

Grado en ingeniería informática
Artificial Intelligence 2021/2022

Introduction to Machine Learning

Lara Quijano Sánchez



Universidad Autónoma
de Madrid

SCHEMA

- ☐ Introduction: Inductive learning from data
- ☐ Classification techniques
- ☐ Regression techniques

Motivation

- ❑ Information society generates an explosive amount of data
- ❑ We want to automate the process of analyzing this data
- ❑ We introduce knowledge through examples to:
 - ❑ customize user-centric systems
 - ❑ modify the behavior of a smart agent
- ❑ Software development is a bottleneck

What can be learned?

- ☐ Medical diagnostics
- ☐ Image recognition
- ☐ Finance: credit approval, market prediction . . .
- ☐ Internet: spam filters, trends in social networks (influencers), taste inference
- ☐ Bioinformatics: Protein structure, gene function, ...
- ☐ Texts: fake news, bots, opinion, hate messages
- ☐ Games and sports: prediction of plays, movements, results
- ☐ Learning analytics
- ☐ Criminology: false reports, prediction of recidivism, clustering of crimes

Data science

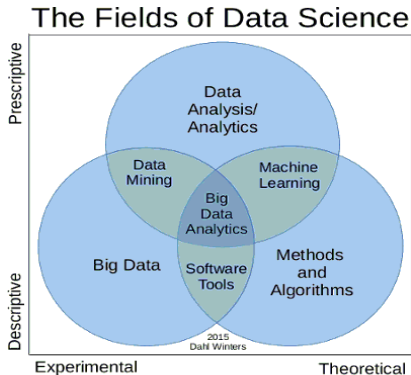
Data science hypothesis

If a model describes the target concept well in a large / significant number of examples, it will also describe the target concept in future examples.

Data science

- ❑ A process that extracts relationships, trends, or patterns that exist in a large group of data
- ❑ Union between many branches of science
- ❑ It is the basis of data analysis in **Big Data**

Data Science Branches

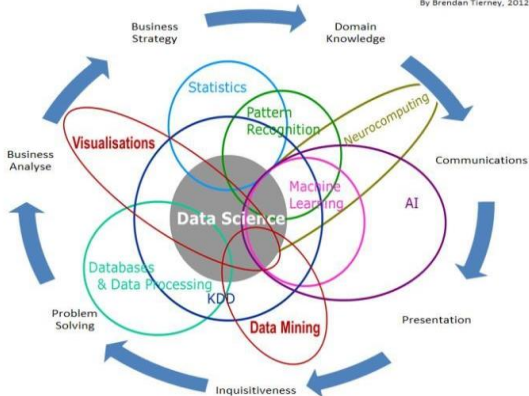


Source: POOJA BISHT. Clarifying Differences between Data Analysis, Data Mining, Data Science, Machine Learning, and Big Data. 2019

Data Science Branches

Data Science Is Multidisciplinary

By Brendan Tierney, 2012



Data science is

- ❑ Statistics, through data **analysis** (data mining), encompasses all the methods used to **extract useful information** from the data and **make decisions**.

Stages of Data Mining

1. **Goal definition**
 - Task to perform: classify, predict, group, ...
2. **Collection and understanding of data**
 - General characteristics, visualization, quality verification
3. **Pre-processing**
 - Integration of sources, generation of attributes, selection of attributes, transformations, ...
4. **Modeling**
 - Machine learning technique to generate / train the model
5. **Evaluation and analysis of results**
 - Performance/error estimation, choice of the best model
6. **Using the model**
 - Development or integration in an information system/
intelligent agent

Machine learning tasks

❑ Classification

- ❑ Determine for a situation / element which **category** it is in

❑ Regression - Predicción

- ❑ Predict for a situation / element what is its approximate **value**

❑ Clustering

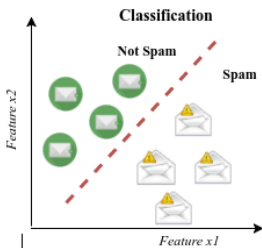
- ❑ Establish in a set of **elements** which ones are **similar to** each other

❑ Frequent Sets

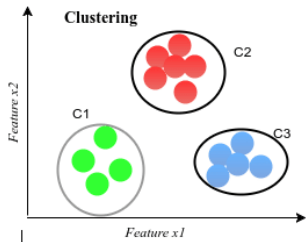
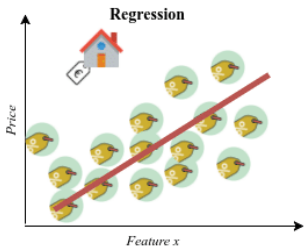
- ❑ Determine in a list of events which **elements are repeated** frequently

Depending on the existing dataset

- ❑ Supervised learning (all data tagged)
- ❑ Unsupervised learning (no tagged data)
- ❑ Semi-supervised learning (some labeled data)



Supervised learning



Unsupervised learning

DATA REPRESENTATION

- ❑ **Dataset:** Table that contains the data that we are going to analyze.

id	age	marital status	savings	Education level	Work	house	amount	class
1	35	single	7,000	highschool	qualified	own	50K	good
2	23	married	2,000	vocational training	qualified	rent	70K	good
3	30	married	1,000	highschool	No-qualified	own	60K	bad
4	26	single	15,000	Bachelor's	autonom.	own	120K	good
5	50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
6	43	single	NA	highschool	autonom.	NA	30K	bad
7	31	divorced	28,000	Master	No-qualified	own	90K	bad
8	33	married	NA	Bachelor's	No-qualified	rent	30K	good
9	40	single	11,000	Master	qualified	own	100K	good

GENERAL CONCEPTS

- **Attribute:** Characteristic that partially describes the elements.

age	marital status	savings	Education level	Work	house	amount	class
35	single	7,000	highschool	qualified	own	50K	good
23	married	2,000	vocational training	qualified	rent	70K	good
30	married	1,000	highschool	No-qualified	own	60K	bad
26	single	15,000	Bachelor's	autonom.	own	120K	good
50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
43	single	NA	highschool	autonom.	NA	30K	bad
31	divorced	28,000	Master	No-qualified	own	90K	bad
33	married	NA	Bachelor's	No-qualified	rent	30K	good
40	single	11,000	Master	qualified	own	100K	good

GENERAL CONCEPTS

- **Instance:** An element defined by the values of its attributes

age	marital status	savings	Education level	Work	house	amount	class
35	single	7,000	highschool	qualified	own	50K	good
23	married	2,000	vocational training	qualified	rent	70K	good
30	married	1,000	highschool	No-qualified	own	60K	bad
26	single	15,000	Bachelor's	autonom.	own	120K	good
50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
43	single	NA	highschool	autonom.	NA	30K	bad
31	divorced	28,000	Master	No-qualified	own	90K	bad
33	married	NA	Bachelor's	No-qualified	rent	30K	good
40	single	11,000	Master	qualified	own	100K	good

GENERAL CONCEPTS

- ▶ **Class:** Disjoint subsets (categories) into which we divide the set of instances.
- ▶ In classification tasks it is the attribute to predict

age	marital status	savings	Education level	Work	house	amount	class
35	single	7,000	highschool	qualified	own	50K	good
23	married	2,000	vocational training	qualified	rent	70K	good
30	married	1,000	highschool	No-qualified	own	60K	bad
26	single	15,000	Bachelor's	autonom.	own	120K	good
50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
43	single	NA	highschool	autonom.	NA	30K	bad
31	divorced	28,000	Master	No-qualified	own	90K	bad
33	married	NA	Bachelor's	No-qualified	rent	30K	good
40	single	11,000	Master	qualified	own	100K	good

GENERAL CONCEPTS

- ▶ **Positive example** of a class: instance that belongs to the subset defined by the class
- ▶ **Negative example** of a class: instance that does not belong to the subset defined by the class
- ▶ **Unknown values (NA):** The value of the attribute in an instance is unknown.
 - ❑ omission in the data collection mechanism: inquiry error
 - ❑ the value does not exist in reality

Stages of Data Mining

1. Goal definition
 - Task to perform: classify, predict, group, ...
2. **Collection and understanding of data**
 - General characteristics, visualization, quality verification
3. Pre-processing
 - Integration of sources, generation of attributes, selection of attributes, transformations, ...
4. Modeling
 - Machine learning technique to generate / train the model
5. Evaluation and analysis of results
 - Performance/error estimation, choice of the best model
6. Using the model
 - Development or integration in an information system/
intelligent agent

Example

❑ Import dataset and print the first rows

❑ Source:

JSON - NYC Parks

❑ https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwicsoC3pr7tAhUKYsAKHY0SB_0QFjAAegQIAxAC&url=https%3A%2F%2Fwww.nycgovparks.org%2Fbigapps%2FDPR_Hiking_001.json&usg=AOvVaw1jujJCwxxV5NVJ7gFCBq-k

```
import pandas as pd
hiking = pd.read_json("datasets/hiking.json")
print(hiking.head())
```

	Accessible Difficulty		Length	Limited_Access
0	Y	None	0.8 miles	N
1	N	Easy	1.0 mile	N
2	N	Easy	0.75 miles	N
3	N	Easy	0.5 miles	N
4	N	Easy	0.5 miles	N

Example

□ Look at all the columns, print the types

```
print(hiking.columns)
```

```
Index(['Accessible', 'Difficulty',  
      'Length', 'Limited_Access',  
      'Location', 'Name',  
      'Other_Details', 'Park_Name',  
      'Prop_ID', 'lat', 'lon'],  
      dtype='object')
```

```
print(hiking.dtypes)
```

```
Accessible      object  
Difficulty      object  
Length         object  
Limited_Access  object  
Location       object  
Name           object  
Other_Details   object  
Park_Name      object  
Prop_ID        object  
lat            float64  
lon            float64  
dtype: object
```

Example

☐ Generate basic stats about the dataframe (each column)

- ☐ Mean

- ☐ Standard Deviation

- ☐ Quartiles

☐ Source

- ☐ https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

```
print(wine.describe())
```

	Type	Alcohol	...	Alcalinity of ash
count	178.000000	178.000000	...	178.000000
mean	1.938202	13.000618	...	19.494944
std	0.775035	0.811827	...	3.339564
min	1.000000	11.030000	...	10.600000
25%	1.000000	12.362500	...	17.200000
50%	2.000000	13.050000	...	19.500000
75%	3.000000	13.677500	...	21.500000
max	3.000000	14.830000	...	30.000000

Why are types important?

```
print(volunteer.dtypes)
```

```
opportunity_id    int64
content_id        int64
vol_requests      int64
...              ...
summary          object
is_priority       object
category_id       float64
```

- object: string/mixed types
- int64: integer
- float64: float
- datetime64 (ortimedelta):
datetime

Source

<https://datahub.io/JohnSnowLabs/nyc-services-volunteer-opportunities>

Always check what types you have

E.g. you might have read and integer as a string and you have to convert it before you start working

```
print(df)
```

```
   A      B      C
1  1  string  1.0
2  2 string2  2.0
3  3 string3  3.0
```

```
print(df.dtypes)
```

```
A      int64
B      object
C      object
dtype: object
```

```
df["C"] = df["C"].astype("float" )
print(df.dtypes)
```

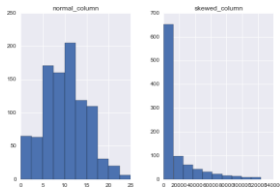
```
A      int64
B      object
C      float64
dtype: object
```

Print the data to understand it

~ check if assumptions are met

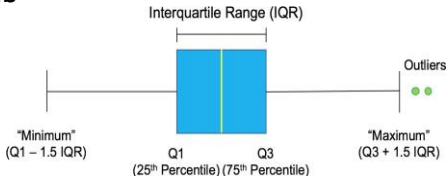
❑ Create histograms

```
import matplotlib as plt
df.hist()
plt.show()
```



❑ Boxplots – help identify outliers

```
df[['column_1']].boxplot()
plt.show()
```

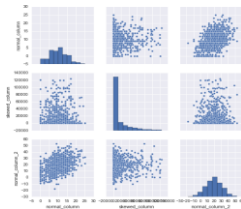


❑ Observe how different features from the dataset interact with each other

❑ Paring distributions

❑ Useful to see if multiple columns are correlated with each other

```
import seaborn as sns
sns.pairplot(df)
```



Stages of Data Mining

1. Goal definition
 - Task to perform: classify, predict, group, ...
2. Collection and understanding of data
 - General characteristics, visualization, quality verification
3. Pre-processing
 - Integration of sources, generation of attributes, selection of attributes, transformations, ...
4. Modeling
 - Machine learning technique to generate / train the model
5. Evaluation and analysis of results
 - Performance/error estimation, choice of the best model
6. Using the model
 - Development or integration in an information system/
intelligent agent

When we preprocess?

- ❑ After exploratory data analysis = idea of what you want to do with your dataset
- ❑ Then -> Prepping data for modelling. Eg:
 - ❑ Cleaning: If you have missing values (NaN) you need to preprocess too
 - ❑ Outliers: data points that differ significantly from other observations
 - ❑ Modelling in ML requires numerical inputs, i.e. If you have categorical variables you need to preprocess
 - ❑ Transforming existing variables
 - ❑ Creating new variables from existing ones
- ❑ Iterative process
- ❑ Requires knowledge of domain/data

Missing data

- ❑ If dataset is big enough and representative
 - ❑ We can afford losing a few rows
 - ❑ drop all NaN's

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.dropna())
```

	A	B	C
1	4.0	7.0	3.0
4	5.0	9.0	7.0

Missing data: drop rows

- ❑ If dataset is big enough and representative
 - ❑ We can afford losing a few rows
 - ❑ drop specific rows

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.drop([1, 2, 3]))
```

	A	B	C
0	1.0	NaN	2.0
4	5.0	9.0	7.0

Missing data: drop columns

- ❑ If dataset is big enough and representative
 - ❑ We can afford losing a few rows
 - ❑ drop a particular column specially if all or most of the values are missing

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df.drop("A", axis=1))
```

	B	C
0	NaN	2.0
1	7.0	3.0
2	NaN	NaN
3	7.0	NaN
4	9.0	7.0

Column
name

Choose
Columns

Missing data: drop only some rows

❑ If dataset is big enough and representative

❑ We can afford losing a few rows

❑ drop rows with NaN on a particular column

```
print(df)
```

	A	B	C
0	1.0	NaN	2.0
1	4.0	7.0	3.0
2	7.0	NaN	NaN
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df[df["B"].notnull()])
```

	A	B	C
1	4.0	7.0	3.0
3	NaN	7.0	NaN
4	5.0	9.0	7.0

```
print(df["B"].isnull().sum())
```

```
2
```

Missing data: count first

❑ How many missing values we have:

❑ Which ones are valid values, zero is ok? Negative? NaN?. Eg. Age

```
# count the number of zero values for each column
num_missing = (df[[1,2,3,4,5]] == 0).sum()
# report the results
print(num_missing)
```

```
1      5
2     35
3    227
4    374
5     11
```

```
# count the number of nan values in each column
print(df.isnull().sum())
```

```
1     10
2     20
3    127
4    424
5     41
```

Missing data: replace

❑ If dataset is NOT big enough

❑ We **CAN NOT** afford losing rows

❑ Approximate the value of the missing data

❑ With the median, mean, mode for the column, etc

❑ A constant value that has meaning within the domain, such as 0, distinct from all other values.

❑ A value from another randomly selected record.

❑ A value estimated by another predictive model.

```
from numpy import nan
# fill missing values with mean column values
df.fillna(df.mean(), inplace=True)
```

```
from numpy import nan
from numpy import isnan
from pandas import read_csv
from sklearn.impute import SimpleImputer
# load the dataset
dataset = read_csv('pima-indians-diabetes.csv', header=None)
# mark zero values as missing or NaN
dataset[[1,2,3,4,5]] = dataset[[1,2,3,4,5]].replace(0, nan)
# retrieve the numpy array
values = dataset.values
# define the imputer
imputer = SimpleImputer(missing_values=nan, strategy='mean')
# transform the dataset
transformed_values = imputer.fit_transform(values)
# count the number of NaN values in each column
print('Missing: %d' % isnan(transformed_values).sum())
```



Different library.
Not pandas

What is standardization?

- ❑ Transform continuous data to make it look normally distributed
- ❑ Some models assume normally distributed data. If it is not you risk biasing your model
- ❑ What can we do, applied to continuous numerical data
 - ❑ Log normalization and feature scaling
- ❑ When we standardize:
 - ❑ Linearity assumptions: Models that operate in a linear space: eg. k-nn, linear regression, k-means clustering... Should have data in a linear space
 - ❑ Possible problems/bias when dataset features have high variance
 - ❑ Eg. A feature that has an order of magnitude or more greater than other features
 - ❑ Dataset features are continuous and on different scales.
 - ❑ Eg. Dataset related to height and weight. To compare them, they must be in the same linear space -> standardized

Log normalization

- ❑ Use with a column with high variance
 - ❑ In a knn prediction, a variable with high variance can make your model's accuracy decrease
- ❑ Applies log transformation
- ❑ Natural log using the constant e (2.718). Eg $e^{3.4} = 30$
- ❑ Captures relative changes, the magnitude of change, and keeps everything in the positive space

Number	Log
30	3.4
300	5.7
3000	8

```
print(df)
```

```
   col1  col2
0  1.00   3.0
1  1.20  45.5
2  0.75  28.0
3  1.60 100.0
```

```
print(df.var())
```

```
col1      0.128958
col2    1691.729167
dtype: float64
```

```
import numpy as np
df["log_2"] = np.log(df["col2"])
print(df)
```

```
   col1  col2  log_2
0  1.00   3.0  1.098612
1  1.20  45.5  3.817712
2  0.75  28.0  3.332205
3  1.60 100.0  4.605170
```

```
print(np.var(df[["col1", "log_2"]]))
```

```
col1      0.096719
log_2      1.697165
dtype: float64
```

Feature scaling: standardization

- ❑ If we want to compare features on different scales
- ❑ Assuming models with linear characteristics
- ❑ Finds the mean of the data and centers/transform the distribution to it
Eg. features around 0 and 1 variance (mean of 0, variance of 1)
- ❑ Transforms to approximately normal distribution

```
print(df)
```

	col1	col2	col3
0	1.00	48.0	100.0
1	1.20	45.5	101.3
2	0.75	46.2	103.5
3	1.60	50.0	104.0

```
print(df.var())
```

col1	0.128958
col2	4.055833
col3	3.526667
dtype:	float64

Low variance in columns
High variance **across** columns

Using a standard method
allows you to repeat the
process easily. Eg test/train set

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
df_scaled = pd.DataFrame(scaler.fit_transform(df),  
                           columns=df.columns)
```

```
print(df_scaled)
```

	col1	col2	col3
0	-0.442127	0.329683	-1.352726
1	0.200967	-1.103723	-0.553388
2	-1.245995	-0.702369	0.799338
3	1.487156	1.476409	1.106776

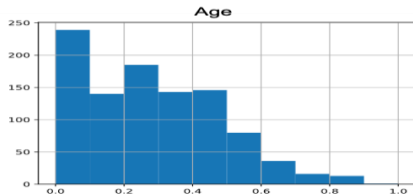
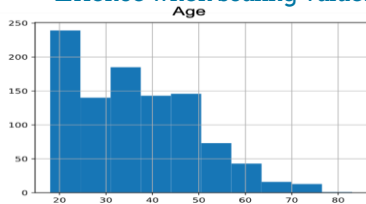
```
print(df.var())
```

col1	1.333333
col2	1.333333
col3	1.333333
dtype:	float64

Normalization as min-max scaling

- Data scaled between a minimum and a maximum

- Hence when scaling values will change but distribution no



```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()  
scaler.fit(df[['Age']])  
df['normalized_age'] =  
scaler.transform(df[['Age']])
```

Outliers

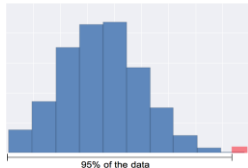
- ❑ Data points that exist far away from the majority of your data
 - ❑ Due to errors or genuine rare occurrences
 - ❑ Remove these values / treat them separately => high impact on models
- ❑ Even after transformation data can be very skewed

❑ Solutions:

❑ Quantile based detection

- ❑ Remove top/bottom 5% of the data
- ❑ Always removes data even if not outliers

```
q_cutoff = df['col_name'].quantile(0.95)
mask = df['col_name'] < q_cutoff
trimmed_df = df[mask]
```



❑ Standard deviation based detection

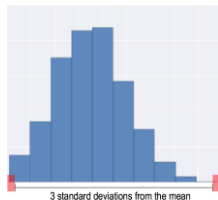
- ❑ Only remove data that is 3 stds away from the mean
 - ❑ Only removes true outliers = data genuinely different from the rest

```
mean = df['col_name'].mean()
std = df['col_name'].std()

cut_off = std * 3

lower, upper = mean - cut_off, mean + cut_off

new_df = df[(df['col_name'] < upper) &
            (df['col_name'] > lower)]
```



Feature engineering

- ❑ Creation of new features based on existing features
- ❑ Insight into relationships between features
- ❑ Extract and expand data
- ❑ Dataset-dependent

user	fav_color
1	blue
2	green
3	orange

Id	Text
1	"Featureengineeringisfun!"
2	"Featureengineeringisalotof work."
3	"Idon'tmindfeatureengineering."

user	test1	test2	test3
1	90.5	89.6	91.4
2	65.5	70.6	67.3
3	781	807	818

Id	Date
4	July 302011
5	January 292011
6	February 052011

Encoding Categorical variables

```
user subscribed fav_color
```

```
0          y         blue
```

```
1          n         green
```

```
2          n         orange
```

```
3          y         green
```

Encoding binary variables - Pandas

```
print(users["subscribed"])
```

```
print(users[["subscribed", "sub_enc"]])
```

```
0    y
1    n
2    n
3    y
Name: subscribed, dtype: object
```

	subscribed	sub_enc
0	y	1
1	n	0
2	n	0
3	y	1

```
users["sub_enc"] = users["subscribed"].apply(lambda val:
                                              1 if val == "y" else 0)
```

Encoding binary variables - scikit-learn

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

users["sub_enc_le"] = le.fit_transform(users["subscribed"])
print(users[["subscribed", "sub_enc_le"]])
```

	subscribed	sub_enc_le
0	y	1
1	n	0
2	n	0
3	y	1

One-hot encoding

fav_color	fav_color_enc
blue	[1, 0, 0]
green	[0, 1, 0]
orange	[0, 0, 1]
green	[0, 1, 0]

Values: [blue, green, orange]

- blue: [1, 0, 0]
- green: [0, 1, 0]
- orange: [0, 0, 1]

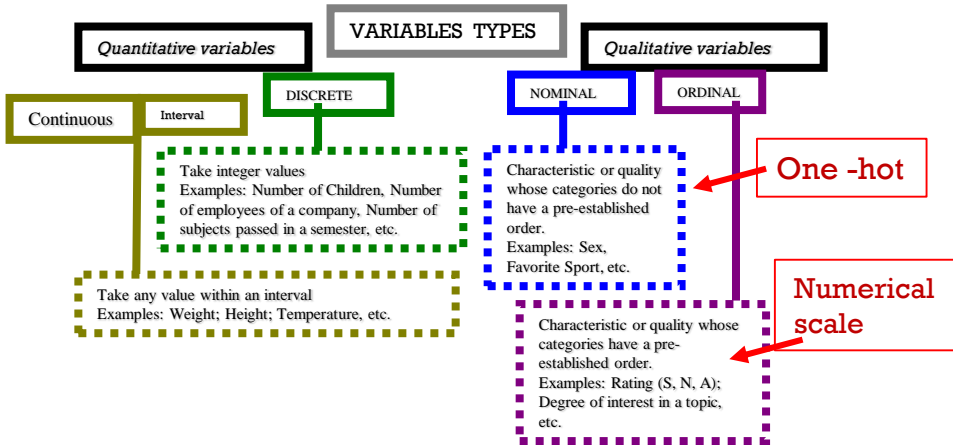
```
print(users["fav_color"])
```

```
0    blue
1    green
2    orange
3    green
Name: fav_color, dtype: object
```

```
print(pd.get_dummies(users["fav_color"]))
```

```
   blue  green  orange
0     1     0      0
1     0     1      0
2     0     0      1
3     0     1      0
```


Types of variables for encoding



Engineering numerical features

- ❑ Aggregating a set of numbers to use in place of features
 - ❑ Reduce dimensionality of feature space
 - ❑ Do not need multiple features close in distance to each other

```
print(df)
```

```
   city  day1  day2  day3
0   NYC  68.3  67.9  67.8
1    SF  75.1  75.5  74.9
2    LA  80.3  84.0  81.3
3 Boston  63.0  61.0  61.2
```

```
columns = ["day1", "day2", "day3"]
df["mean"] = df.apply(lambda row: row[columns].mean(), axis=1)
print(df)
```

```
   city  day1  day2  day3  mean
0   NYC  68.3  67.9  67.8  68.00
1    SF  75.1  75.5  74.9  75.17
2    LA  80.3  84.0  81.3  81.87
3 Boston  63.0  61.0  61.2  61.73
```

Engineering Dates

```
print(df)
```

```
      date purchase
0   July 30 2011  $45.08
1 February 01 2011  $19.48
2  January 29 2011  $76.09
3   March 31 2012  $32.61
4 February 05 2011  $75.98
```

```
df["date_converted"] = pd.to_datetime(df["date"])
df["month"] = df["date_converted"].apply(lambda row: row.month)
print(df)
```

Convert to date type

Get day, month, year, compute age, etc

```
      date purchase date_converted month
0   July 30 2011  $45.08  2011-07-30     7
1 February 01 2011  $19.48  2011-02-01     2
2  January 29 2011  $76.09  2011-01-29     1
3   March 31 2012  $32.61  2012-03-31     3
4 February 05 2011  $75.98  2011-02-05     2
```

Creating new variables

- ❑ Ratio

- ❑ N° of times a word/symptom appears with respect to document length/times in the doctor

- ❑ Frequency

- ❑ N° of times a word/symptom appears

- ❑ Binary

- ❑ Does a word/symptom appear

- ❑ Difference between events

- ❑ Exogenous data

- ❑ City, weather, economic data

- ❑

Feature selection

- ❑ Selecting features to be used for modelling
- ❑ Doesn't create new features
- ❑ Improve model's performance
- ❑ Redundancies:
 - ❑ Remove noisy features
 - ❑ Eg having city name + latitude&longitude
 - ❑ Remove correlated features
 - ❑ Eg dog and dog race
 - ❑ Remove duplicated features
- ❑ Correlated features:
 - ❑ Statistically correlated: features move together directionally
 - ❑ Linear models assume feature independence
 - ❑ Pearson correlation coefficient (1/-1 cor; 0 not cor)

Feature selection

```
# Print out the column correlations of the wine dataset
print(wine.corr())

# Take a minute to find the column where the correlation
value is greater than 0.75 at least twice
to_drop = "Flavanoids"

# Drop those columns from the dataset
volunteer_subset = volunteer.drop(to_drop, axis=1)
```

```
print(df)
```

	A	B	C
0	3.06	3.92	1.04
1	2.76	3.40	1.05
2	3.24	3.17	1.03
...			

```
print(df.corr())
```

	A	B	C
A	1.000000	0.787194	0.543479
B	0.787194	1.000000	0.565468
C	0.543479	0.565468	1.000000

Dimensionality reduction

- ❑ Unsupervised learning method
- ❑ Combines/decomposes a feature space
 - ❑ In linear or non linear fashion
- ❑ Feature extraction - here we'll use to reduce feature space
- ❑ Eg:
 - ❑ Principal component analysis (PCA)
 - ❑ Linear transformation to reduce features in a space where they are uncorrelated
 - ❑ The feature space is reduced and the variance is captured in a meaningful way by **combining features into components**
 - ❑ Captures as much variance as possible in each component
 - ❑ Use when you have a lot of features and no strong candidates for elimination
 - ❑ LASSO

Stages of Data Mining

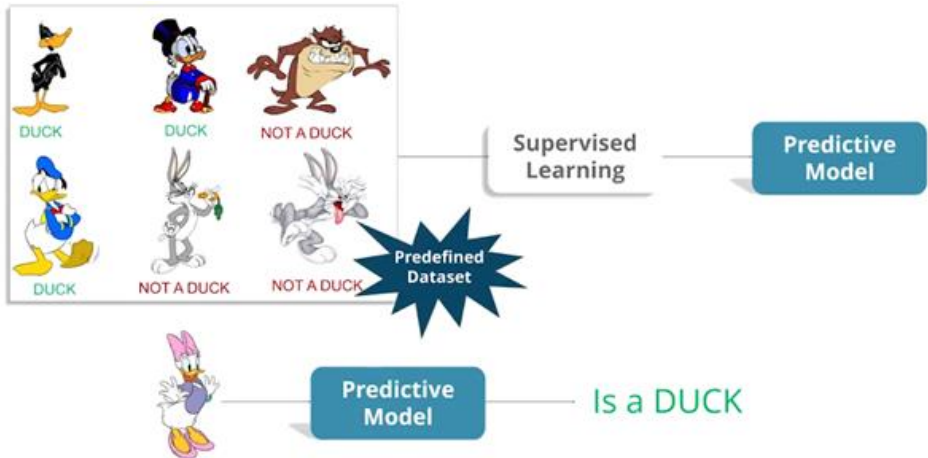
1. Goal definition
 - Task to perform: classify, predict, group, ...
2. Collection and understanding of data
 - General characteristics, visualization, quality verification
3. Pre-processing
 - Integration of sources, generation of attributes, selection of attributes, transformations, ...
4. **Modeling**
 - Machine learning technique to generate / train the model
5. Evaluation and analysis of results
 - Performance/error estimation, choice of the best model
6. Using the model
 - Development or integration in an information system/
intelligent agent

SCHEMA

- ❑ Introduction: Inductive learning from data
- ❑ **Classification techniques**
- ❑ Regression techniques

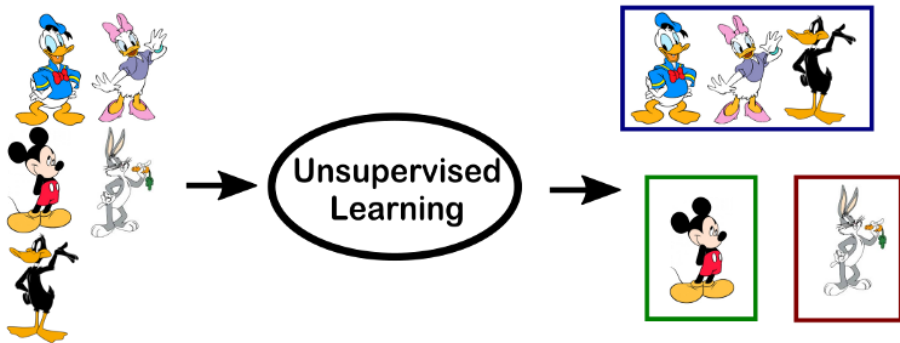
Classification problems

- ▶ When there are tags, it is supervised = identify/ classify me this based on examples



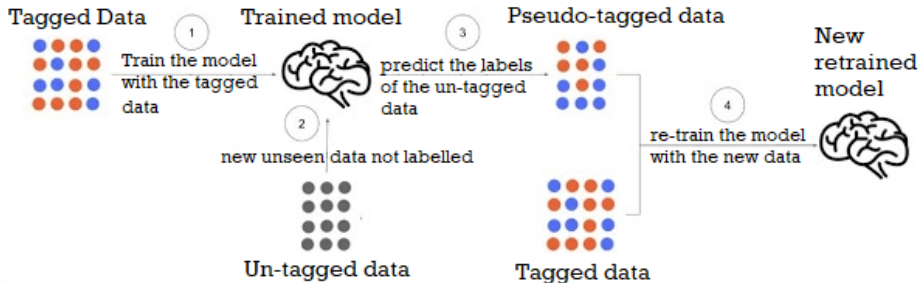
Classification problems

- ▶ When there are no tags, for example clustering, it is unsupervised= find me the best grouping (optimize a function)



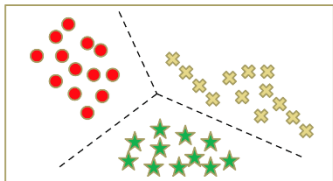
Classification problems

- When there are some tags, but few, we can do semi supervised

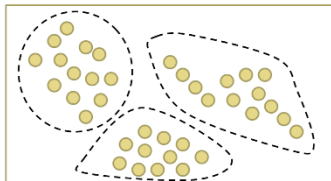


In summary

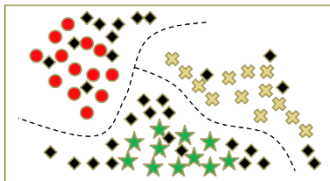
□ See how the frontiers change



Aprendizaje Supervisado

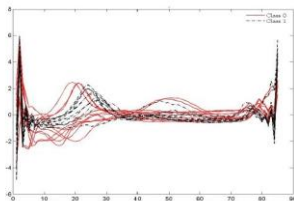
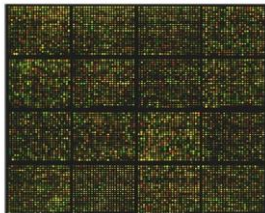
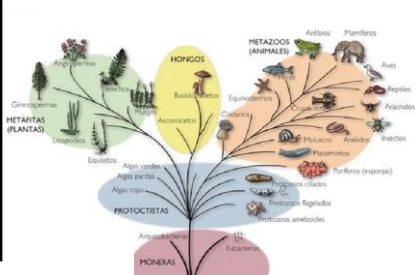
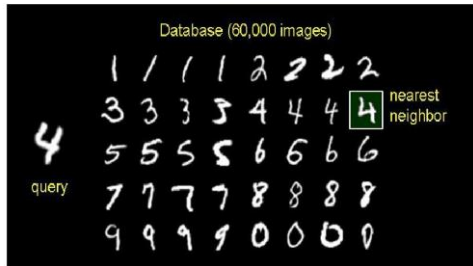


Aprendizaje No Supervisado



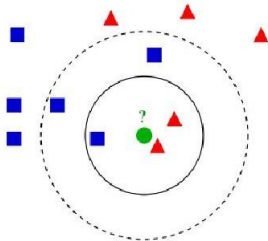
Aprendizaje Semi-Supervisado

Supervised Classification: Examples



Problem definition

Given a new observation X whose class we do not know, the discrimination problem (Fisher 1936) consists in deciding (from the sample) to which population, P_0 or P_1 belongs.



Methodology

- ❑ Data collection: sampling, representativeness ...
- ❑ Exploratory Data Analysis: representation of data, choice of variables, filtering ...
- ❑ Choice of classifier: (non-) parametric, (non-) linear, global or local, mixtures ...
- ❑ Training: algorithms, convergence, rates ...
- ❑ Validation: generalization, cross validation, bootstrap ...
- ❑ Evaluation: obtaining and representing results, margins, confidence ...
- ❑ Decision.

Exploratory Data Analysis

Tasks

- ❑ Fix retrieval problems: imbalance, outliers ...
- ❑ Complete the data: missing values, interpolation ...
- ❑ Filtering: recover X from $Z = X + s$.
- ❑ Representation/coding: basic, categorical variables, functions, derivatives ...
- ❑ Information extraction: redundancy, latent structures ...
- ❑ Select the framework and define the tools.

Keys in a prediction model

1. How is the model represented?
 - ☐ A tree, a set of rules, a probability distribution, a formula ,. . .
 - ☐ Can it be interpreted?
2. How is the model generated / trained?
 - ☐ The learning algorithm: how much data do you need?
 - ☐ How long it takes?
3. How do you predict a new instance?
 - ☐ Applying the formula, answering the questions until finding the class ,. . .

Methods

Classification
Models

Decision Trees

Decision Rules

Bayesian Modeling

Lazy approaches

Neural Networks

Support Vector Machines

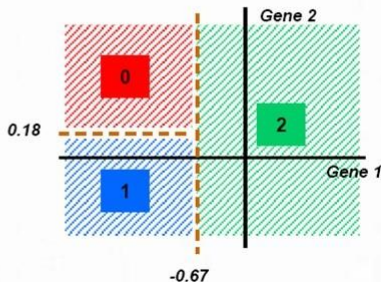
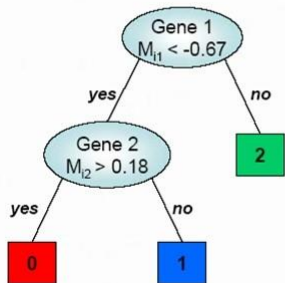
Discriminant Analysis

Methods: Decision trees

Classification and Regression Tree (CART)

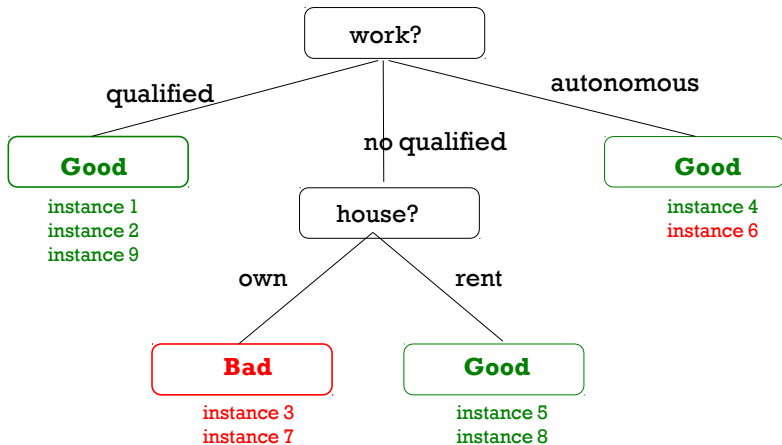
- ▶ Partition the sample space into rectangles and then predict a simple model in each of them
- ▶ binary trees discriminate the space in two subsamples (nodes) from a previous one

Classification tree



Source: George C. Tseng. Classification and clustering problems in microarray analysis and some recent advances. 2004

Methods: Decision tree

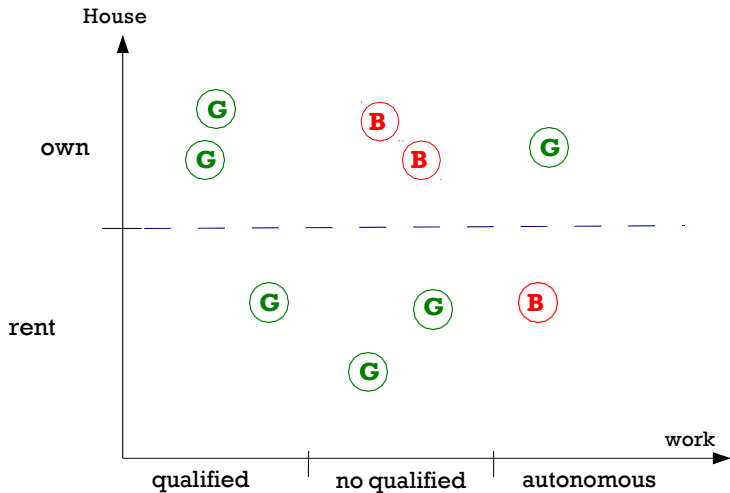


id	age	marital status	savings	Education level	Work	house	amount	class
1	35	single	7,000	highschool	qualified	own	50K	good
2	23	married	2,000	vocational training	qualified	rent	70K	good
3	30	married	1,000	highschool	No-qualified	own	60K	bad
4	26	single	15,000	Bachelor's	autonom.	own	120K	good
5	50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
6	43	single	NA	highschool	autonom.	NA	30K	bad
7	31	divorced	28,000	Master	No-qualified	own	90K	bad
8	33	married	NA	Bachelor's	No-qualified	rent	30K	good
9	40	single	11,000	Master	qualified	own	100K	good

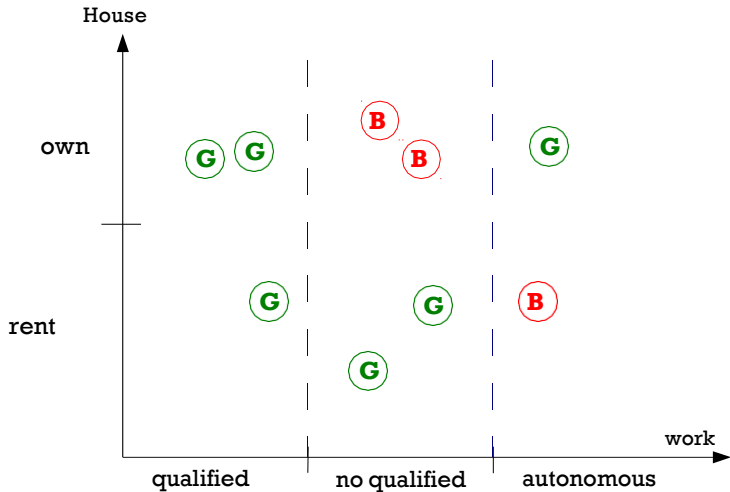
Decision trees (concepts)

- ❑ Diagram representing successive conditions on attributes to classify an instance
- ❑ Node types
 - ❑ Internal nodes:
 - ❑ Conditional questions about attributes
 - ❑ Each answer follows a bow or arrow
 - ❑ Complete separation of the examples between the possible answers
 - ❑ Leaf nodes
 - ❑ Class → prediction
 - ❑ Prediction confidence
 - ❑ Training examples that fulfilled the conditions up to the leaf node
- ❑ Modelling objectives
 - ❑ Build the simplest tree that best separates the instances by class
 - ❑ The final model must generalize to classify future instances well

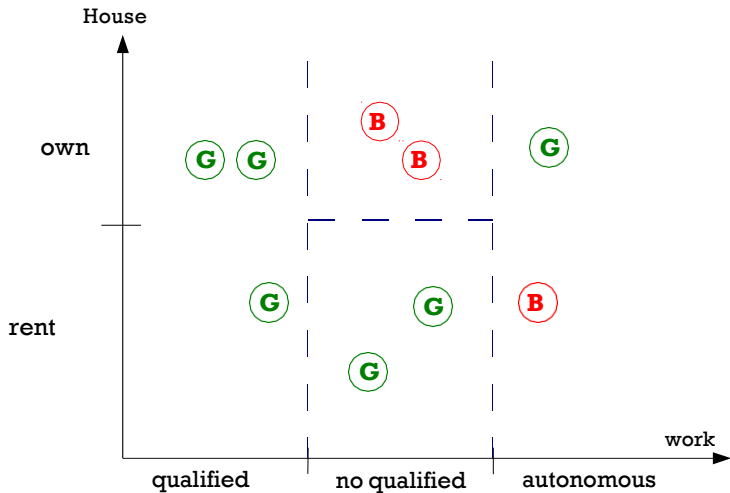
Decision tree strategy (inefficient)



Decision tree strategy



Decision tree strategy



Decision tree generation

- ❑ Inductive construction: An attribute with all its possible answers is added in each step
- ❑ The attribute that **best separates** (orders) the examples according to the classes is selected
- ❑ Separation criteria
 - ❑ Information gain
 - ❑ Gini impurity
 - ❑ Entropy
- ❑ The process stops when adding a new attribute does not improve the separation criteria

Attribute relevance

- ❑ Decision tree construction performs implicit attribute selection

Other (hiper)parameters

- ☐ Minimum gain
- ☐ Maximum tree depth
- ☐ Minimum examples per leaf
- ☐ Pruning tree

Decision Rules

Example

IF house= own AND age<= 28

THEN class = **good**

IF work= no qualified AND amount =>
50K

THEN class = **bad**

id	age	marital status	savings	Education level	Work	house	amount	class
1	35	single	7,000	highschool	qualified	own	50K	good
2	23	married	2,000	vocational training	qualified	rent	70K	good
3	30	married	1,000	highschool	No-qualified	own	60K	bad
4	26	single	15,000	Bachelor's	autonom.	own	120K	good
5	50	divorced	3,500	Bachelor's	No-qualified	rent	40K	good
6	43	single	NA	highschool	autonom.	NA	30K	bad
7	31	divorced	28,000	Master	No-qualified	own	90K	bad
8	33	married	NA	Bachelor's	No-qualified	rent	30K	good
9	40	single	11,000	Master	qualified	own	100K	good

Decision Rule

Example

IF house= own AND age<= 28
THEN class = **good**

IF work= no qualified AND amount =>
50K
THEN class = **bad**

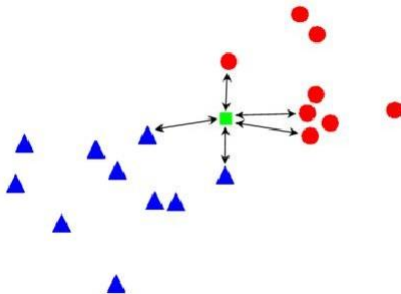
- ❑ Inductive construction based on a separation or coverage criterion
- ❑ Complete separation of instances is not required (overlapping rules)
- ❑ All decision trees can be translated into decision rules

Instance-Based Learning

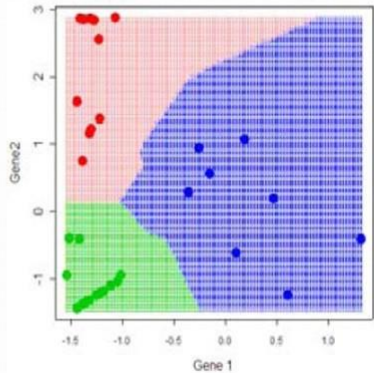
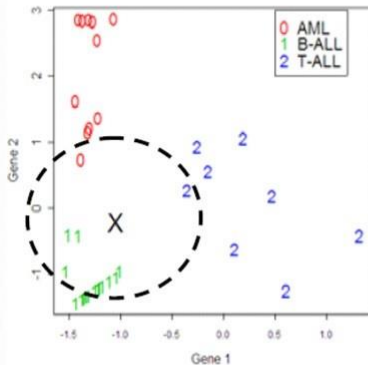
- ❑ Another technique for generating classifiers
- ❑ Training
 - ❑ There is no generation of a model!
 - ❑ Only all instances are stored
- ❑ Evaluation or Use (generalization step):
 - ❑ A set of instances that are **similar** to the ones you want to classify is extracted
 - ❑ A decision is made based on the selected instances
- ❑ Type algorithm: k-nearest neighbors (k-NN)

k-NN

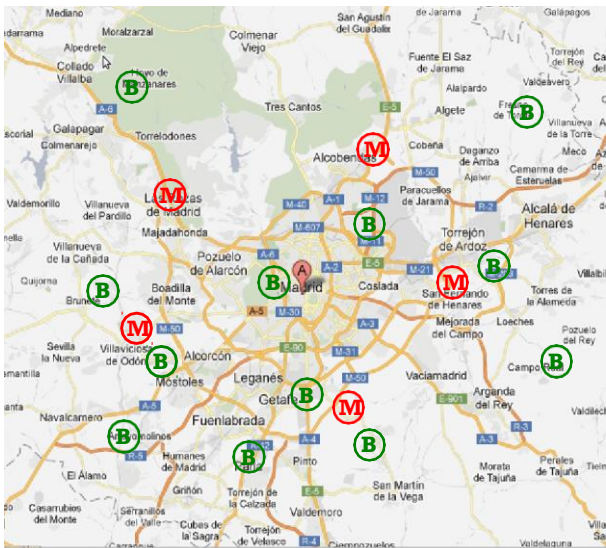
The idea behind k-NN is very simple. Given an observation x , the k observations closest to x (k-neighbors) among the X_i of the training set are searched. The majority class among k-neighbors is assigned x .



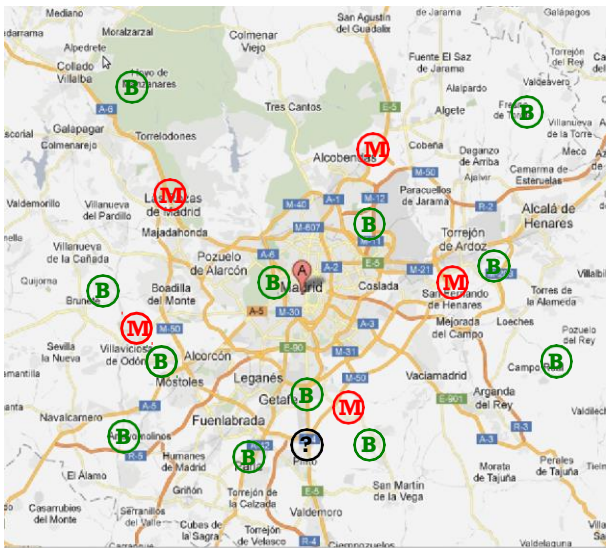
Nearest neighbor rule



k-NN



k-NN



k-NN

- ❑ The (hyper)parameter k defines the number of neighbors that are chosen to make the decision
- ❑ Similarity measure
 - ❑ Each instance is in a space of dimension n
 - ❑ The nearest neighbors are determined by a distance measure, like the Euclidean:

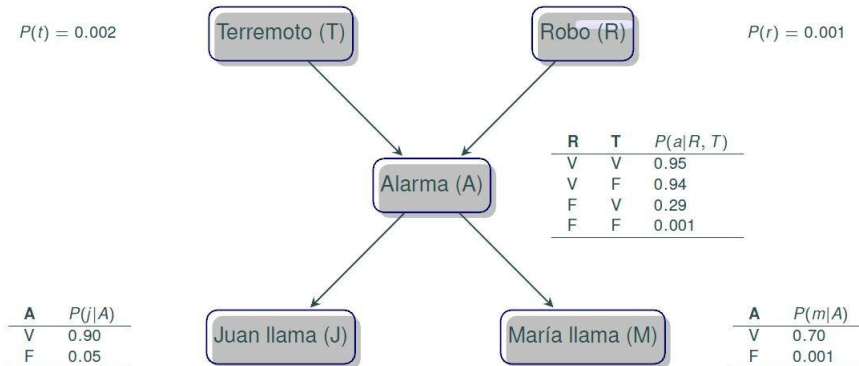
$$d(x_i, x_j) = \sqrt{\sum_{a=1}^n (x_i[a] - x_j[a])^2}$$

Bayesian learning

- ❑ Bayesian decision theory is based on two assumptions
 - ❑ The decision problem can be described in probabilistic terms
 - ❑ All probabilities of the problem are known or at least can be estimated
- ❑ Decisions are made based on observed data

Bayesian Network: example alarm

- Each root node (without parents) has a Priority Probability Distribution
- Each internal node has a Conditional Probability Table (CPT) that quantifies the effect its parents have on it



How to generate a Bayesian network?

☐ We need

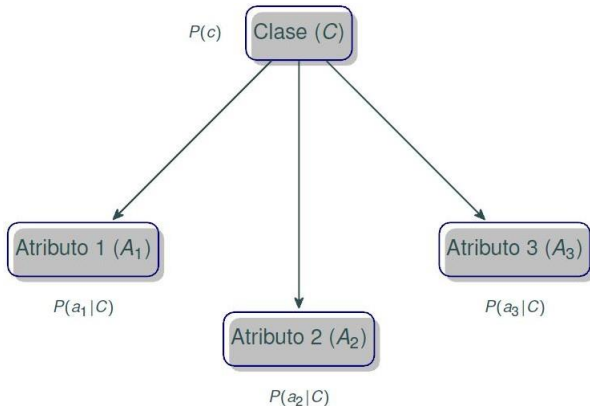
- ☐ Random variables and their domains
- ☐ The structure of the network: conditional independence
- ☐ The conditional probability tables (CPTs)

☐ Where do we get it from?

- ☐ We have to design the network
- ☐ We can ask an expert
- ☐ CPTs can be learned from data
- ☐ The structure can also be learned from the data, but it is more complicated

Naïve Bayes Classifier

is a special case of a Bayesian network with the following structure:



Notation

- Set of classes

$$C = \{c_1, c_2, \dots, c_n\}$$

- Attribute set

$$A = \{a_1, a_2, \dots, a_n\}$$

- Instance: attribute values

$$X = \{x_1, x_2, \dots, x_n\}$$

- Conditional probabilities:

- $P(c_j | X)$ Probability of observing class c_j given instance X
- $P(X | c_j)$ Probability of observing instance X given class c_j

Bayesian classifiers

Observations

id	age	marital status	savings	Education level	Work	house	amount	class
1	35	single	7,000	<u>highschool</u>	qualified	own	50K	good
2	23	married	2,000	vocational training	qualified	rent	70K	good
3	30	married	1,000	<u>highschool</u>	<u>No-qualified</u>	own	60K	bad
4	26	single	15,000	Bachelor's	<u>autonom.</u>	own	120K	good
5	50	divorced	3,500	Bachelor's	<u>No-qualified</u>	rent	40K	good
6	43	single	NA	<u>highschool</u>	<u>autonom.</u>	NA	30K	bad
7	31	divorced	28,000	Master	<u>No-qualified</u>	own	90K	bad
8	33	married	NA	Bachelor's	<u>No-qualified</u>	rent	30K	good
9	40	single	11,000	Master	<u>qualified</u>	own	100K	good

Decision problem

☐ $P(\text{class} = \text{good} \mid \text{work} = \text{qualified}, \text{savings} = 50 - 100\text{K}, \text{home} = \text{own}, \text{age} = 35-40)$

☐ $P(\text{class} = \text{bad} \mid \text{work} = \text{qualified}, \text{savings} = 50 - 100\text{K}, \text{home} = \text{own}, \text{age} = 35-40)$

☐ ¿good o bad?

Bayesian classifiers

Bayes theorem

$$P(c_j|x_j) = \frac{P(x_j|c_j)P(c_j)}{P(x_j)}$$

- The a priori probability of an instance x_i is independent of the value of the class, so $P(x_i)$ is generally not calculated
- The idea of the classifier is to choose the most probable class according to the posterior probability $P(c_j | x_i)$

$$\text{BayesianClassifier}(x_i) = \underset{c_j \in \mathcal{C}}{\operatorname{argmax}} P(x_i|c_j)P(c_j)$$

Naïve Bayes Classifier

- $P(X_i \mid c_j)P(c_j) = P(x_1, x_2, \dots, x_n \mid c_j)P(c_j)$
- attribute values are assumed to be independent

$$NaiveBayes(X) = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in A}^n P(x_i | c_j)$$

Naïve Bayes - Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Naïve Bayes - Example

$x = \langle \text{Outlook}=\text{Sunny}, \text{Temp}=\text{Cool}, \text{Hum}=\text{High}, \text{Wind}=\text{Strong} \rangle$

$$h_{NB} = \operatorname{argmax} P(h) P(x|h)$$

$$= \operatorname{argmax} P(h) \prod P(a_i | h)$$

$$= \operatorname{argmax} P(h) P(\text{Outlook}=\text{Sunny} | h) P(\text{Temp}=\text{Cool} | h) \\ P(\text{Hum}=\text{High} | h) P(\text{Wind}=\text{Strong} | h)$$

Aproximando las probabilidades por la frecuencia:

$$P(\text{PlayTennis} = \text{yes}) = 9/14 = 0.64$$

$$P(\text{PlayTennis} = \text{no}) = 5/14 = 0.36$$

$$P(\text{Wind} = \text{Strong} | \text{PlayTennis} = \text{yes}) = 3/9 = 0.33$$

$$P(\text{Wind} = \text{Strong} | \text{PlayTennis} = \text{no}) = 3/5 = 0.60$$

Aplicandolo a las fórmulas:

$$P(\text{yes}) P(\text{Sunny}|\text{yes}) P(\text{Cool}|\text{yes}) P(\text{High}|\text{yes}) P(\text{String}|\text{yes}) = 0.0053$$

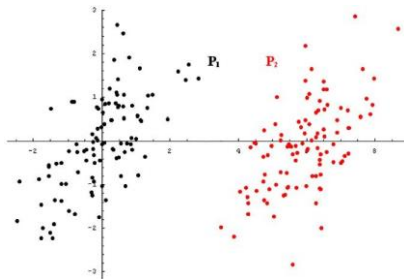
$$P(\text{no}) P(\text{Sunny}|\text{no}) P(\text{Cool}|\text{no}) P(\text{High}|\text{no}) P(\text{String}|\text{no}) = \mathbf{0.0206}$$

⇒ Answer: **PlayTennis = no**

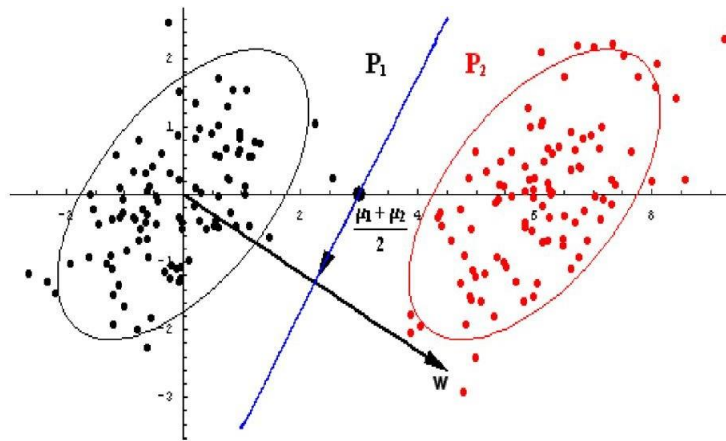
⇒ **Con 79.5% de certeza**

Methods: Fisher's linear discriminant

Fisher's idea is to find the "best linear classifier", that is, to find the hyperplane equation that best separates (in terms of classification error) the two populations.



Methods: Fisher's linear discriminant



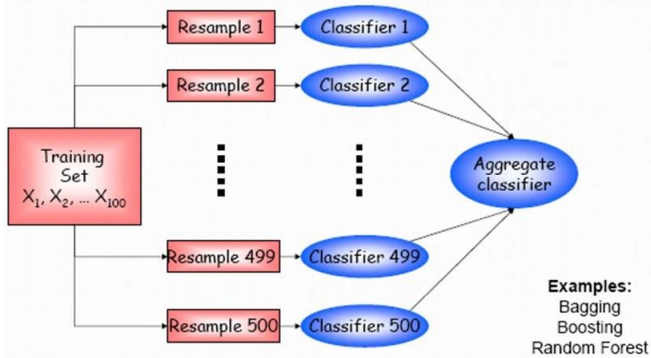
Ensemble methods: bagging, boosting and stacking

A mixture of classifiers is a set $g_1(x), \dots, g_k(x)$ whose estimate of the class Y is a combination of the estimates of each of the classifiers.

- ☐ Hierarchical.
- ☐ Bagging (parallel)
- ☐ Boosting (sequential)
- ☐ Stacking.

Mixing experts

Another component in classification rule:
aggregating classifiers



bias-variance tradeoff

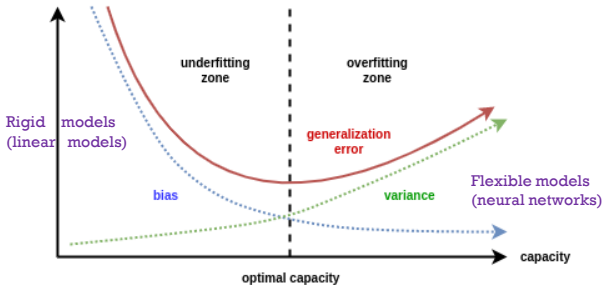
Weak models can have

high bias -> underfitting

Bias = the difference between the model's expected value and the true value of the parameter being estimated

high variance, to be robust -> overfitting

Variance = how much we expect the performance of our model to **vary** as a function of independent resampling of data from the data generating distribution. (Remember that the squared root of variance = standard error)



Stages of Data Mining

1. Goal definition
 - Task to perform: classify, predict, group, ...
2. Collection and understanding of data
 - General characteristics, visualization, quality verification
3. Pre-processing
 - Integration of sources, generation of attributes, selection of attributes, transformations, ...
4. Modeling
 - Machine learning technique to generate / train the model
5. **Evaluation and analysis of results**
 - Performance/error estimation, choice of the best model
6. Using the model
 - Development or integration in an information system/
intelligent agent

Assessment of the Learned Model

Questions

- ❑ How good is the model we have generated for predictions of future instances?
- ❑ Will our predictions be better than a simple random or majority ranking??
- ❑ Default Classifier
 - ❑ Always predict with the mode of training examples (majority class)
 - ❑ **Beware:** Precision Trap in Unbalanced Datasets

Evaluation Design: Data Division

- ❑ **Training** Set

- ❑ Used to generate the model

- ❑ **Test** Set

- ❑ The value or class to predict is known
 - ❑ This value is compared with the prediction of the model to assess its quality

- ❑ **Production** Set

- ❑ The value or class of the instances is not known
 - ❑ It is used in the final production phase with the final model generated

Splitting up your dataset: so that the model doesn't learn it by hard

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

By default 75/25

	X_train	y_train
0	1.0	n
1	4.0	n
	...	
5	5.0	n
6	6.0	n

	X_test	y_test
0	9.0	y
1	1.0	n
2	4.0	n

❑ **Problem:** with imbalanced datasets

❑ Not the same representation of each class

Solution to imbalance 1: Stratified sampling

□ Sampling taking into account the distribution in your dataset

- 100 samples, 80 class 1 and 20 class 2
- Training set: 75 samples, 60 class 1 and 15 class 2
- Test set: 25 samples, 20 class 1 and 5 class 2

Representing in the train and test the same distribution of the original. i.e. 75/25

```
# Total "labels" counts  
y["labels"].value_counts()
```

```
class1    80  
class2    20  
Name: labels, dtype: int64
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, stratify=y)
```

```
y_train["labels"].value_counts()
```

```
y_test["labels"].value_counts()
```

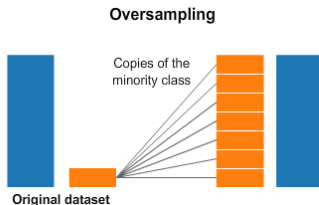
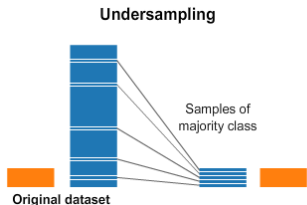
```
class1    60  
class2    15  
Name: labels, dtype: int64
```

```
class1     20  
class2      5  
Name: labels, dtype: int64
```


Solution to imbalance 2: Manually balance

❑ If the sample is big enough: Undersampling

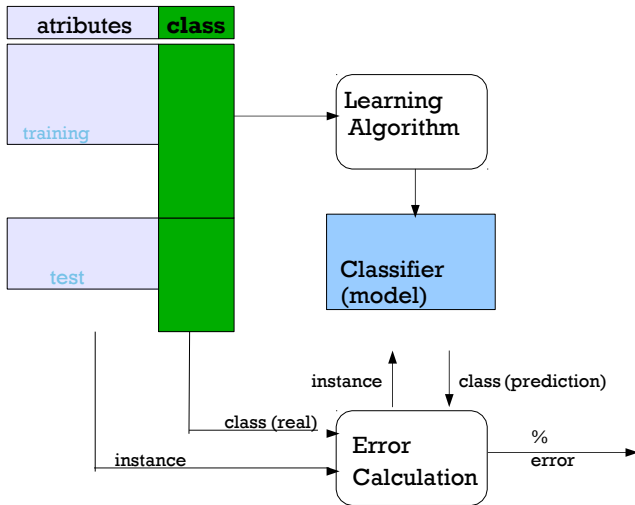
- ❑ Keep all instances of minority class ($N = A$)
- ❑ Randomly select instances to the majority class (keep length = A)
- ❑ Can do in every k-fold
- ❑ Can cause loss of information
 - ❑ **Cluster Centroids:** can cluster the records of the majority class, and do the under-sampling by removing records from each cluster, thus seeking to preserve information



❑ Oversampling

- ❑ duplicate random records from the minority class
- ❑ can cause overfitting
 - ❑ **SMOTE** (Synthetic Minority Oversampling TEchnique) introduce small variations into those copies, creating more diverse synthetic samples
- ❑ these techniques have their weaknesses (there is no free lunch).

Model Evaluation



Confusion Matrix

		Real class	
		positive	negative
Predicted class	positive	true positive (TP)	false positive (FP)
	negative	false negative (FN)	True negative (TN)

Metrics (1)

- *Accuracy*: Total hits

$$accuracy = \frac{TP + TN}{P + N}$$

Metrics (2)

► *Precision:*

$$precision = \frac{TP}{TP + FP}$$

How many of the ones I predict are correct?

► *Recall:* Ratio of true positives

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

How many of the ones I should predict am I predicting?

Metrics (3)

- Specificity: True negatives Rate

$$specificity = \frac{TN}{TN + FP} = \frac{TN}{N}$$

How many of the ones I
exclude are correct?

- False positive ratio(1 - specificity)

$$1 - specificity = \frac{FP}{FP + TN} = \frac{FP}{N}$$

How many of the ones I
should exclude am I
wrongly predicting?

Recap: Model Evaluation

Overfitting

The models are so refined that they describe training instances well but get high error in external examples

Causes

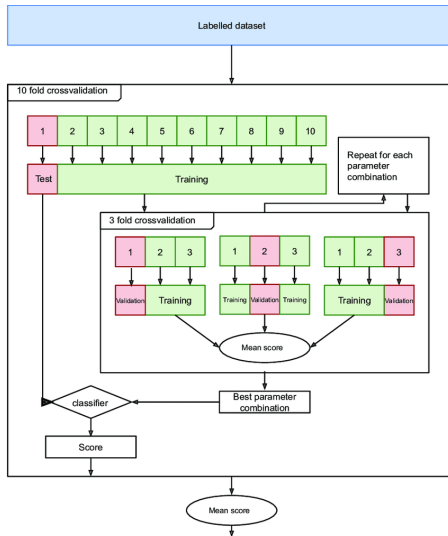
- ❑ few examples
- ❑ training instances are not a representative sample
- ❑ noise in training instances
- ❑ data errors

Cross Validation

K- Cross Validation

- ❑ Divide the complete sample set (E) into k equal parts, E_i
- ❑ Run k times:
 - ❑ Train a model with $E - E_i$ dataset
 - ❑ Compute error with E_i
- ❑ The real error is estimated by taking the mean of the errors
- ❑ **Leave-one out** validation $K = N - 1$

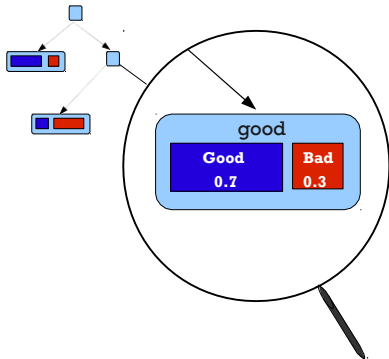
Grid Search: Hyperparameter tuning



Source: Pereira-Kohatsu, J. C., Quijano-Sánchez, L., Liberatore, F., & Camacho-Collados, M. (2019). Detecting and monitoring hate speech in Twitter. Sensors

Prediction Confidence

- ❑ Proportion of examples in training instances
- ❑ Estimation of prediction success
- ❑ Confidence threshold
 - ❑ By default 0.5
 - ❑ Prediction of the majority class

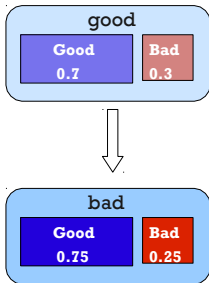


Threshold Selection

Reasons

- Find a better ratio between precision and sensitivity
- Errors have different costs

If $\text{trust}[\text{good}] > 0.75$ then class = **good**
else
class = **bad**



Multiclass classification

		Real class			
		A+	A	B	C
prediction	A+				
	A				
	B				
	C				

- ❑ With N classes, the confusion matrix is always $N \times N$.
- ❑ With sorting scales the error costs are not the same

SCHEMA

- ❑ Introduction: Inductive learning from data
- ❑ Classification techniques
- ❑ Regression techniques

Regression

- In the training instances $\{x_1, x_2, \dots, x_n, y\} = \{X, y\}$ we are interested in predicting the value of y , which is a **continuous value**.
- The regression task consists of finding a function that approximates the values of the variable to be predicted..

$$y \approx f(X, W)$$

Regression

$$\square y \approx f(X, W)$$

- W is the vector of unknown parameters that also determines the model
- the functional form of f is in principle unknown..
 - Knowing the attributes relationship, it can be assumed that the model follows some form of linear, polynomial, exponential equation
 - Use a technique that allows us to approximate f

Regression

- Find the best function f

$$y \approx f(X, W)$$

$$y_i = f(X, W) + \varepsilon_i$$

- It will be the function that minimizes the error in the training data $[\{X_1, y_1\}, \{X_2, y_2\}, \dots, \{X_m, y_m\}]$

$$E = \sum_{j=1}^m \left(y_j - f(x_j, w) \right)^2$$

Classic Regression

- If we specify that f follows a model equation, the goal is to find the vector of parameters $W = \{w_0, w_1, \dots, w_n\}$
- Linear regression

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n + \varepsilon$$

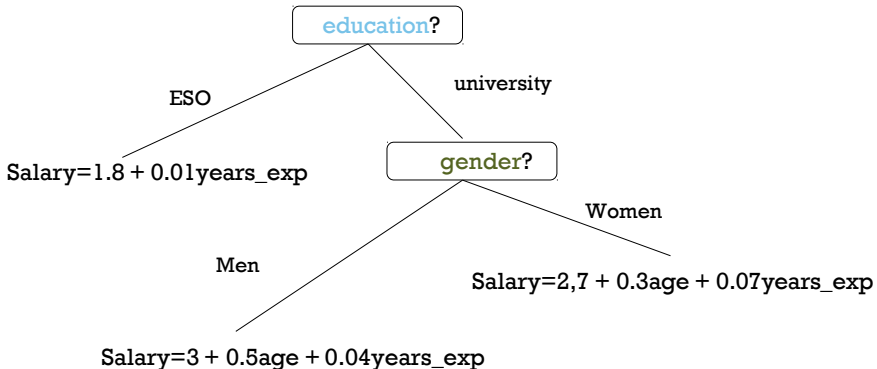
- Exponential regression

$$y = w_0e^{w_1x_1} + \varepsilon$$

Regression trees

- ❑ Decision trees similar to those generated for classification, but with linear regression models for the examples on each leave.
- ❑ Inductive construction: Choose each time the attribute that minimizes the variation of the values of Y in each subset of instances.
- ❑ Stop criterion: When there are few examples in the current node or when there is not much variation in the values of y .

Regression tree



Neural Networks

- They arose as an attempt to reproduce biological models
- Today they are considered one of the best techniques to generate classification and prediction models
- They are able to approximate nonlinear functions
- **Given some parameters there is a way to combine them to predict a certain result.**

Neural Networks : Example

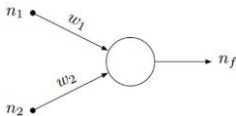
- Knowing the pixels of an image there will be a way to know what number is written,
- Or, knowing the server load of a Data Processing Center (DPC), its temperature and others, there will be a way to know how much they are going to consume
- The problem is that we do not know how to combine them.

Neuronal Networks

- ❑ Neural networks are a model to find that combination of parameters and apply it at the same time.
- ❑ In other words, finding the combination that best fits is "training" the neural network.
- ❑ An already trained network can then be used to make predictions or classifications, that is, to "apply" the combination.

Neural Networks : Example

You are students of a class in which the teacher has not said exactly how he is going to grade the exams. To begin, suppose that you have only taken two exams and you have the grade for each of them and the final.



How do we use a neural network to find out how much each exam is worth? Here the fundamental unit of the neural network will suffice: the perceptron. A perceptron is an element that has several inputs with a certain weight each. If the sum of these inputs for each weight is greater than a certain number, the output of the perception is a one. If it is less, the output is a zero.

Neural Networks: Example

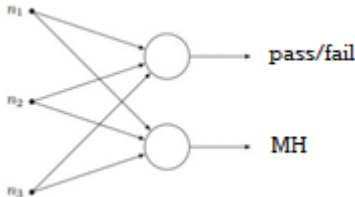
In our example, the inputs would be the two test scores. If the exit is one (that is, the sum of the grades with their corresponding weight is greater than five), it is a pass. If it is zero, fail. Weights are what we have to find with training. In this case, our training will consist of starting with two random weights (for example, 0.5 and 0.5, the same weight for each exam) and seeing what result the neural network gives for each student. If it fails in any case, we will adjust the weights little by little until everything is well adjusted.

Neural Networks : Example

For example, if a student with a very good grade in the second exam has failed the course, we will lower the weight of the second exam because it clearly does not influence too much.

Little by little we will end up finding the weights that adjust to the grades that the teacher put. The idea of the adjustment or feedback is to adapt the network to the "hidden" information that the data that we pass to it has, so that it learns.

Neural Networks : Example



Maybe we want to complicate it more, putting more tests (more input nodes) or wanting to get more results, such as a perceptron whose output is one if the student gets MH.

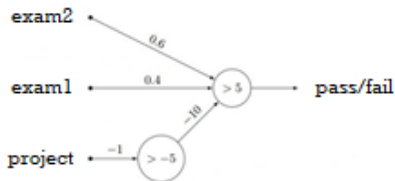
Multiplying the power: multilayer networks

It turns out that there is a curious situation. There are two students who have the same grade in the exams, two 10s, but one has a 7 in the project and the other a 4. The one with 7 has passed the course, but the one with 4 has not. There is a student who has a 10 in the project and 4.99 in the two exams and is failing.

We can try to train a neural network like the one before but in this situation it is not going to work well. It seems that the project grade does not influence unless you fail it, in which case you directly fail. It is a filter, a one or a zero that we have to add in the neural network before being able to give the result of passed or failed in the course.

Multiplying the power: multilayer networks

We need more layers. We need an intermediate perceptron that tells us if the project is passed or not, and count that in the output perceptron:



The first perceptron looks at whether the grade of the project multiplied by negative one is greater than negative five (or, what is the same, if the note is less than five). If it is, then your output is one. By multiplying it by minus ten at the input of the second perceptron, you will always force a fail. If the project is passed, the output of the first perceptron will be 0 and will not affect the average of the exams.

What are layers for?

To add information that was not there before. We take the input data, explore it, and extract the characteristics that best help us understand what is happening.







- ❑ The good news come when, during the learning process, each layer "learns" to find and detect the characteristics that best help classify the data.
- ❑ In our example, during adjustment the first layer would learn that students with failed project fail the course.
- ❑ If we took a network to detect numbers written by hand, the hidden layers may learn to detect straight or curved lines that serve to decide whether we are dealing with a 1 or an 8, for example.

Beyond perceptrons: sigmoids, deep networks, and convolutional networks

- ❑ So far we have focused on simplifications to get a good understanding of neural network concepts.
- ❑ In reality, things get quite complicated. For example, perceptrons are substituted by other "neurons" with a softer behavior, using functions such as the sigmoid.
- ❑ The idea is that small changes in the weights cause small changes in the output of the network, in order to make learning "easier".

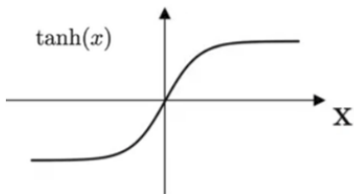
Name	Graphic	Function
------	---------	----------

Activation functions

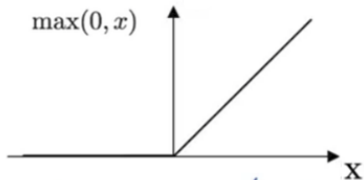
Linear		$f(x) = x \cdot w + b$
Binary step		$\begin{aligned} &\text{if } x \geq 0, && \text{then } f(x) = 1, \\ &\text{if } x < 0, && \text{then } f(x) = 0, \end{aligned}$
Piecewise linear		$\begin{aligned} &\text{if } x \geq x_{\text{upper}}, && \text{then } f(x) = 1, \\ &\text{if } x_{\text{upper}} > x > && \text{then } f(x) = x \cdot \frac{x_{\text{upper}} - x_{\text{lower}}}{x_{\text{upper}} - x_{\text{lower}}} + x_{\text{lower}}, \\ &\text{if } x \leq x_{\text{lower}}, && \text{then } f(x) = 0, \end{aligned}$
Sigmoid		$f(x) = \frac{1}{1 + e^{-x}}$ interval (0,1)
Gaussian		$f(x) = e^{-x^2},$ interval (0,1]
Hyperbolic tangent		$f(x) = \frac{2}{1 + e^{-2x}} - 1,$ interval [-1,1]

Activation functions

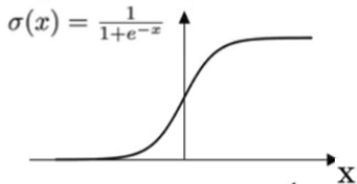
Hyper Tangent Function



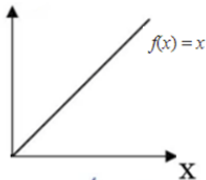
ReLU Function



Sigmoid Function



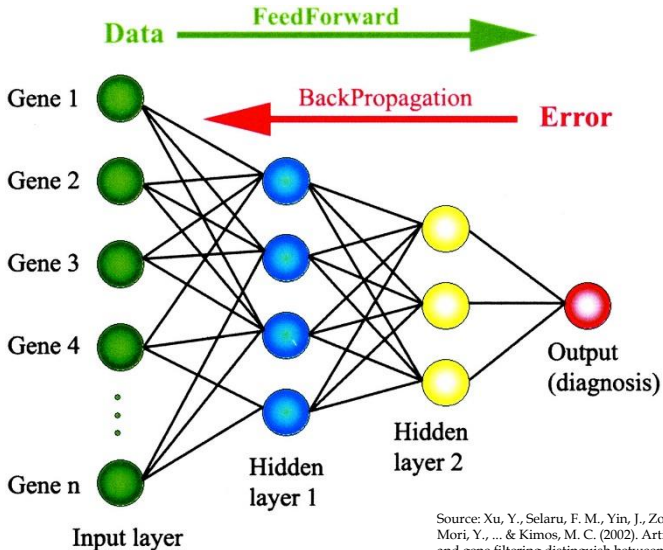
Identity Function



Artificial Neural Networks : Summary

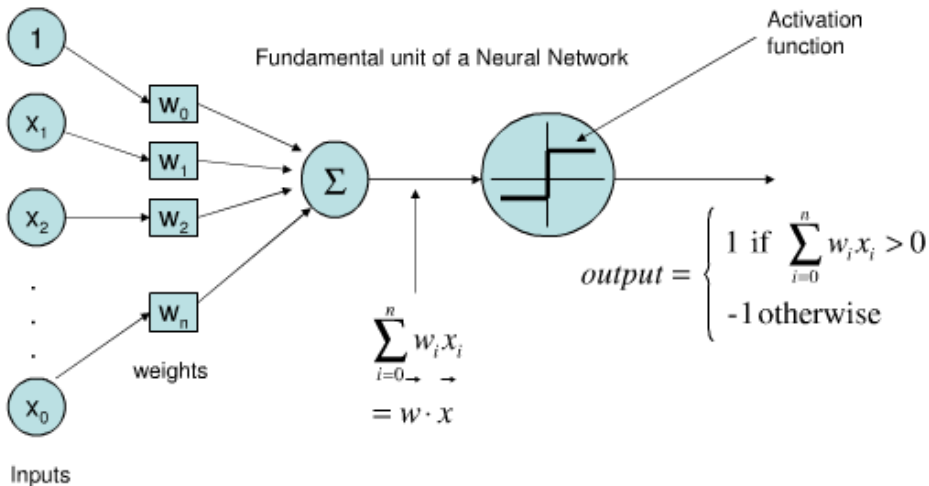
- ❑ Set of artificial neurons connected to each other by a set of connections
- ❑ Connections have associated real numbers called **weights**
- ❑ Generally neurons are distributed in **layers** of different levels
- ❑ Each neuron receives **input signals** from the outside or from other neurons
- ❑ The **output signal** is calculated as a **function** of the inputs

Neural Networks



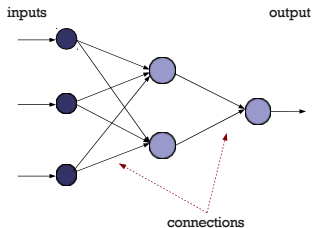
Source: Xu, Y., Selaru, F. M., Yin, J., Zou, T. T., Shustova, V., Mori, Y., ... & Kimos, M. C. (2002). Artificial neural networks and gene filtering distinguish between global gene expression profiles of Barrett's esophagus and esophageal cancer. *Cancer Research*.

Artificial Neuron: inside



Model Construction

- ❑ Structure of the Network:
Number of neurons, number of layers, connections, etc.
- ❑ Network Learning: Iterative modification of the weights so that the network output is correct



Regression Evaluation

- ❑ A model should be better than the default model, which in regression returns the mean of the target attribute.
- ❑ The calculation of the error on the training data tells us how the model fits the data
- ❑ Cross validation
 - ❑ Same scheme as the classification evaluation
 - ❑ This error calculation estimates how well the model generalizes and adapts to future data

Regression metrics

□ **Mean Squared Error:**

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_i)^2$$

□ **Root Mean Squared Error**

$$RMSE = \sqrt{MSE}$$

□ **Mean Absolute Error**

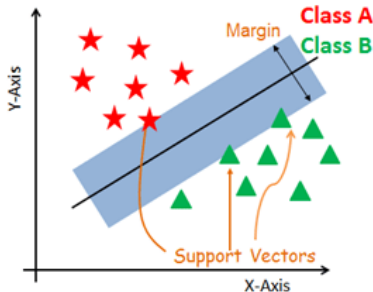
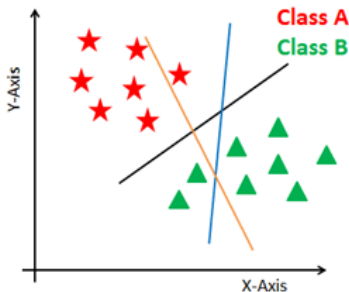
$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \bar{y}_i|$$

Appendix

SVM

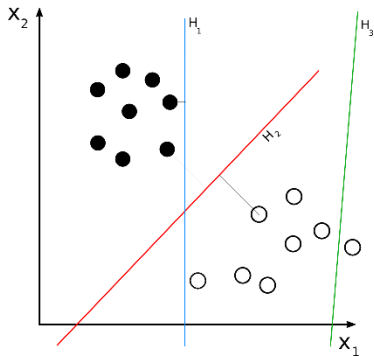
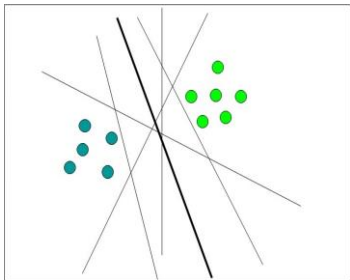
- support vector machines (svm) models are a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall

The parameters are adjusted to maximize the margin solving the problem.



SVM

look for the hyperplane that cuts the sample containing the largest margin

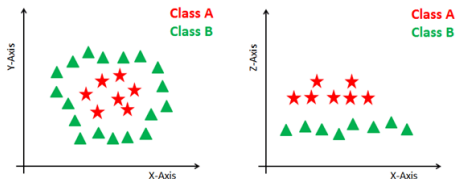


SVM

- The simplest way to separate is by a straight line, a straight plane, or an N-dimensional hyperplane.

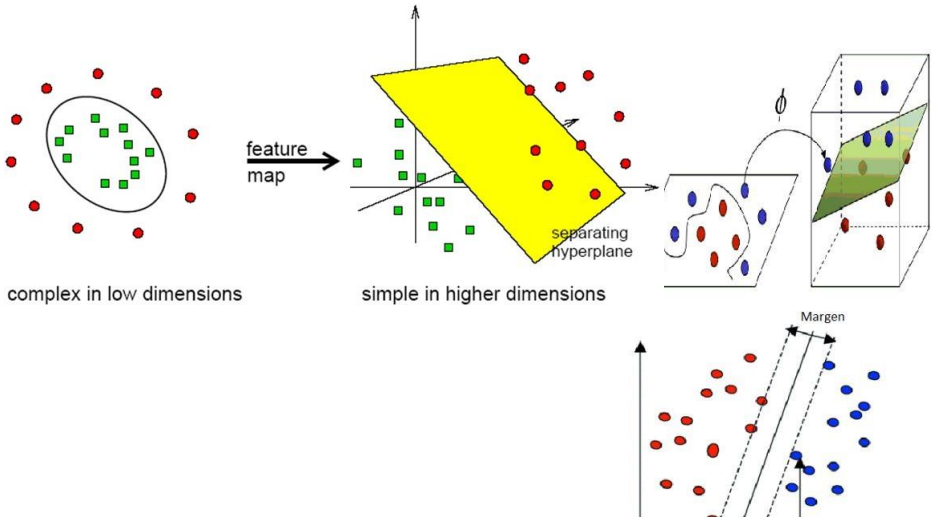
Unfortunately, the universes to study do not usually appear in idyllic two-dimensional cases as in the previous example, but rather an SVM algorithm must deal with a) more than two predictor variables, b) non-linear separation curves, c) cases where the data sets cannot be completely separated, d) classifications in more than two categories.

Due to the computational limitations of linear learning machines, they cannot be used in most real world applications. Representation by means of Kernel functions offers a solution to this problem, projecting the information to a larger space of characteristics, which increases the computational capacity of linear learning machines. That is, we will map the input space X to a new feature space of higher dimensionality (Hilbert).



SVM

Separation may be easier in higher dimensions



Bootstrapping

- ❑ Samples of size B from the original dataset **with replacement**



- ❑ Good properties:
 - ❑ drawn directly from the data distribution and independently from each other
- ❑ Conditions -> N has to be big enough:
 - ❑ To capture most of the complexity of the underlying distribution (**representativity**)
 - ❑ Compared to B so that samples are not too much correlated (**independence**)

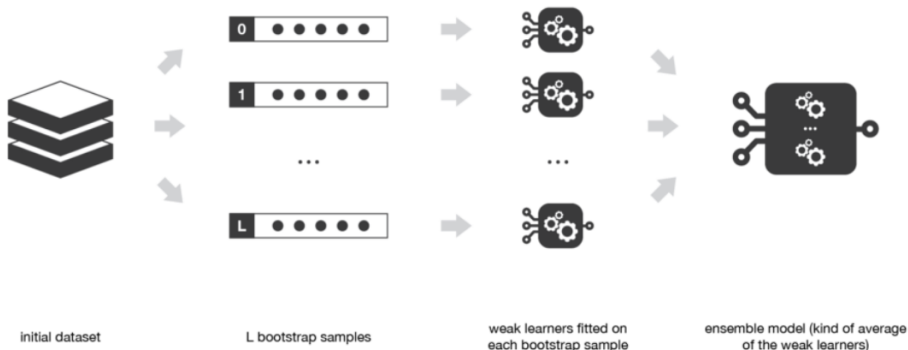
Bagging

- ❑ Idea: use bootstrapping samples & fit several “independent” models and “average” their predictions in order to obtain a model with a lower variance -> more robust -> less overfitting

- ❑ Aggregations:

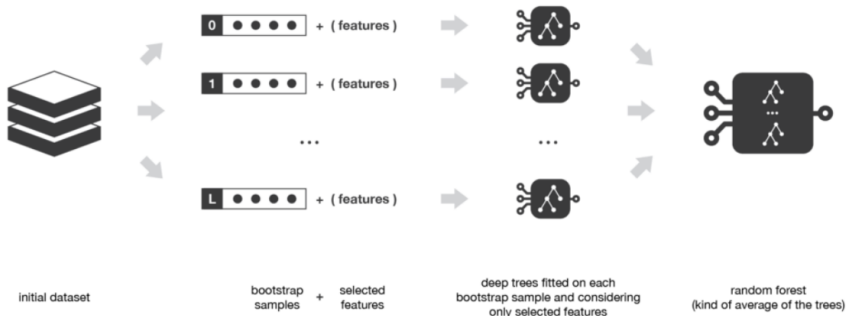
- ❑ Simple/weighted average (regression)
- ❑ Majority/weighted vote (classification)

- ❑ Works with parallelization



Random forests

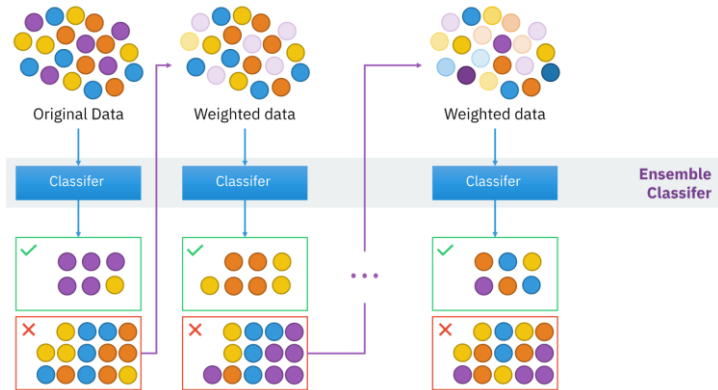
- ❑ Idea: use **bagging** on deep trees -> lower variance -> more robust -> less overfitting
 - ❑ Deep trees = lot of depths, low bias but high variance
 - ❑ Shallow trees = few depths, less variance but higher bias
- ❑ Difference with bagging
 - ❑ **Sample not only on observations but also on features**
 - ❑ Advantages
 - ❑ Trees do not look at the exact same information to make their decisions -> reduces correlation
 - ❑ makes the decision making process more robust to missing data



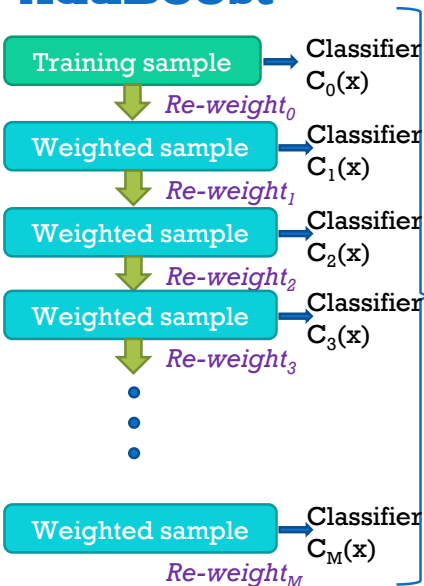
Source: Joseph Rocca. Ensemble methods: bagging, boosting and stacking. 2019

Boosting

- Mainly focused on reducing bias (although it can reduce variance too)
- Sequential
- Idea: fitting sequentially models in a very adaptative way
 - giving more importance to observations in the dataset that were badly handled by the previous models in the sequence
 - Each new model **focus its efforts on the most difficult observations** to fit up to now



Boosting (ii): Adaptive boosting = AdaBoost



- AdaBoost re-weights events misclassified by previous classifier:

$$re - weight_i = \frac{1 - Error}{Error}$$

$$\text{with } Error = \frac{\text{misclassified-instances}}{N}$$

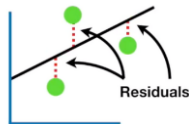
- AdaBoost weights the classifiers also using the error rate of the individual classifier according to:

$$\square y(x) = \sum_i^M \text{classifier} \log(re -$$

Boosting (iii): Gradient boosting

Gradient boosting updates values of the observations at each iteration. Weak models are trained to fit the pseudo-residuals that indicate the direction to find the correct prediction of the current ensemble model and lower the error

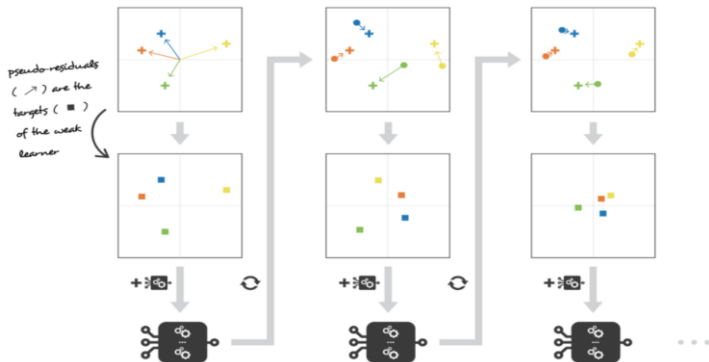
The term pseudo-residual is based on linear regression where the difference between the observed and the predicted values are residuals



+ train a weak model and aggregate it to the ensemble model

update the pseudo-residuals considering predictions of the current ensemble model

+ dataset values
● predictions of the current ensemble model
■ pseudo-residuals (targets of the weak learner)



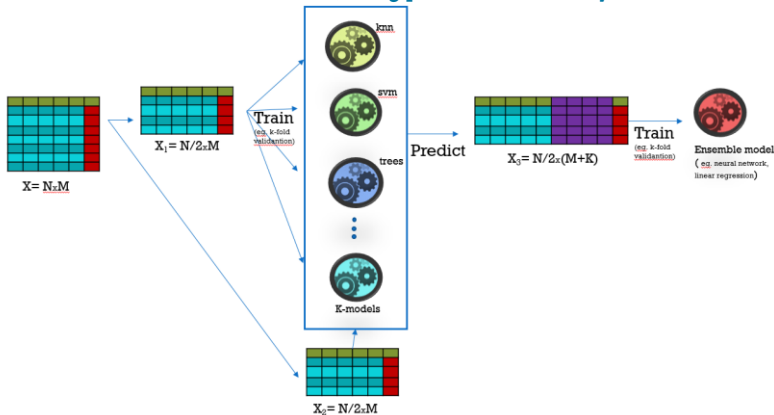
Stacking

Differences with bagging and boosting

- different learning algorithms are combined, i.e before combine models that use the same technique now we combine different techniques
- Trains a meta-model for the combination, before we used deterministic algorithms

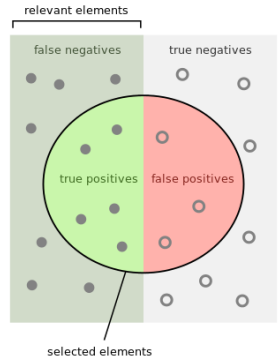
Method: split the training data in two folds (or k-fold)

- Elegimos L modelos débiles y ajustamos a los datos del primer pliegue
- for each of the L weak models, make predictions for observations in the second fold
- fit the meta-model on the second fold, using predictions made by the weak learners as inputs



Metrics (4)

- F1 (**F-score**) . It is used in determining a single weighted precision and recall value.



How many selected items are relevant?

Precision =



How many relevant items are selected?

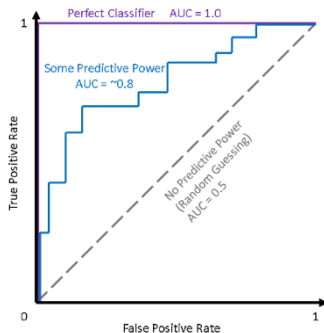
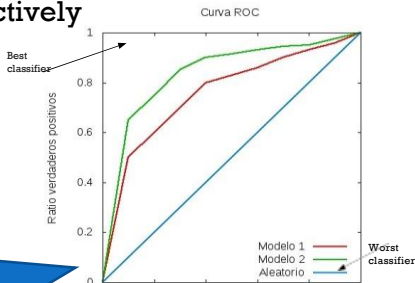
Recall =



$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}.$$

ROC Curve: Receiver Operating Characteristic

- ▶ They are curves in which the sensitivity is presented as a function of the false positives for different cut-off points
- ▶ TPR measures the extent to which a classifier is able to detect positive cases correctly, out of all positive cases. The FPR defines how many positive results are incorrect out of all negative cases.
- ▶ An ROC space is defined by FPR and TPR as x and y axes respectively



Does not
depend
on
threshold