the observations from majority class
- Possible problems:
  - To loose significant information characterizing the majority class

Over-sampling:

- Replicates the observations from minority class to balance the data
- *Random method*:
  - It randomly duplicate records in the minority
- *Informative method*:
  - It follows a pre-specified selection criterion to duplicate observations from minority class
- Possible problems:
  - By duplicating information in the training set it is possible to introduce bias in the training causing overfitting

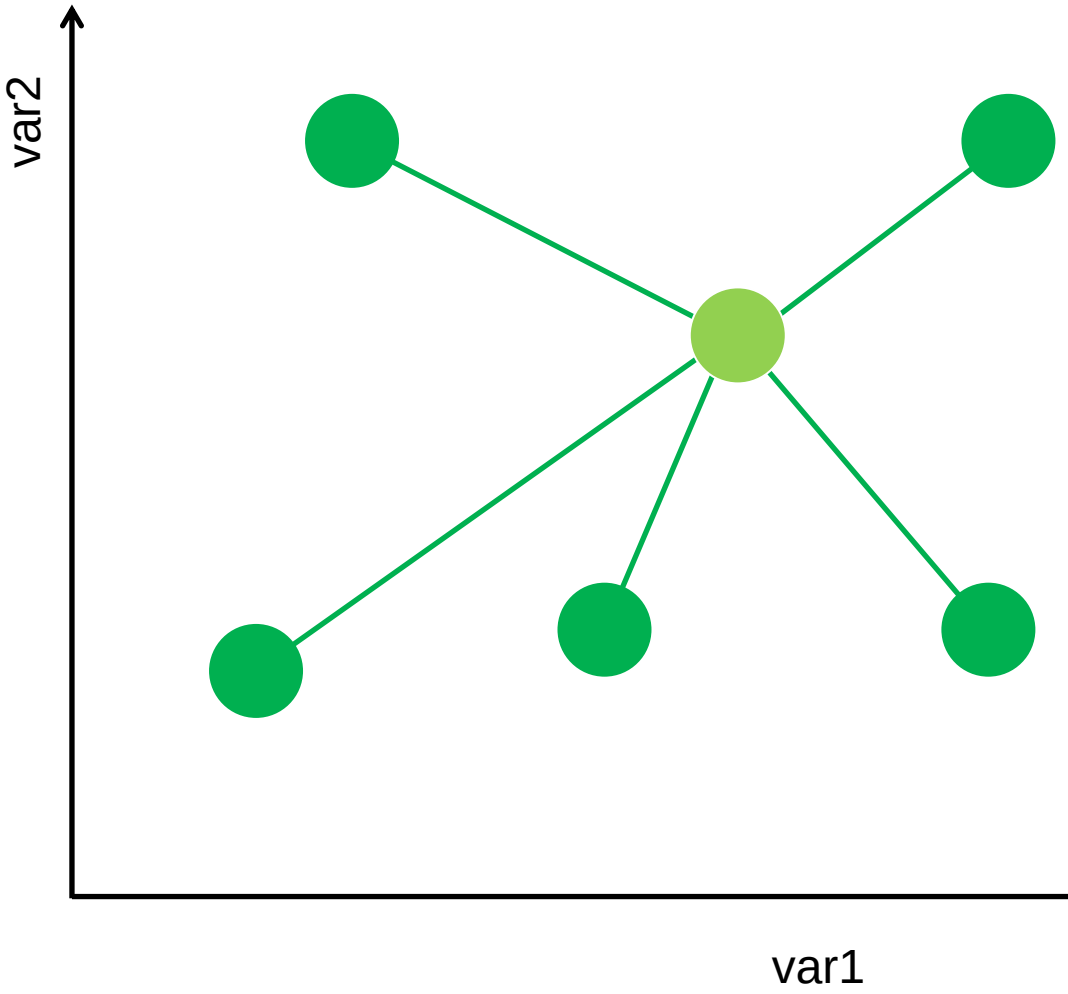
Class unbalancing: synthetic data generation

- It is a special case of over-sampling techniques
- In this case the minority class records are not duplicated
- A set of new synthetic events is created by sampling minority class events in the feature space



Class unbalancing: synthetic data generation

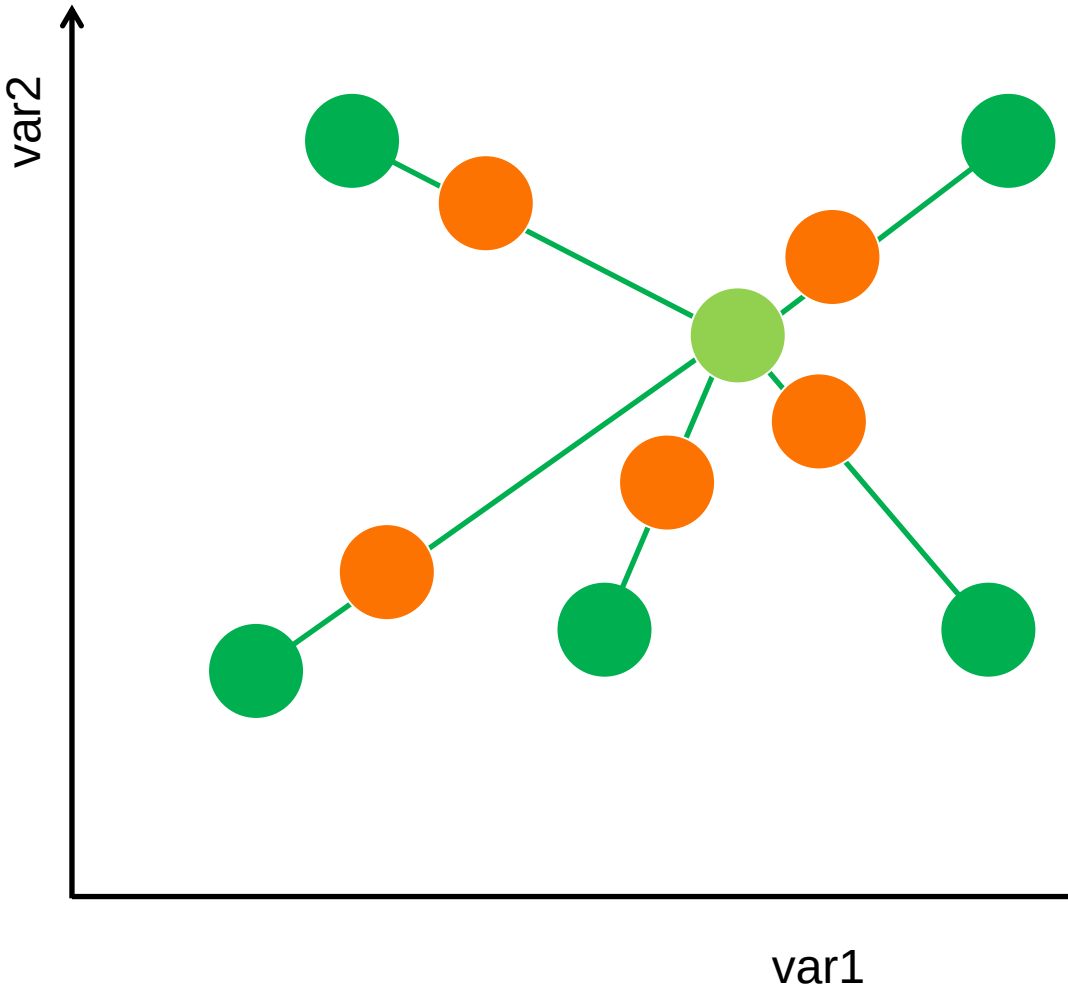
- It is a special case of over-sampling techniques
- In this case the minority class records are not duplicated
- A set of new synthetic events is created by sampling minority class events in the feature space



1. Start from 1 data point belonging to a minority class
2. In the space of the feature consider the *k-nearest neighbours* belonging to the same minority class

Class unbalancing: synthetic data generation

- It is a special case of over-sampling techniques
- In this case the minority class records are not duplicated
- A set of new synthetic events is created by sampling minority class events in the feature space



1. Start from 1 data point belonging to a minority class
2. In the space of the feature consider the *k-nearest neighbours* belonging to the same minority class
3. Generate synthetic data whose coordinates in the feature space are an a random weighted average of the coupled points

Class unbalancing: cost sensitive learning

- This method evaluates the cost associated with misclassifying observations
- It does not create balanced data distribution
- It highlights the imbalanced learning problem by using cost matrices which describes the cost for misclassification in a particular scenario

Example:

- We want write a classifier to identify a terrorist in an airport using a brilliant dataset with many useful features
- The most of the people in airports are standard passengers and not terrorists
- If the model by mistake identify a normal passenger as a terrorist the cost of this error is just some minutes of working time of the wards who have to do al necessary checks on the terrorist candidate
- If the model by mistake doesn't identify a terrorist we are in big trouble: hundreds or thousand of life will be lost and eventually there will be billion of costs in structural damages to the airport
- Therefore we define a cost matrix

		prediction	
		terrorist	normal
true	terrorist	0	100000
	normal	1	0

- The actual loss function that the model will use for training will be not just the *precision* in example but a recomputed precision where each misclassified terrorist counts as 100000 misclassified normal passengers
- Since in this case we are only assigning a weight to the event and we are not duplicating any event the models are less prone to overfitting with respect to models trained with oversampling

GENERALI

MHPC Master in High Performance Computing SISSA ICTP

Day 2: To give structure to the unstructured

Dealing with textual unstructured data

- In many cases the data we have to work with are not in a nice structured form
  - You need to scrap the data from the web
  - You need to classify thousands of news
  - …
- It might happen that the useful information are hidden inside plain text
- How to approach this problem using ML?
- Can we teach to a computer to read and understand?

From text to features: bag of words representation

- The simplest thing to do is to consider that the information in a text is usually expressed by words
  - We can use the words as features of the model
  - We could use the count of the words in a sentence or document that we cant to classify as variables

text	*class*
A dog is eating a shoe	animals
Tom is wearing a blue shoe	human
Lisa is eating a sandwich	human

dog	*is*	*eating*	*a*	*shoe*	*tom*	*wearing*	*blue*	*lisa*	*sandwich*	*class*
1	1	1	2	1	0	0	0	0	0	animals
0	1	0	1	1	1	1	1	0	0	human
0	1	1	1	0	0	0	0	1	1	human

- Is it that simple? Which could be a first problem that needs to be solved?

From text to features: bag of words representation

dog	*is*	*eating*	*a*	*shoe*	*tom*	*wearing*	*blue*	*lisa*	*sandwich*	*class*
1	1	1	2	1	0	0	0	0	0	animals
0	1	0	1	1	1	1	1	0	0	human
0	1	1	1	0	0	0	0	1	1	human

- There are words which are very common and appear anywhere

From text to features: bag of words representation

dog	*is*	*eating*	*a*	*shoe*	*tom*	*wearing*	*blue*	*lisa*	*sandwich*	*class*
1	1	1	2	1	0	0	0	0	0	animals
0	1	0	1	1	1	1	1	0	0	human
0	1	1	1	0	0	0	0	1	1	human

- There are words which are very common and appear anywhere
- There are words which alone can be ambiguous

From text to features: bag of words representation

dog	*is*	*eating*	*a*	*shoe*	*tom*	*wearing*	*blue*	*lisa*	*sandwich*	*class*
1	1	1	2	1	0	0	0	0	0	animals
0	1	0	1	1	1	1	1	0	0	human
0	1	1	1	0	0	0	0	1	1	human

- There are words that are very common and appear anywhere
- There are words that alone can be ambiguous
- There are words that looks significant for the classification but they are only related to the training set (why a dog should not eat a sandwich?)

Many possible features

Very easy to over-fit a model!!!!

Ambiguity

The Term Frequency and Inverse Document Frequency (TF IDF)

- The feature selection step is very important in text analytics
- The simple word counting is prone to over-consider words which are not too significant and too common
- A standard solution is:
  - suppress words which are too common in a language using an hard-coded *stop-words* list (skip all words contained in the list)
  - Use the *term frequency* (TF) and the *inverse document frequency* (IDF) to create a weight for each word in the dictionary which suppress common words and enhance rare words

$$\text{tf}{-}\text{idf}(w,d) = \text{tf}(w,d) \times \text{idf}(w)$$

$$\text{idf}(w) = log\left(\frac{1+n_d}{1+\text{df}(w)}\right)$$

n_d = total number of documents in the dataset

$\text{df}(w)$ = number of documents where the word *w* is present

The Term Frequency and Inverse Document Frequency (TF IDF)

$$\text{tf}{-}\text{idf}(w,d) = \text{tf}(w,d) \times \text{idf}(w)$$

$$\text{idf}(w) = log\left(\frac{1+n_d}{1+\text{df}(w)}\right)$$

- After the tf-idf transformation each feature vector is normalized to the unit to avoid bias due to different document lengths