

Grado en ingeniería informática
Inteligencia Artificial 2020/2021

Introducción al Machine Learning

Lara Quijano Sánchez



Universidad Autónoma
de Madrid

ESQUEMA

- ❑ Introducción: Aprendizaje inductivo, a través de datos
- ❑ Técnicas de Clasificación
- ❑ Técnicas de Regresión

Motivación

- ❑ La sociedad de la información genera una cantidad explosiva de datos
- ❑ Queremos automatizar el proceso de análisis de esos datos
- ❑ Introducimos conocimiento a través de ejemplos para:
 - ❑ personalizar sistemas centrados en el usuario
 - ❑ modificar el comportamiento de un agente inteligente
- ❑ El desarrollo de software es un cuello de botella

¿Qué se puede aprender?

- ☐ Diagnósticos médicos
- ☐ Reconocimiento de imágenes
- ☐ Finanzas: aprobación de créditos, predicción bursátil, ...
- ☐ Internet: filtros spam, tendencias en redes sociales (influencers), inferencia de gustos
- ☐ Bioinformática: Estructura de proteínas, función de genes, ...
- ☐ Textos: fake news, bots, opinión, mensajes de odio
- ☐ Juegos y deportes: predicción de jugadas, movimientos, resultados
- ☐ Analíticas para el aprendizaje (learning analytics)
- ☐ Criminología: denuncias falsas, predicción de reincidencia, clusterización de delitos

Ciencia de datos

Hipótesis de la ciencia de datos

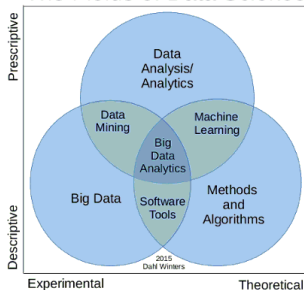
Si un modelo describe bien el concepto meta en un número suficientemente grande/significativo de ejemplos, también describirá el concepto meta en futuros ejemplos

Ciencia de Datos

- ❑ Proceso que extrae relaciones, tendencias o patrones que existen en un grupo grande de datos.
- ❑ Unión entre muchas ramas de ciencias
- ❑ Es la base del análisis de datos en el **Big Data**

Ciencia de Datos: Ramas de ciencias

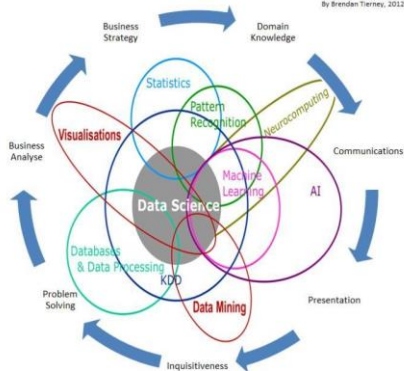
The Fields of Data Science



Ciencia de Datos: Ramas de ciencias

Data Science Is Multidisciplinary

By Brendan Tierney, 2012



Ciencia de Datos

La estadística, a través del análisis de datos (minería de datos), engloba todos los métodos utilizados para extraer información útil de los datos y tomar decisiones.

Etapas de un proyecto Data Science

1. Definición del objetivo

Tarea a realizar: clasificar, predecir, agrupar, ...

2. Recolección y comprensión de los datos

Características generales, visualización, verificación de la calidad

3. Pre-procesado

Integración de fuentes, generación de atributos, selección de atributos, transformaciones,...

4. Modelado

Técnica de aprendizaje automático para generar/entrenar el modelo

5. Evaluación y análisis de resultados

Estimación del rendimiento/error, elección del mejor modelo

6. Uso del modelo

Desarrollo o integración en un sistema informático/agente inteligente

Tareas de la Ciencia de Datos

☐ Clasificación

- ☐ Determinar para una situación/elemento en qué **categoría** se encuentra

☐ Regresión - Predicción

- ☐ Predecir para una situación/elemento cuál es su **valor** aproximado

☐ Agrupamiento - Clustering

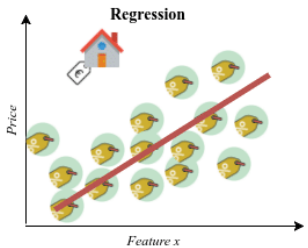
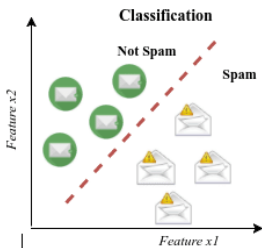
- ☐ Establecer en un conjunto de **elementos** cuáles son **semejantes** entre sí

☐ Conjuntos Frecuentes

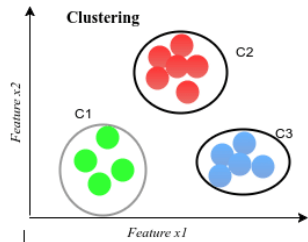
- ☐ Determinar en una lista de sucesos qué **elementos se repiten** con frecuencia

Dependiendo del conjunto de datos existente

- ☐ Aprendizaje supervisado (todos los datos etiquetados)
- ☐ Aprendizaje no supervisado (sin datos etiquetados)
- ☐ Aprendizaje semi-supervisado (algunos datos etiquetados)



Supervised learning



Unsupervised learning

REPRESENTACIÓN DATOS

- **Dataset:** Tabla que contiene los datos que vamos a analizar.

id	edad	estado civil	ahorros	formación	trabajo	casa	cantidad	clase
1	35	soltero	7,000	básica	Cualif.	propia	50K	bueno
2	23	casado	2,000	f.p	Cualif.	alquilada	70K	bueno
3	30	casado	1,000	básica	No-cual	propia	60K	malo
4	26	soltero	15,000	univ.	autono	propia	120K	bueno
5	50	divorciado	3,500	univ.	No-cual	alquilada	40K	bueno
6	43	soltero	N.S.	básica	autono	N.S	30K	malo
7	31	divorciado	28,000	máster o +	No-cual	propia	90K	malo
8	33	casado	N.S.	univ.	No-cual	alquilada	30K	bueno
9	40	soltero	11,000	máster o +	cual	propia	100K	bueno

CONCEPTOS GENERALES

- **Atributo:** Característica que describe parcialmente a los elementos.

edad	estado civil	ahorros	formación	trabajo	casa	cantidad	clase
35	soltero	7,000	básica	Cualif.	propia	50K	bueno
23	casado	2,000	f.p	Cualif.	alquilada	70K	bueno
30	casado	1,000	básica	No-cual	propia	60K	malo
26	soltero	15,000	univ.	autono	propia	120K	bueno
50	divorciado	3,500	univ.	No-cual	alquilada	40K	bueno
43	soltero	N.S.	básica	autono	N.S	30K	malo
31	divorciado	28,000	máster o +	No-cual	propia	90K	malo
33	casado	N.S.	univ.	No-cual	alquilada	30K	bueno
40	soltero	11,000	máster o +	cual	propia	100K	bueno

CONCEPTOS GENERALES

- **Instancia:** Un elemento definido por los valores de sus atributos

edad	estado civil	ahorros	formación	trabajo	casa	cantidad	clase
35	soltero	7,000	básica	Cualif.	propia	50K	bueno
23	casado	2,000	f.p	Cualif.	alquilada	70K	bueno
30	casado	1,000	básica	No-cual	propia	60K	malo
26	soltero	15,000	Univ.	autono	propia	120K	bueno
50	divorciado	3,500	univ.	No-cual	alquilada	40K	bueno
43	soltero	N.S.	básica	autono	N.S	30K	malo
31	divorciado	28,000	máster o +	No-cual	propia	90K	malo
33	casado	N.S.	univ.	No-cual	alquilada	30K	bueno
40	soltero	11,000	máster o +	cual	propia	100K	bueno

Conceptos Generales

- **Clase:** Subconjuntos disjuntos (categorías) en los que se quiere dividir el conjunto de instancias.
- En tareas de clasificación es el atributo a predecir

edad	estado civil	ahorros	formación	trabajo	casa	cantidad	clase
35	soltero	7,000	básica	Cualif.	propia	50K	bueno
23	casado	2,000	f.p	Cualif.	alquilada	70K	bueno
30	casado	1,000	básica	No-cual	propia	60K	malo
26	soltero	15,000	univ.	autono	propia	120K	bueno
50	divorciado	3,500	univ.	No-cual	alquilada	40K	bueno
43	soltero	N.S.	básica	autono	N.S	30K	malo
31	divorciado	28,000	máster o +	No-cual	propia	90K	malo
33	casado	N.S.	univ.	No-cual	alquilada	30K	bueno
40	soltero	11,000	máster o +	cual	propia	100K	bueno

Conceptos Generales

- ▶ **Ejemplo positivo** de una clase: instancia que pertenece al subconjunto definido por la clase
- ▶ **Ejemplo negativo** de una clase: instancia que no pertenece al subconjunto definido por la clase
- ▶ **Valores omitidos**: Se desconoce el valor del atributo en una instancia.
 - ❑ omisión en el mecanismo de recolección de datos: encuestas
 - ❑ el valor no existe en la realidad

Etapas de un proyecto Data Science

1. Definición del objetivo

Tarea a realizar: clasificar, predecir, agrupar, ...

2. Recolección y comprensión de los datos

Características generales, visualización, verificación de la calidad

3. Pre-procesado

Integración de fuentes, generación de atributos, selección de atributos, transformaciones,...

4. Modelado

Técnica de aprendizaje automático para generar/entrenar el modelo

5. Evaluación y análisis de resultados

Estimación del rendimiento/error, elección del mejor modelo

6. Uso del modelo

Desarrollo o integración en un sistema informático/agente inteligente

Ejemplo

❑ Importar conjunto de datos e imprimir las primeras filas

❑ Fuente: **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

Ejemplo

□ Mirar todas las columnas, imprimir los tipos

```
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
```

Ejemplo

- ❑ Generar estadísticas básicas sobre el dataframe (cada columna)
 - ❑ Media
 - ❑ Desviación Estándar
 - ❑ Cuartiles
- ❑ Fuente
 - ❑ 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

Por qué son importantes los tipos?

```
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

❑ Fuente

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

❑ Siempre verificamos qué tipos tenemos

❑ E.j. es posible que hayamos leído un integer como un string y tengamos que convertirlo antes de comenzar a trabajar

```
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
```

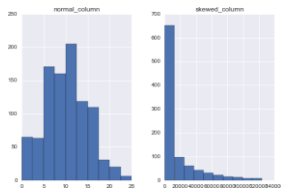
```
df["C"] = df["C"].astype("float" )
print(df.dtypes)
```

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

Imprimir los datos para comprender ~ verificar si se cumplen las suposiciones

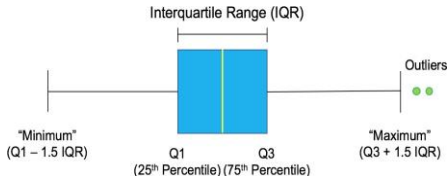
❑ Crear histogramas

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



❑ Boxplots – identificar outliers

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

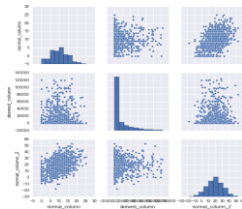


❑ Observar cómo las diferentes características del conjunto de datos interactúan entre sí.

❑ Paring distributions

❑ Útil para ver si varias columnas están correlacionadas entre sí

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



Etapas de un proyecto Data Science

1. Definición del objetivo

Tarea a realizar: clasificar, predecir, agrupar, ...

2. Recolección y comprensión de los datos

Características generales, visualización, verificación de la calidad

3. Pre-procesado

Integración de fuentes, generación de atributos, selección de atributos, transformaciones,...

4. Modelado

Técnica de aprendizaje automático para generar/entrenar el modelo

5. Evaluación y análisis de resultados

Estimación del rendimiento/error, elección del mejor modelo

6. Uso del modelo

Desarrollo o integración en un sistema informático/agente inteligente

¿Cuándo preprocesamos?

- ❑ Después del análisis exploratorio de datos = idea de lo que quiere hacer con su conjunto de datos
- ❑ Luego -> Preparar datos para modelar. P.ej:
 - ❑ Limpieza: si le faltan valores (NaN), también debe preprocesar
 - ❑ Outliers: puntos de datos que difieren significativamente de otras observaciones
 - ❑ El modelado en ML requiere entradas numéricas, es decir, si tiene variables categóricas, necesita preprocesar
 - ❑ Transformar variables existentes
 - ❑ Crear nuevas variables a partir de las existentes
- ❑ Proceso iterativo
- ❑ Requiere conocimiento de dominio / datos

Missing data (Datos faltantes)

❑ Si el conjunto de datos es lo suficientemente grande y representativo

❑ Podemos permitirnos perder algunas filas

❑ Eliminar todos los NaN

```
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: descartar filas

❑ Si el conjunto de datos es lo suficientemente grande y representativo

❑ Podemos permitirnos perder algunas filas

❑ descartar filas específicas

```
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: descartar columnas

❑ Si el conjunto de datos es lo suficientemente grande y representativo

❑ Podemos permitirnos perder algunas filas

❑ Descartar un columna particular especialmente si faltan todos o la mayoría de los valores

```
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: descartar sólo algunas filas

- ❑ Si el conjunto de datos es lo suficientemente grande y representativo

- ❑ Podemos permitirnos perder algunas filas

- ❑ Descartar filas con NaN de una columna particular

```
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: contemos primero

❑ ¿Cuántos valores perdidos tenemos? :

❑ ¿Qué valores son los válidos, el cero es ok? Negativos? NaN?. Eg. Edad

```
# 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: remplazar

❑ Si el dataset NO es suficientemente grande

❑ NO podemos permitirnos perder algunas filas

❑ Aproximamos el valor de los datos faltantes

❑ Con la mediana, media, moda de la columna, etc.


❑ Un valor constante que tiene significado dentro del dominio, como 0, distinto de todos los demás valores.

❑ Un valor de otro registro seleccionado al azar.

❑ Un valor estimado por otro modelo predictivo.

```
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

¿Qué es la estandarización?

- ❑ Transformamos datos continuos para que parezcan distribuidos normalmente
- ❑ Algunos modelos asumen datos distribuidos normalmente. Si no es así, se arriesga a sesgar su modelo
- ❑ ¿Qué podemos hacer, aplicado a datos numéricos continuos?
 - ❑ Normalización de registros y escalado de funciones
- ❑ Cuando estandarizamos:
 - ❑ Supuestos de linealidad: Modelos que operan en un espacio lineal: ej. k-nn, regresión lineal, agrupamiento de k-medias ... Debería tener datos en un espacio lineal
 - ❑ Posibles problemas / sesgo cuando las características del conjunto de datos tienen una gran variación
 - ❑ P.ej. Una característica que tiene un orden de magnitud o más que otras características
 - ❑ Las características del conjunto de datos son continuas y están en diferentes escalas.
 - ❑ P.ej. Conjunto de datos relacionados con la altura y el peso. Para compararlos, deben estar en el mismo espacio lineal -> estandarizados

Normalización Logarítmica

❑ Usar en columnas con alta varianza

❑ Ej: En una predicción de knn, una variable con alta varianza puede hacer que la precisión de su modelo disminuya

❑ Aplica transformación logarítmica

❑ Logaritmo natural usa la constante e (2.718). Ej $e^{3.4} = 30$

❑ Captura cambios relativos, la magnitud del cambio y mantiene todo en el espacio positivo.

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
```

Escalado de features/características

- ❑ Si queremos comparar características en diferentes escalas
- ❑ Asumiendo modelos con características lineales
- ❑ Encuentra la media de los datos y centra/transforma la distribución en función Ej. media de 0, varianza de 1
- ❑ Se transforma en una distribución aproximadamente normal

Baja varianza en columnas
Alta varianza **entre** columnas

Usar un método estándar
permite repetir el proceso
fácilmente. Ej test/train set

```
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

```
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

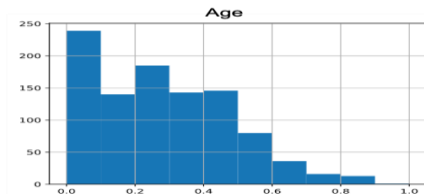
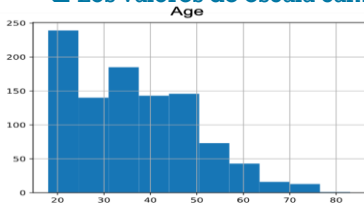
```
print(df.var())
```

col1	0.128958
col2	4.055833
col3	3.526667
dtype:	float64

Normalización -> min-max scaling

❑ Datos escalados entre un mínimo y un máximo

❑ Los valores de escala cambiarán pero la distribución no



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

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

Outliers

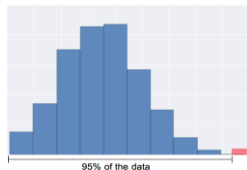
- ❑ Datapoints lejos de la mayoría de sus datos
 - ❑ Debido a errores o ocurrencias raras genuinas
 - ❑ Eliminar estos valores / tratarlos por separado => alto impacto en los modelos
- ❑ Incluso después de la transformación, los datos pueden estar muy sesgados

❑ Soluciones:

❑ Quantile based detection

- ❑ Eliminar el 5% superior / inferior de los datos
- ❑ Siempre elimina los datos. incluso si no son valores atípicos

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



❑ Standard deviation based detection

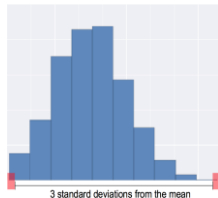
- ❑ Solo elimina los datos que estén a 3 stds de la media
- ❑ Solo elimina los valores atípicos verdaderos

```
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 (Ingeniería de características)

- ☐ Creación de nuevas funciones basadas en funciones existentes
- ☐ Conocimiento de las relaciones entre características
- ☐ Extraer y expandir datos
- ☐ Depende del conjunto de datos

user	fav_color
1	blue
2	green
3	orange

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

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

Id	Date
4	July 302011
5	January 292011
6	February 052011

Codificación de variables categóricas

```
user subscribed fav_color
0          y         blue
1          n         green
2          n         orange
3          y         green
```

Codificación de variables binarias - 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)
```

Codificación de variables binarias - 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
```

Codificación One-hot

fav_color
blue
green
orange
green

fav_color_enc
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]
[0, 1, 0]

Valores: [blue, green, orange]

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

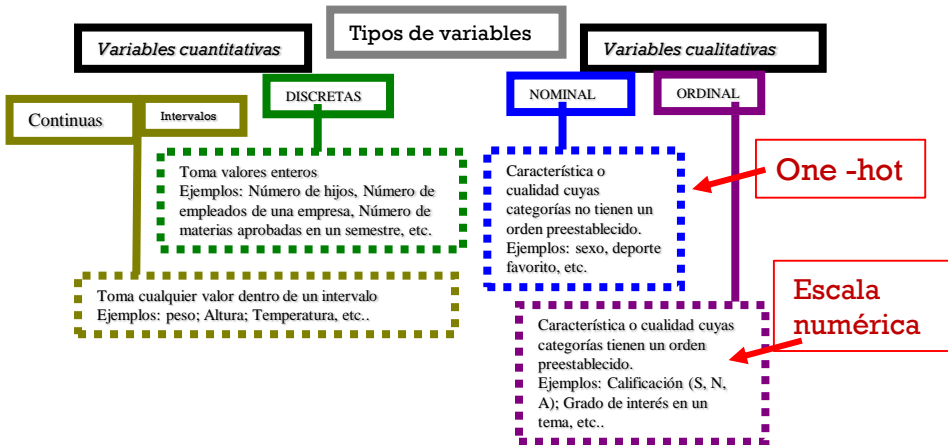
```
print(users["fav_color"])
```

```
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
```


Tipos de variables para codificar



Engineering características numéricas

- ❑ Agregar un conjunto de números para usar en lugar de funciones
 - ❑ Reducir la dimensionalidad del espacio de características
 - ❑ No necesitamos muchas características muy similares (distancia) entre sí

```
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 Fechas

```
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)
```

Convertir a tipo fecha

Coger día, mes, años, calcular edad, 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

Creando nuevas variables

- ☐ Proporción/Ratio

- ☐ N° de veces que aparece una palabra / síntoma respecto a la duración / tiempos del documento en el médico

- ☐ Frecuencia

- ☐ N° de veces que aparece una palabra / síntoma

- ☐ Binario

- ☐ ¿Aparece una palabra / síntoma?

- ☐ Diferencia entre eventos

- ☐ Datos exógenos

- ☐ Ciudad, clima, datos económicos....

Selección de características

- ☐ Seleccionar características que usar para modelar
- ☐ No crea nuevas características/columnas
- ☐ Mejora el rendimiento del modelo
- ☐ Redundancias:
 - ☐ Eliminar características ruidosas
 - ☐ Por ejemplo, tener el nombre de la ciudad + latitud y longitud
 - ☐ Eliminar características correlacionadas
 - ☐ Por ejemplo, raza de perros y perros
 - ☐ Quitar características duplicadas
- ☐ Características correlacionadas:
 - ☐ Correlación estadística: las características se mueven juntas direccionalmente
 - ☐ Los modelos lineales asumen independencia de características
 - ☐ Coeficiente de correlación de Pearson (1 / -1 cor; 0 no cor)

Selección de características

```
# 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

Reducción de dimensionalidad

- ❑ Método de aprendizaje no supervisado
- ❑ Combina / descompone un espacio de características
 - ❑ De forma lineal o no lineal
- ❑ Extracción de características: aquí la usaremos para reducir el espacio de características
- ❑ Ej:
 - ❑ Principal component analysis (PCA)
 - ❑ Transformación lineal para reducir features/características en un espacio donde no están correlacionadas
 - ❑ El espacio de características se reduce y la variación se captura de manera significativa **combinando características en componentes**
 - ❑ Captura tanta varianza como sea posible en cada componente
 - ❑ Usar cuando tengamos muchas features y no haya buenas candidatas para la eliminación
 - ❑ LASSO

Etapas de un proyecto Data Science

1. Definición del objetivo

Tarea a realizar: clasificar, predecir, agrupar, ...

2. Recolección y comprensión de los datos

Características generales, visualización, verificación de la calidad

3. Pre-procesado

Integración de fuentes, generación de atributos, selección de atributos, transformaciones,...

4. Modelado

Técnica de aprendizaje automático para generar/entrenar el modelo

5. Evaluación y análisis de resultados

Estimación del rendimiento/error, elección del mejor modelo

6. Uso del modelo

Desarrollo o integración en un sistema informático/agente inteligente

ESQUEMA

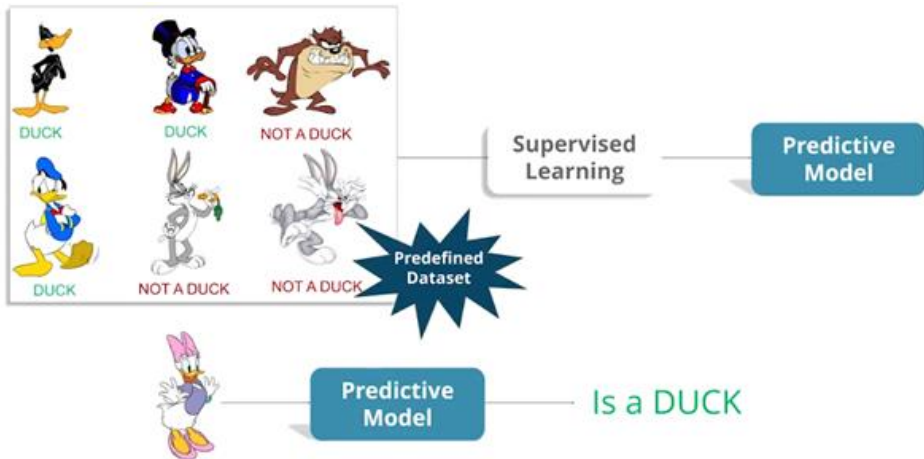
☐ Introducción

☐ Técnicas de Clasificación

☐ Técnicas de Regresión

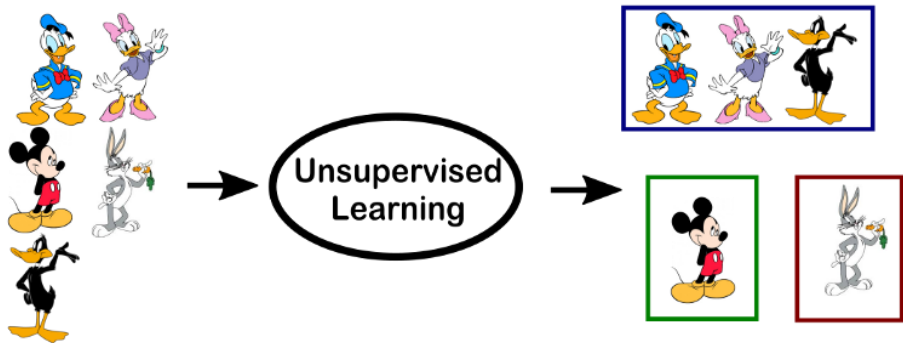
El problema de clasificación

- ▶ Cuando hay etiquetas, es supervisado =
Identifica/clásica basándote en ejemplos



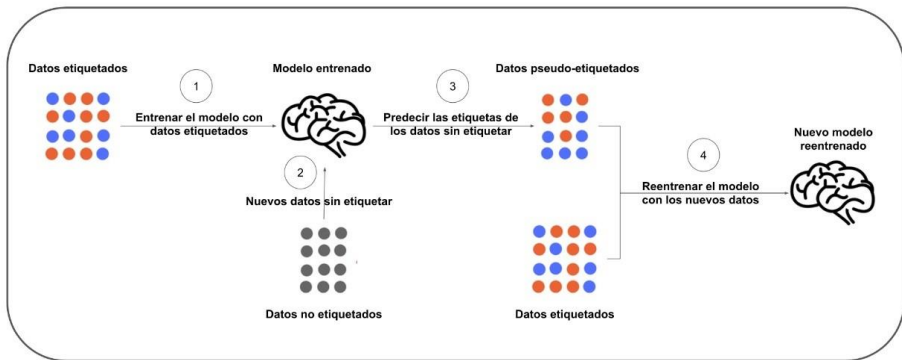
El problema de clasificación

- ▶ Cuando no hay, por ejemplo clustering, es no supervisado
= encuéntrame el mejor agrupamiento (optimiza una función)



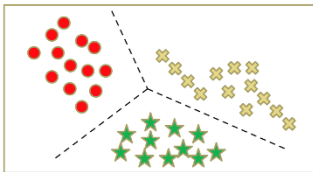
El problema de clasificación

- Cuando sólo hay algunas etiquetas, es semi supervisado

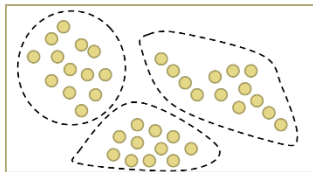


En resumen

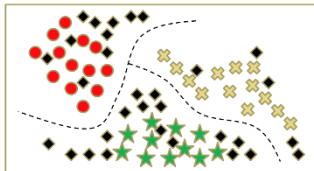
- Observamos como las fronteras cambian



Aprendizaje Supervisado

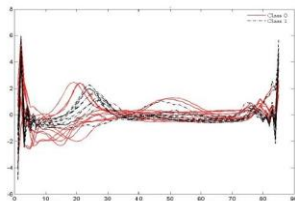
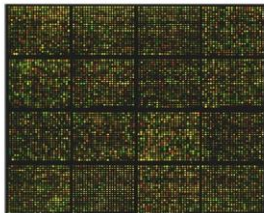
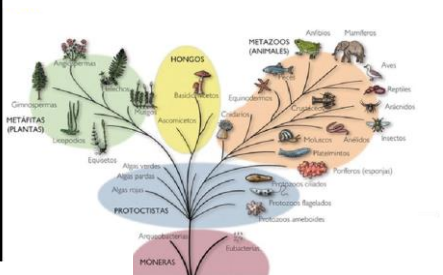
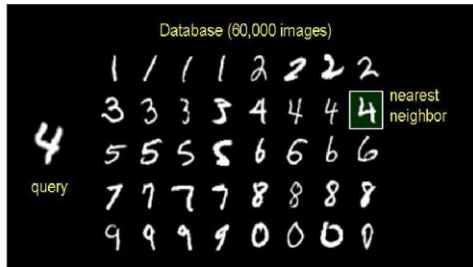


Aprendizaje No Supervisado



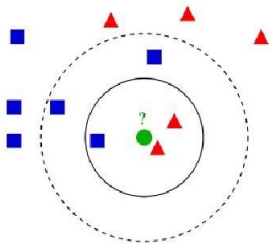
Aprendizaje Semi-Supervisado

Clasificación supervisada: Ejemplos



Definición del problema

Dada una nueva observación X cuya clase desconocemos, el problema de discriminación (Fisher 1936) consiste en decidir (a partir de la muestra) a qué población, P_0 o P_1 pertenece.



Metodología

- ❑ Obtención de datos: muestreo, representatividad...
- ❑ Análisis Exploratorio de Datos: representación de datos, elección de variables, filtrado...
- ❑ Elección del clasificador: (no-)paramétrico, (no-)lineal, global o local, mezclas...
- ❑ Entrenamiento: algoritmos, convergencia, tasas...
- ❑ Validación: generalización, validación cruzada, bootstrap...
- ❑ Evaluación: obtención y representación de resultados, márgenes, confianza...
- ❑ Decisión.

Análisis Exploratorio de Datos

Tareas

- ❑ Solucionar problemas de captura: descuadre, outliers...
- ❑ Completar los datos: missing values, interpolación...
- ❑ Filtrado: recuperar X a partir de $Z = X + s$.
- ❑ Representación: bases, variables categóricas, funciones, derivadas...
- ❑ Extracción de información: redundancia, estructuras latentes...
- ❑ Seleccionar el marco de trabajo y definir las herramientas.

Las claves en un modelo de predicción

1. ¿Cómo se representa el modelo?

- ☐ Un árbol, un conjunto de reglas, una distribución de probabilidad, una fórmula, ...
- ☐ ¿se puede interpretar?

2. ¿Cómo se genera/entrena el modelo?

- ☐ El algoritmo de aprendizaje: ¿cuántos datos necesita?
- ☐ ¿cuánto tarda?

3. ¿Cómo se realiza la predicción de una nueva instancia?

- ☐ Aplicando la formula, respondiendo las preguntas hasta encontrar la clase, ...

Métodos

Modelado
para
Clasificación

Árboles de Decisión

Reglas de Decisión

Modelado Bayesiano

Aprendizaje Vago

Redes de Neuronas

Máquinas Vector de Soporte

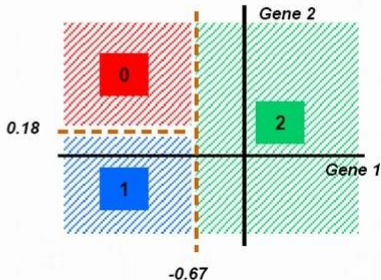
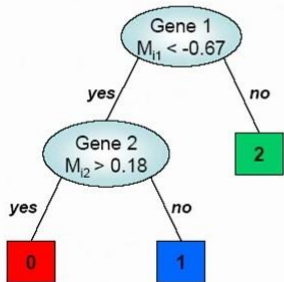
Análisis Discriminante

Métodos: árbol de Decisión

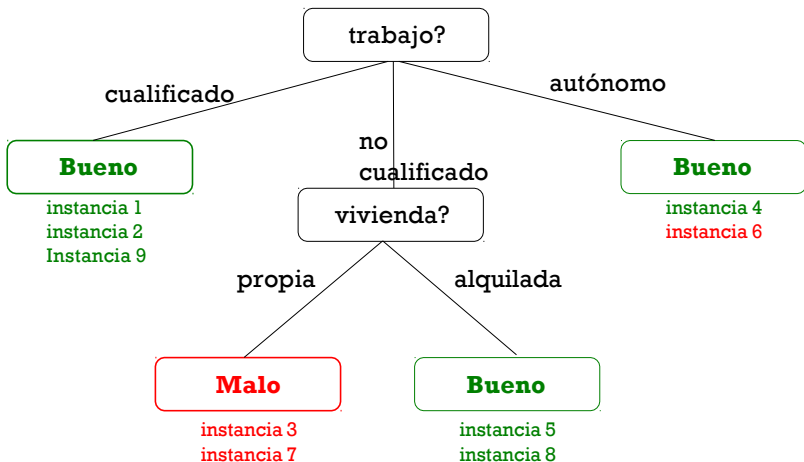
Classification and Regression Tree (CART)

- ▶ Particiona el espacio de la muestra en rectángulos y luego predice un modelo simple en cada uno de ellos
- ▶ Los árboles binarios van discriminando el espacio en dos submuestras (nodos) a partir de una anterior

Classification tree



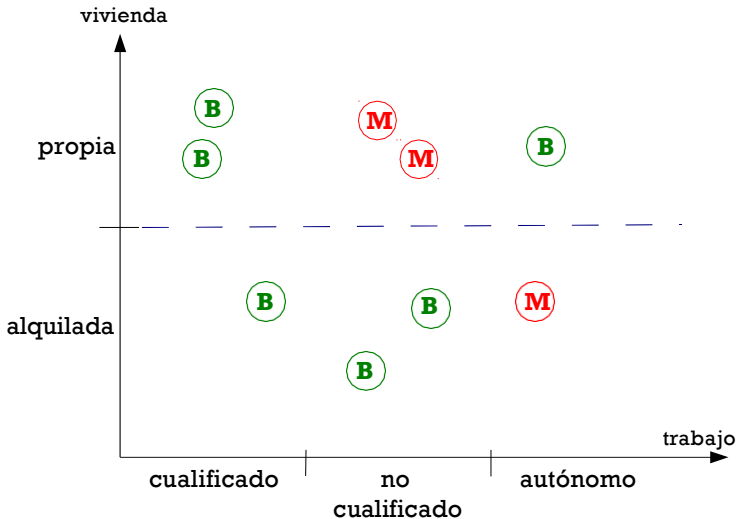
Métodos: árbol de Decisión



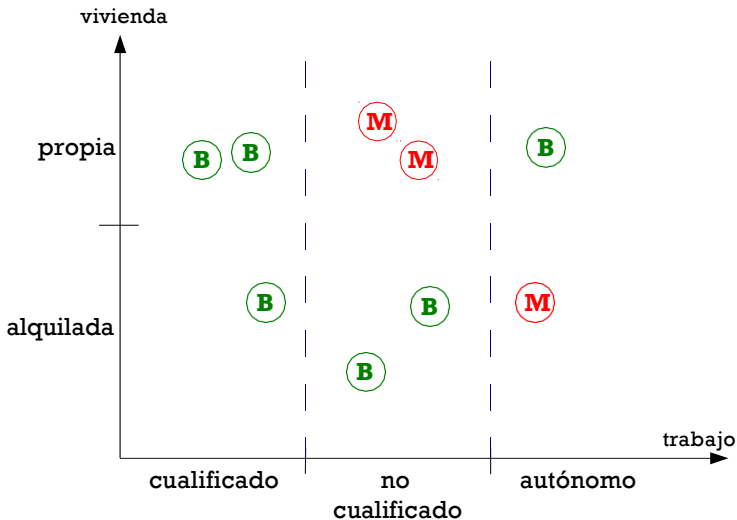
Árboles de decisión (conceptos)

- ❑ Diagrama que representa condiciones sucesivas sobre los atributos para clasificar una instancia
- ❑ Tipos de nodos
 - ❑ Nodos internos:
 - ❑ Preguntas condicionales sobre los atributos
 - ❑ Cada respuesta sigue a un arco o flecha
 - ❑ Separación completa de los ejemplos entre las posibles respuestas
 - ❑ Nodos hoja
 - ❑ Clase \rightarrow predicción
 - ❑ Confianza de predicción
 - ❑ Ejemplos de entrenamiento que cumplieron las condiciones hasta el nodo hoja
- ❑ Objetivos del modelado
 - ❑ Construir el árbol más sencillo que mejor separe las instancias por clase
 - ❑ El modelo final debe generalizar para clasificar bien futuras instancias

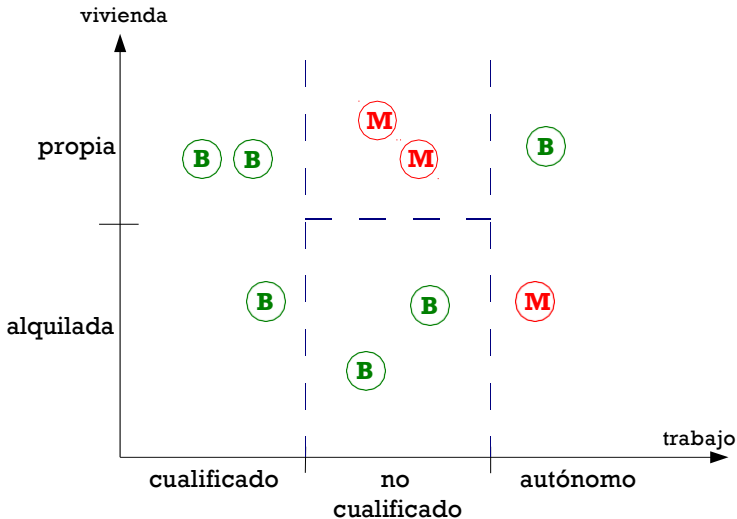
Estrategia árbol de decisión (ineficiente)



Estrategia árbol de decisión



Estrategia árbol de Decisión



Generación del árbol de decisión

- ❑ Construcción inductiva: Se añade en cada paso un atributo con todas sus posibles respuestas
- ❑ Se selecciona el atributo que **mejor separe** (ordene) los ejemplos de acuerdo a las clases
- ❑ Criterios de separación
 - ❑ Ganancia de información/ Ratio de ganancia de información
 - ❑ Entropía
 - ❑ Impureza gini
- ❑ El proceso se detiene cuando añadir un nuevo atributo no mejora el criterio de separación
- ❑ **Relevancia de atributos**
- ❑ La construcción de árboles de decisión realiza una selección de atributos implícita

Otros (hiper)parámetros

- ☐ Ganancia mínima
- ☐ Profundidad máxima del árbol
- ☐ Número mínimo de ejemplos por nodo hoja
- ☐ Poda del árbol

Reglas de Decisión

Ejemplo

SI vivienda = propia Y edad \leq 28

ENTONCES clase = **bueno**

SI trabajo = no cualificado Y cantidad \Rightarrow
50K

ENTONCES clase = **malo**

id	edad	estado civil	ahorros	formación	trabajo	casa	cantidad	clase
1	35	soltero	7,000	básica	<u>Cualif.</u>	propia	50K	bueno
2	23	casado	2,000	f.p	Cualif.	alquilada	70K	bueno
3	30	casado	1,000	básica	No-cual	propia	60K	malo
4	26	soltero	15,000	univ.	<u>autono</u>	propia	120K	bueno
5	50	divorciado	3,500	univ.	No-cual	alquilada	40K	bueno
6	43	soltero	N.S.	básica	<u>autono</u>	N.S	30K	malo
7	31	divorciado	28,000	máster o +	No-cual	propia	90K	malo
8	33	casado	N.S.	univ.	No-cual	alquilada	30K	bueno
9	40	soltero	11,000	máster o +	cual	propia	100K	bueno

Reglas de Decisión

Ejemplo

SI vivienda = propia Y duración \leq 24

ENTONCES clase = **bueno**

SI trabajo = no cualificado Y ahorros \leq 20K

ENTONCES clase = **malo**

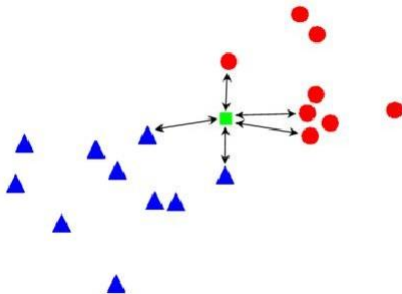
- ❑ Construcción inductiva en base a un criterio de separación o de cobertura
- ❑ No es obligatorio una separación completa de instancias (reglas con solapamiento)
- ❑ Todos los árboles de decisión pueden traducirse a reglas de decisión

Aprendizaje Basado en Instancias

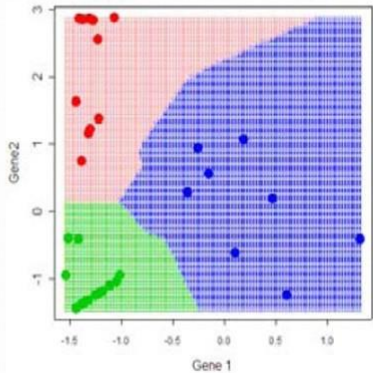
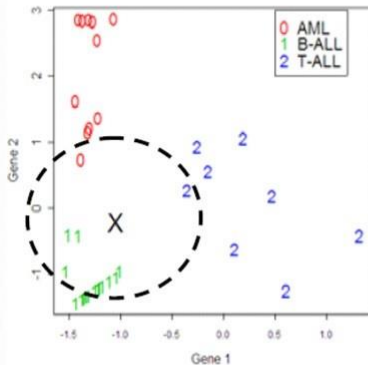
- ❑ Otra técnica más para generar clasificadores
- ❑ Entrenamiento
 - ❑ No hay generación de un modelo!
 - ❑ Sólo se almacenan todas las instancias
- ❑ Evaluación o Uso (paso de generalización):
 - ❑ Se extrae un conjunto de instancias que sean **similares** a las que se quiera clasificar
 - ❑ Se toma una decisión en función de las instancias seleccionadas
- ❑ Algoritmo tipo: k-vecinos más cercanos (k-NN)

k-NN

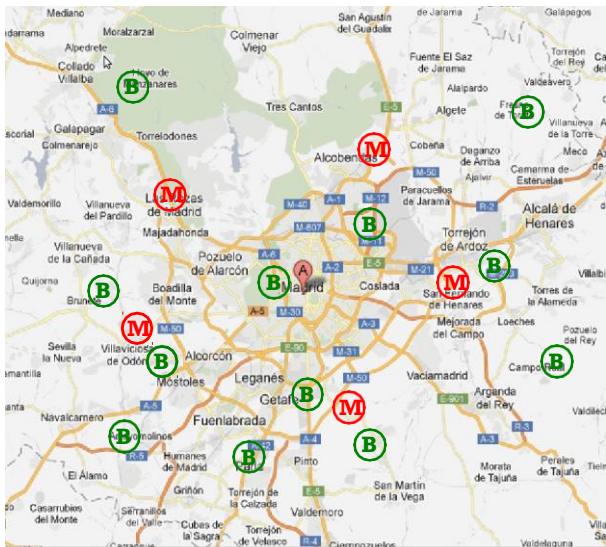
La idea detrás de k-NN es muy sencilla. Dada una observación x se buscan las k observaciones más cercanas a x (k-vecinos) entre los X_i del conjunto de entrenamiento. Se asigna x a la clase mayoritaria entre los k-vecinos.



Nearest neighbor rule



k-NN



k-NN

- ❑ El parámetro k define el número de vecinos que se eligen para tomar la decisión
- ❑ Medida de similitud
 - ❑ Cada instancia está en un espacio de dimensión n
 - ❑ Los vecinos más cercanos se determinan por la distancia euclídea

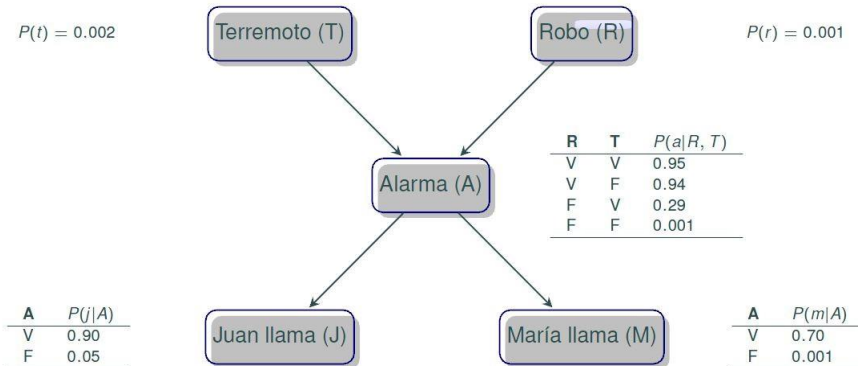
$$d(x_i, x_j) = \sqrt{\sum_{a=1}^n (x_i[a] - x_j[a])^2}$$

Aprendizaje Bayesiano

- ❑ La teoría de decisión bayesiana se basa en dos supuestos
 - ❑ El problema de decisión se puede describir en términos probabilísticos
 - ❑ Todas las probabilidades del problema son conocidas o al menos pueden estimarse
- ❑ Las decisiones se toman en función de los datos observados

Red Bayesiana: ejemplo alarma

- ▶ Cada nodo raíz (sin padres) tiene una Distribución de Probabilidad a Priori
- ▶ Cada nodo interno tiene una Tabla de Probabilidad Condicional (CPT) que cuantifica el efecto que sus padres tienen sobre él



¿Cómo generar una red bayesiana?

☐ Necesitamos

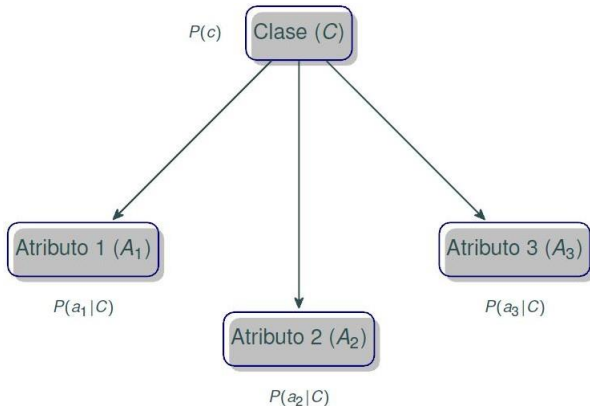
- ☐ Las variables aleatorias y sus dominios
- ☐ La estructura de la red: independencia condicional
- ☐ Las tablas de probabilidad condicionada (CPTs)

☐ ¿De dónde lo obtenemos?

- ☐ Hay que diseñar la red
- ☐ Podemos preguntar a un experto
- ☐ Las CPTs se pueden aprender de los datos
- ☐ La estructura también se puede aprender de los datos, pero es más complicado

Clasificador Naïve Bayes

es un caso especial de red bayesiana con la siguiente estructura:



Notación

- Conjunto de clases

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

- Conjunto de atributos

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

- Instancia: valores de los atributos

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

- Probabilidades condicionales:

- $P(c_j | X)$ Probabilidad de observar la clase c_j dada la instancia X
- $P(X | c_j)$ Probabilidad de observar la instancia X dada la clase c_j

Clasificadores Bayesianos

Observaciones

Inst.	Trabajo	Ahorros	Vivienda	Duración	Clase
1	cualificado	> 20K	propia	24	bueno
2	cualificado	10 - 20K	alquilada	40	bueno
3	no cualificado	10 - 20K	propia	12	malo
4	autónomo	no conocidos	propia	36	bueno
5	no cualificado	0 - 10K	alquilada	12	bueno
6	autónomo	10 - 20K	alquilada	12	malo
7	no cualificado	0 - 10K	propia	30	malo
8	no cualificado	> 20K	alquilada	12	bueno

Problema de decisión

- $P(\text{clase} = \text{bueno} \mid \text{trabajo}=\text{cualificado}, \text{ahorros}=10 - 20\text{K}, \text{vivienda}=\text{propia}, \text{duración}=40)$
- $P(\text{clase} = \text{malo} \mid \text{trabajo}=\text{cualificado}, \text{ahorros}=10 - 20\text{K}, \text{vivienda}=\text{propia}, \text{duración}=40)$
- ¿bueno o malo?

Clasificadores Bayesianos

Teorema de Bayes

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

- ❑ La probabilidad a priori de una instancia x_i es independiente del valor de la clase, por lo que generalmente $P(x_i)$ no se calcula
- ❑ La idea del clasificador consiste en elegir la clase más probable según la probabilidad a posteriori $P(c_j | x_i)$

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

Clasificador Naïve Bayes

- $P(X_i \mid c_j)P(c_j) = P(x_1, x_2, \dots, x_n \mid c_j)P(c_j)$
- Se parte del supuesto que los valores de los atributos son independientes

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

Naïve Bayes - Ejemplo 2

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 - Ejemplo 2

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

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

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

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

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:

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

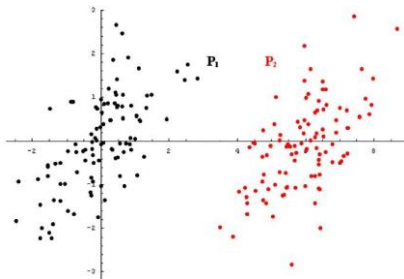
$$P(\text{no}) P(\text{Sunny}|\text{no}) P(\text{Cool}|\text{no}) P(\text{High}|\text{no}) P(\text{String}|\text{no}) = \mathbf{0.0206}$$

⇒ Answer: **PlayTennis = no**

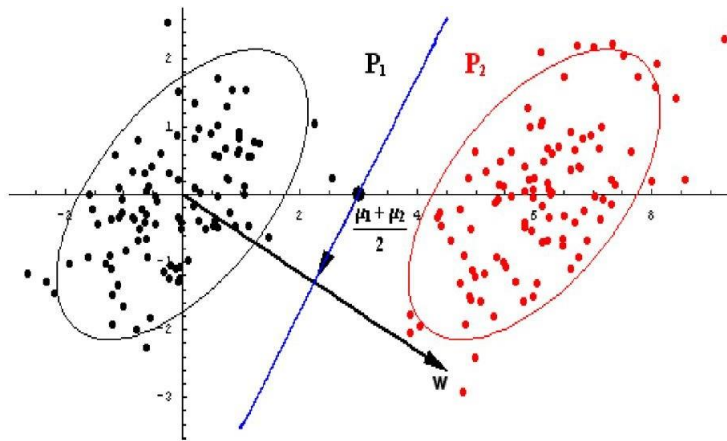
⇒ **Con 79.5% de certeza**

Métodos: Discriminante lineal de Fisher

La idea de Fisher es encontrar el “mejor clasificador lineal”, esto es, encontrar la ecuación del hiperplano que separe mejor (en términos de error de clasificación) las dos poblaciones.



Métodos: Discriminante lineal de Fisher



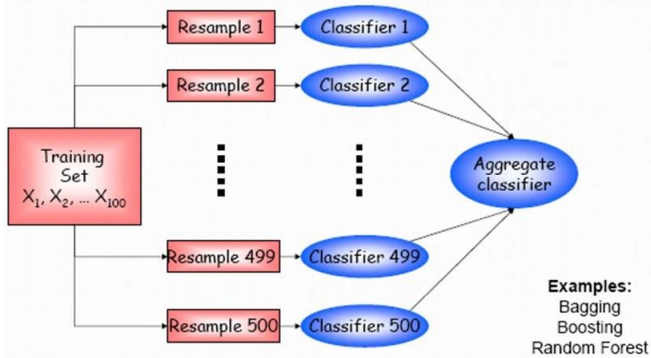
Mezclas de expertos

Una mezcla de clasificadores es un conjunto $g_1(x), \dots, g_k(x)$ cuya estimación de la clase Y es una combinación de las estimaciones de cada uno de los clasificadores. Es necesario tener en cuenta una serie de requisitos como la independencia.

- ☐ Jerárquico.
- ☐ Bagging: voto sin peso.
- ☐ Boosting: voto ponderado.
- ☐ Stacking

Mezclas de expertos

Another component in classification rule:
aggregating classifiers



compensación sesgo-varianza

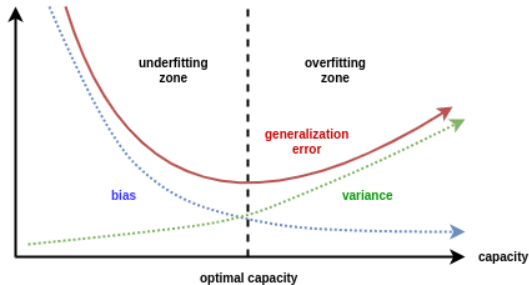
❑ Los modelos débiles pueden tener

❑ alto sesgo/bias -> desajuste/underfitting

❑ Sesgo = la diferencia entre el valor esperado de nuestro modelo y el valor real del parámetro que se estima

❑ alta varianza para ser robustos -> sobreajuste /overfitting

❑ Varianza = cuánto esperamos que **varíe** el rendimiento de nuestro modelo en función del remuestreo independiente de datos a diferencia de la distribución generadora de datos. (Recordamos que la raíz cuadrada de la varianza = error estándar)



Etapas de un proyecto Data Science

1. Definición del objetivo

Tarea a realizar: clasificar, predecir, agrupar, ...

2. Recolección y comprensión de los datos

Características generales, visualización, verificación de la calidad

3. Pre-procesado

Integración de fuentes, generación de atributos, selección de atributos, transformaciones,...

4. Modelado

Técnica de aprendizaje automático para generar/entrenar el modelo

5. Evaluación y análisis de resultados

Estimación del rendimiento/error, elección del mejor modelo

6. Uso del modelo

Desarrollo o integración en un sistema informático/agente inteligente

Evaluación del Modelo Aprendido

Preguntas

- ❑ ¿Cómo de bueno es el modelo que hemos generado para predicciones de futuras instancias?
- ❑ ¿Serán nuestras predicciones mejores que una simple clasificación al azar o por mayoría?
- ❑ Clasificador por Defecto
 - ❑ Predice siempre con la **moda** de los ejemplos de entrenamiento (clase mayoritaria)
 - ❑ Trampa de la precisión en conjuntos no balanceados

Diseño de la Evaluación:

División de los Datos

- ❑ Conjunto de **entrenamiento**
 - ❑ - Se utiliza para generar el modelo
- ❑ Conjunto de **prueba**
 - ❑ Se conoce el valor o clase a predecir
 - ❑ Este valor se compara con la predicción del modelo para evaluar la calidad del mismo
- ❑ Conjunto de **producción**
 - ❑ No se conoce el valor o clase de las instancias
 - ❑ Se utiliza en la fase de uso con el modelo final generado

Dividir el conjunto de datos: para que el modelo no lo aprenda de memoria

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

Por defecto 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

❑ **Problema:** datasets desbalanceados

❑ No hay la misma representación de cada clase

Solución al desbalanceo 1: muestreo estratificado

☐ Muestreo teniendo en cuenta la distribución en el conjunto de datos

- 100 muestras, 80 clase 1 y 20 clase 2
- Training set: 75 muestras, 60 clase 1 y 15 clase 2
- Test set: 25 muestras, 20 clase 1 y 5 clase 2

Representando entrenamiento y test con misma distribución del original. es decir, 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()
```

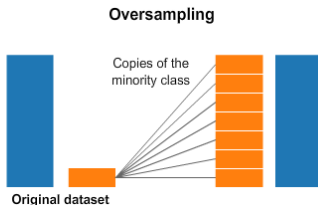
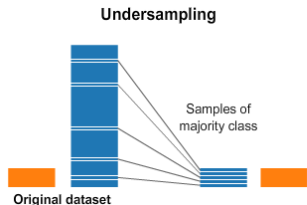
```
y_test["labels"].value_counts()
```

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

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


Solución al desbalanceo 2: Manually balance

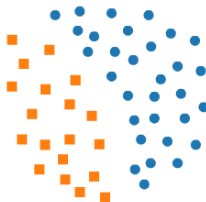
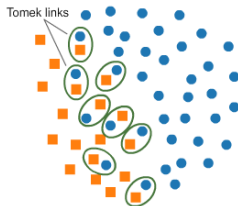
- ❑ Si la muestra es suficientemente grande: Infra-muestreo
 - ❑ Mantener todas las instancias de clase minoritaria ($N = A$)
 - ❑ Seleccionar instancias al azar para la clase mayoritaria (keep length = A)
 - ❑ Se puede hacer en cada k-fold
 - ❑ Puede causar pérdida de información
 - ❑ **Cluster Centroids**: puede agrupar los registros de la clase mayoritaria y realizar el submuestreo eliminando registros de cada grupo, buscando así preservar la información



- ❑ Sobre-muestreo
 - ❑ registros aleatorios duplicados de la clase minoritaria
 - ❑ Puede causar sobre ajuste
 - ❑ **SMOTE** (Synthetic Minority Oversampling TEchnique) introducir pequeñas variaciones en esas copias, creando muestras sintéticas más diversas
 - ❑ estas técnicas tienen sus debilidades(no hay free lunch).

Solución al desbalanceo 3: Avanzado

□Infra-muestreo: links Tomek



```
from imblearn.under_sampling import TomekLinks

tl = TomekLinks(return_indices=True, ratio='majority')
X_tl, y_tl, id_tl = tl.fit_sample(X, y)

print('Removed indexes:', id_tl)

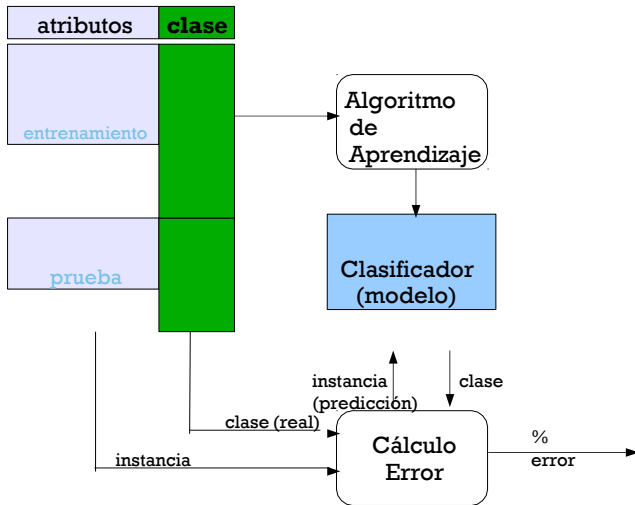
plot_2d_space(X_tl, y_tl, 'Tomek links under-sampling')
```

```
def plot_2d_space(X, y,
label='Classes'):
    colors = ['#1F77B4', '#FF7F0E']
    markers = ['o', 's']
    for l, c, m in zip(np.unique(y),
colors, markers):
        plt.scatter(
            X[y==l, 0],
            X[y==l, 1],
            c=c, label=l, marker=m)
    plt.title(label)
    plt.legend(loc='upper right')
    plt.show()
```

```
Removed indexes: [ 0  1  2  4  5  6  7  8  9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
73 74 75 76
77 78 79 80 81 82 83 84 85 86 87 88 90 91 92 93 94 95 97 98 99]
```

Source: [www.kaggle.com. Resampling strategies for imbalanced datasets. 2017.](https://www.kaggle.com/rafaa/resampling-strategies-for-imbalanced-datasets)
<https://www.kaggle.com/rafaa/resampling-strategies-for-imbalanced-datasets>

Evaluación de los Modelos



Matriz de Confusión

		clase real	
		positiva	negativa
clase predicción	positiva	verdadero positivo (TP)	falso positivo (FP)
	negativa	falso negativo (FN)	verdadero negativo (TN)

Métricas (1)

- Precisión global (*accuracy*): Aciertos totales

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

Métricas (2)

- Precisión de clase (*precision*):

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

¿Cuántos de los que predigo son correctos?

- Sensibilidad (*recall*): Ratio de verdaderos positivos

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

¿Cuántos de los que debería predecir estoy prediciendo?

Métricas (3)

- Especificidad (specificity): Ratio de verdaderos negativos

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

¿Cuántos de los que excluyo son correctos?

- Ratio de falsos positivos (1 - specificity)

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

¿Cuántos de los que debería excluir estoy prediciéndolos erróneamente?

Evaluación de los Modelos

Sobre-adeacuación (Overfitting)

Los modelos se refinan tanto que describen bien las instancias de entrenamiento pero obtienen un error alto en ejemplos externos

Causas

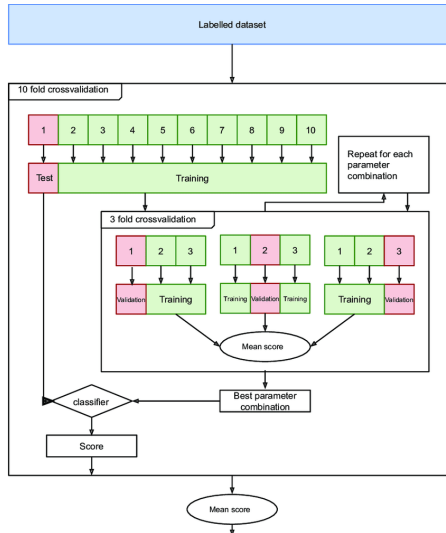
- ❑ pocos ejemplos
- ❑ las instancias de entrenamiento no son una muestra representativa
- ❑ ruido en las instancias de entrenamiento
- ❑ error en los datos

Validación Cruzada

Validación Cruzada k -veces

- ❑ Se divide el conjunto completo de ejemplos (E) en k partes iguales, E_i
- ❑ Para cada vez se ejecuta:
 - ❑ Se crea un modelo con el conjunto $E - E_i$
 - ❑ Se calcula el error con E_i
- ❑ Se estima el error real haciendo la media de los errores

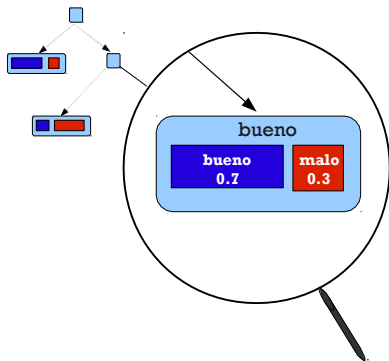
Grid Search: Hyperparameter tuning



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

Confianza de Predicción

- ❑ Proporción de ejemplos en las instancias de entrenamiento
- ❑ Estimación del acierto en predicción
- ❑ Umbral de confianza
 - ❑ Por defecto 0.5
 - ❑ Predicción de la clase mayoritaria



Selección del Umbral

Razones

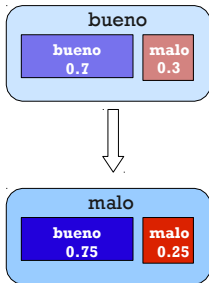
- ❑ Buscar una mejor proporción entre precisión y sensibilidad
- ❑ Los errores tienen diferente coste

Si confianza[**bueno**] > 0.75 entonces

clase = **bueno**

de lo contrario

clase = **malo**



Clasificación Multiclase

		clase real			
		A+	A	B	C
predicción	A+				
	A				
	B				
	C				

- ❑ Con N clases, la matriz de confusión siempre es $N \times N$.
- ❑ Con escalas de ordenación los costes de error no son iguales

ESQUEMA

- ❑ Introducción
- ❑ Técnicas de Clasificación
- ❑ **Técnicas de Regresión**

Regresión

- En las instancias de entrenamiento $\{x_1, x_2, \dots, x_n, y\} = \{X, y\}$ nos interesa predecir el valor de y , que es un **valor continuo**.
- La tarea de regresión consiste en encontrar una función que aproxime los valores de la variable a predecir.

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

Regresión

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

- W es el vector de parámetros desconocidos que también determina el modelo
- la forma funcional de f es en principio desconocida.
 - Conociendo la relación de los atributos puede asumirse que el modelo sigue alguna forma de ecuación lineal, polinómica, exponencial
 - Utilizar alguna técnica que permita aproximar f sin que esta siga alguna forma de ecuación

Regresión

- ❑ Encontrar la mejor función f

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

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

- ❑ Será la función que minimice el error en los datos de entrenamiento $[\{X_1, y_1\}, \{X_2, y_2\}, \dots, \{X_m, y_m\}]$

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

Regresión Clásica

- Si especificamos que f sigue un tipo de ecuación, el objetivo consiste en encontrar el vector de parámetros $W = \{w_0, w_1, \dots, w_n\}$
- Regresión lineal

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

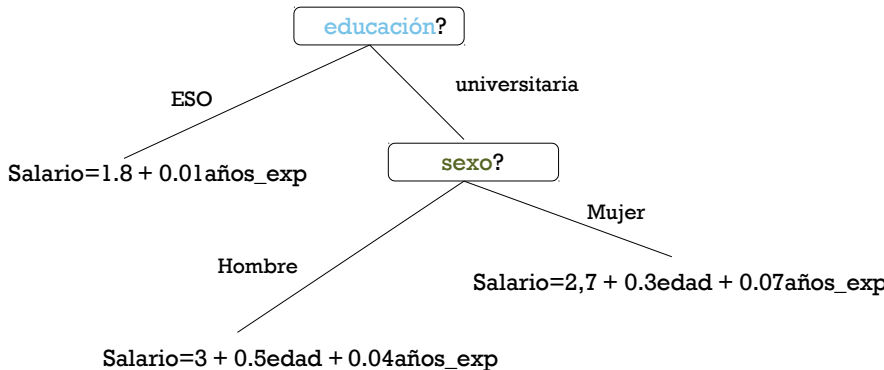
- Regresión exponencial

$$y = w_0 e^{w_1 x_1} + \varepsilon$$

Árboles de Regresión

- ❑ Árboles de decisión similares a los generados para clasificación, pero con modelos de regresión lineal para los ejemplos de cada hoja.
- ❑ Construcción inductiva: Elegir cada vez el atributo que minimice la variación de los valores de y en cada subconjunto de instancias.
- ❑ Criterio de parada: Cuando haya pocos ejemplos en el nodo actual o cuando no haya mucha variación en los valores de y .

árboles de Regresión



Redes de Neuronas

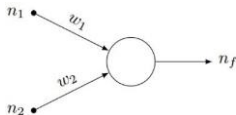
- ❑ Surgieron como un intento de reproducir un modelo biológico de la inteligencia
- ❑ Hoy están consideradas como una técnica más para generar modelos de clasificación y predicción
- ❑ Son capaces de aproximar funciones no lineales
- ❑ **Dados unos parámetros hay una forma de combinarlos para predecir un cierto resultado.**

Redes de Neuronas: Ejemplo

- ❑ Sabiendo los píxeles de una imagen habrá una forma de saber qué número hay escrito,
- ❑ O, conociendo la carga de servidores de un Centro de Procesamiento de Datos (CPD), su temperatura y demás existirá una manera de saber cuanto van a consumir
- ❑ El problema, está en que no sabemos cómo combinarlos.

Redes de Neuronas: Ejemplo

Sois alumnos de una clase en la que el profesor no ha dicho exactamente cómo va a poner las notas. Para empezar, supongamos que sólo habéis hecho dos exámenes y tenéis la nota de cada uno de ellos y la final.



¿Cómo usamos una red neuronal para saber cuanto vale cada examen? Aquí nos bastará con la unidad fundamental de la red neuronal: el perceptrón. Un perceptrón es un elemento que tiene varias entradas con un cierto peso cada una. Si la suma de esas entradas por cada peso es mayor que un determinado número, la salida del perceptrón es un uno. Si es menor, la salida es un cero.

Redes de Neuronas: Ejemplo

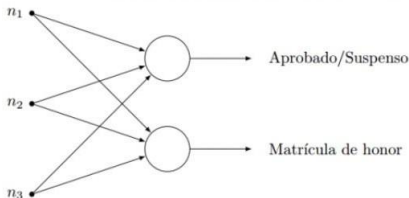
En nuestro ejemplo, las entradas serian las dos notas de los exámenes. Si la salida es uno (esto es, la suma de las notas por su peso correspondiente es mayor que cinco), es un aprobado. Si es cero, suspenso. Los pesos son lo que tenemos que encontrar con el entrenamiento. En este caso, nuestro entrenamiento consistirá en empezar con dos pesos aleatorios (por ejemplo, 0.5 y 0.5, el mismo peso a cada examen) y ver qué resultado da la red neuronal para cada alumno. Si falla en algún caso, iremos ajustando los pesos poco a poco hasta que esté todo bien ajustado.

Redes de Neuronas: Ejemplo

Por ejemplo, si un alumno con muy buena nota en el segundo examen ha suspendido el curso, bajaremos el peso del segundo examen porque claramente no influye demasiado.

Poco a poco acabaremos encontrando los pesos que se ajusten a las notas que puso el profesor. La idea del ajuste o retroalimentación es ir adaptando la red a la información “oculta” que tienen los datos que le pasamos para que aprenda.

Redes de Neuronas: Ejemplo



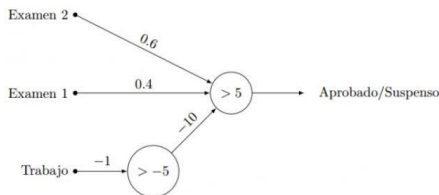
Quizás queramos complicarlo más, poniendo más exámenes (más nodos de entrada) o queriendo sacar más resultados, como pueda ser un perceptrón cuya salida sea uno si el alumno tiene matrícula de honor.

Multiplicando la potencia: redes multicapa

Resulta que se da una situación curiosa. Hay dos alumnos que tienen la misma nota en los exámenes, dos dieces, pero uno tiene un 7 en el trabajo y otro un 4. El del 7 ha aprobado el curso, pero el del 4 no. Hay un alumno que tiene un 10 en el trabajo y 4.99 en los dos exámenes y que está suspenso. Podemos intentar entrenar una red neuronal como la de antes pero en esta situación no va a funcionar bien. Y es que parece que la nota del trabajo no influye salvo que lo suspendas, en cuyo caso estás suspenso directamente. Es un filtro, un uno o un cero que tenemos que sacar en la red neuronal antes de poder dar el resultado de aprobado o suspendido en el curso...

Multiplicando la potencia: redes multicapa

Necesitamos máscapas. Necesitamos un perceptrón intermedio que nos diga si el trabajo está aprobado o no, y contar eso en el perceptrón de salida:



El primer perceptrón mira si la nota del trabajo multiplicada por menos uno es mayor que menos cinco (o, lo que es lo mismo, si la nota es menor que cinco). Si lo es, entonces su salida es uno. Al multiplicarla por menos diez en la entrada del segundo perceptrón, forzará siempre un suspenso. Si el trabajo está aprobado, la salida del primer perceptrón será 0 y no afectará a la media de los exámenes.

¿para qué nos sirven las capas?

Para añadir información que no estaba antes. Cogemos los datos de entrada, los exploramos y sacamos las características que mejor nos ayuden a entender que está pasando.

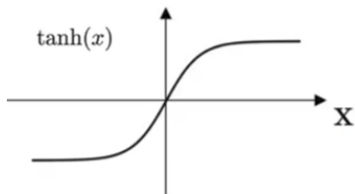
- ❑ Lo bueno viene cuando, durante el proceso de aprendizaje, cada capa “aprende” a encontrar y detectar las características que mejor ayudan a clasificar los datos.
- ❑ En nuestro ejemplo, durante el ajuste la primera capa aprendería que los alumnos con el trabajo suspenso suspenden el curso.
- ❑ Si cogiésemos una red para detectar números escritos a mano, puede que las capas ocultas aprendiesen a detectar trazos rectos o curvados que sirvan para decidir si estamos ante un uno o un ocho, por ejemplo

Más allá de perceptrones: sigmoides, redes profundas y redes convolucionales

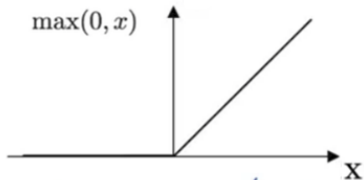
- ❑ Hasta ahora nos hemos centrado en simplificaciones para entender bien los conceptos de redes neuronales.
- ❑ En la realidad, las cosas se complican bastante. Por ejemplo, se dejan de usar perceptrones para usar otras “neuronas” con un comportamiento más suave, usando funciones como la sigmoide.
- ❑ La idea es que pequeños cambios en los pesos provoquen pequeños cambios en la salida de la red, para así poder hacer más “fácil” el aprendizaje.

Funciones de Activación

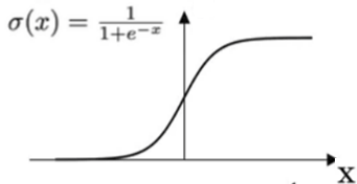
Hyper Tangent Function



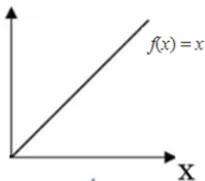
ReLU Function









Sigmoid Function



Identity Function



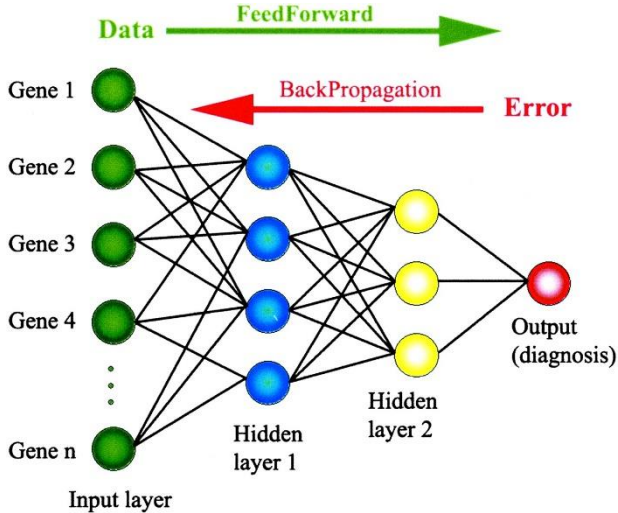
Funciones de Activación

Name	Graphic	Function
Linear		$\varphi(\varphi) = \varphi \cdot \varphi + \varphi$
Binary step		$\begin{aligned} &\text{if } \varphi \geq 0, && \text{then } \varphi(\varphi) = 1, \\ &\text{if } \varphi < 0, && \text{then } \varphi(\varphi) = 0, \end{aligned}$
Piecewise linear		$\begin{aligned} &\text{if } \varphi \geq \varphi_{\text{upper}}, && \text{then } \varphi(\varphi) = 1, \\ &\text{if } \varphi_{\text{upper}} > \varphi > \varphi_{\text{lower}}, && \text{then } \varphi(\varphi) = \varphi \cdot \varphi + \varphi_{\text{lower}}, \\ &\text{if } \varphi \leq \varphi_{\text{lower}}, && \text{then } \varphi(\varphi) = 0, \end{aligned}$
Sigmoid		$\varphi(\varphi) = \frac{1}{1 + \varphi^{-\varphi}}, \quad \text{interval } (0,1)$
Gaussian		$\varphi(\varphi) = \varphi^{-\varphi^2}, \quad \text{interval } (0,1)$
Hyperbolic tangent		$\begin{aligned} &\varphi(\varphi) \\ &= \frac{2}{1 + \varphi^{-2\varphi}} - 1, \quad \text{interval } [-1,1] \end{aligned}$

Redes de Neuronas Artificiales: Resumen

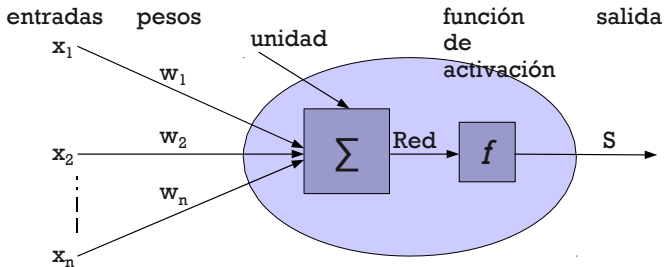
- ❑ Conjunto de neuronas artificiales conectadas entre sí mediante un conjunto de conexiones
- ❑ Las conexiones tienen números reales asociados llamados **pesos**
- ❑ Generalmente las neuronas se distribuyen en **capas** de distintos niveles
- ❑ Cada neurona recibe **señales de entrada** provenientes del exterior o de otras neuronas
- ❑ La **señal de salida** se calcula como una función de las entradas

Redes de Neuronas



Fuente: 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.

Neurona Artificial

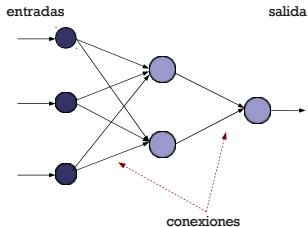


$$Red = u + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

$$S = f(Red)$$

Construcción del Modelo

- ❑ Estructura de la Red: Número de neuronas, número de capas, conexiones, etc.
- ❑ Aprendizaje de la Red: Modificación iterativa de los pesos para que la salida de la red sea la correcta



Evaluación en Regresión

- ❑ Un modelo debería ser mejor que el modelo por defecto, que en regresión devuelve la media del atributo objetivo.
- ❑ El cálculo del error sobre los datos de entrenamiento nos indica como se ajusta el modelo a los datos
- ❑ Validación cruzada
 - ❑ Mismo esquema que la evaluación en clasificación
 - ❑ El cálculo del error estima como el modelo generaliza ante futuros datos

Métricas para Regresión

- ❑ Error Cuadrático Medio (**M**ean **S**quared **E**rror):

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

- ❑ **R**oot **M**ean **S**quared **E**rror

$$RMSE = \sqrt{MSE}$$

- ❑ **M**ean **A**bsolute **E**rror

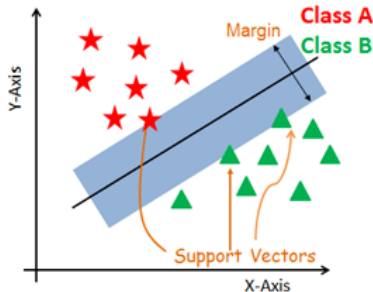
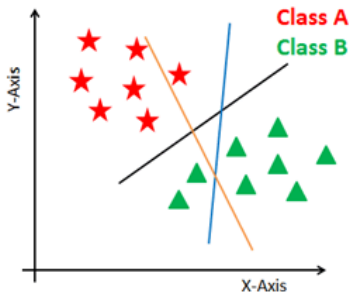
$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \bar{y}_i|$$

Apéndice

SVM

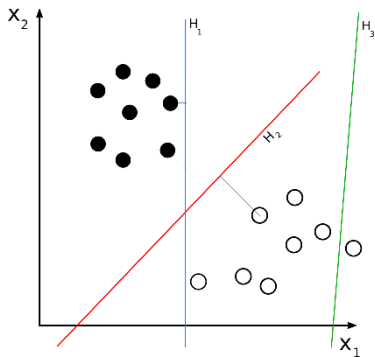
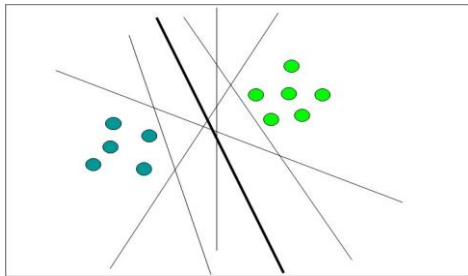
- Los modelos de máquinas de vectores soporte (svm) son una representación de los ejemplos como puntos en el espacio, mapeados de modo que los ejemplos de las categorías separadas estén divididos por una brecha clara que sea lo más amplia posible. Luego, los nuevos ejemplos se mapean en ese mismo espacio y se predice que pertenecen a una categoría según el lado de la brecha en la que caen.

Los parámetros se ajustan para maximizar el margen resolviendo el problema.



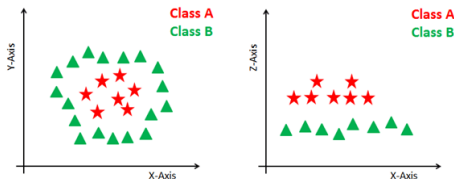
SVM

buscan el hiperplano que corta la muestra que contenga el margen mayor



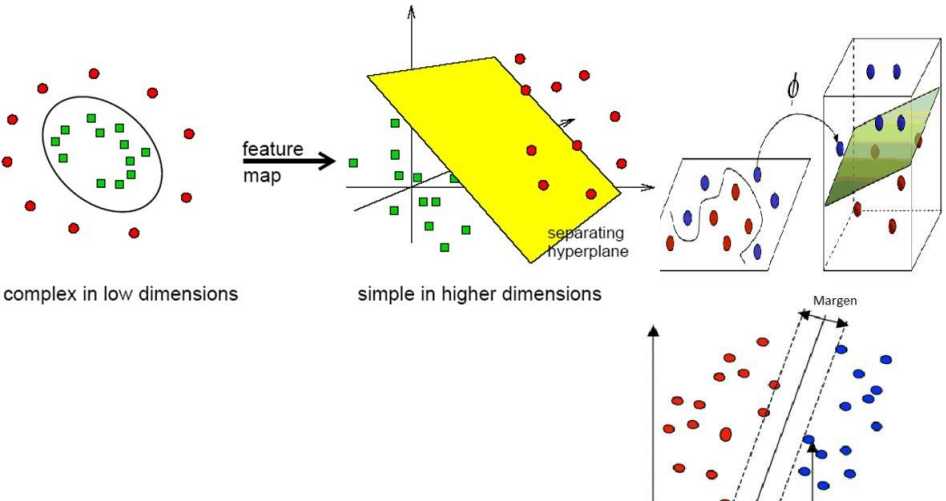
SVM

- ❑ La manera más simple de realizar la separación es mediante una línea recta, un plano recto o un hiperplano N-dimensional.
- ❑ Desafortunadamente los universos a estudiar no se suelen presentar en casos idílicos de dos dimensiones como en el ejemplo anterior, sino que un algoritmo SVM debe tratar con a) más de dos variables predictoras, b) curvas no lineales de separación, c) casos donde los conjuntos de datos no pueden ser completamente separados, d) clasificaciones en más de dos categorías.
- ❑ Debido a las limitaciones computacionales de las máquinas de aprendizaje lineal, éstas no pueden ser utilizadas en la mayoría de las aplicaciones del mundo real. La representación por medio de funciones Kernel ofrece una solución a este problema, proyectando la información a un espacio de características de mayor dimensión el cual aumenta la capacidad computacional de la máquinas de aprendizaje lineal. Es decir, mapearemos el espacio de entradas X a un nuevo espacio de características de mayor dimensionalidad (Hilbert).



SVM

Separation may be easier in higher dimensions



Bootstrapping

- ❑ Muestras de tamaño B del conjunto de datos original **con reemplazo**



- ❑ Buenas propiedades:
 - ❑ extraídos directamente de la distribución de datos e independientemente unos de otros
- ❑ Condiciones -> N tiene que ser lo suficientemente grande :
 - ❑ Para capturar la mayor parte de la complejidad de la distribución subyacente (**representatividad**)
 - ❑ Comparado con B para que las muestras no estén demasiado correlacionadas (**independencia**)

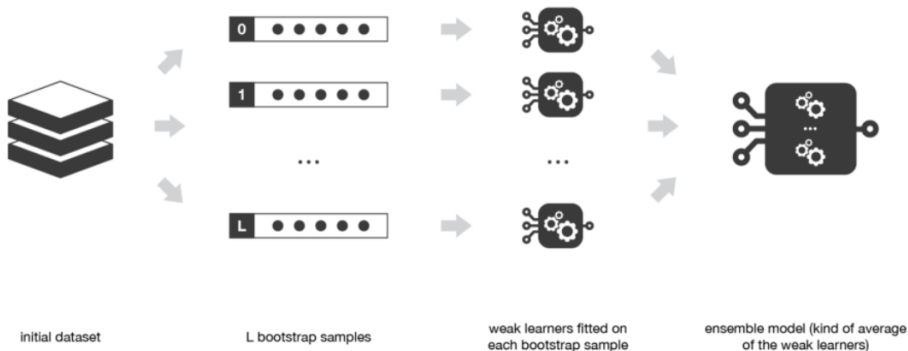
Bagging

- ❑ Idea: usar muestras de bootstrapping y ajustar varios modelos "independientes" y "promediar" sus predicciones para obtener un modelo con una varianza más baja -> más robusto -> menos sobreajuste

- ❑ Agregaciones:

- ❑ Simple/weighted average (regresión)
- ❑ Majority/weighted vote (clasificación)

- ❑ Funciona con paralelización



Random forests

❑ Idea: usar **bagging** en árboles profundos -> menor varianza -> más robusto -> menos sobreajuste

❑ Árboles profundos = muchas profundidades, sesgo bajo pero varianza alta

❑ Árboles poco profundos = pocas profundidades, menor variación pero mayor sesgo

❑ Diferencias con bagging

❑ Muestreo no solo en instancias sino también en features

❑ Ventajas

❑ Los árboles no miran exactamente la misma información para tomar sus decisiones -> reduce la correlación

❑ Hace que el proceso de toma de decisiones sea más robusto a los datos faltantes.



initial dataset

bootstrap
samples + selected
features

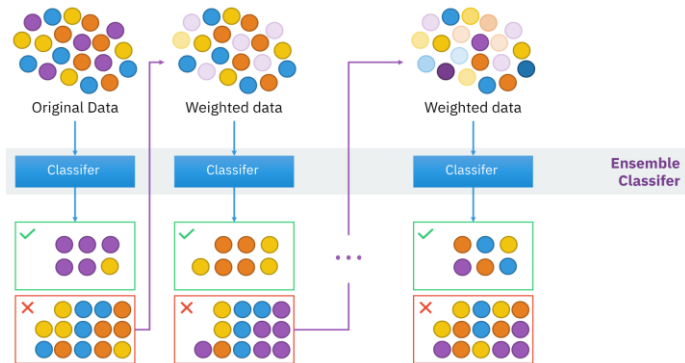
deep trees fitted on each
bootstrap sample and considering
only selected features

random forest
(kind of average of the trees)

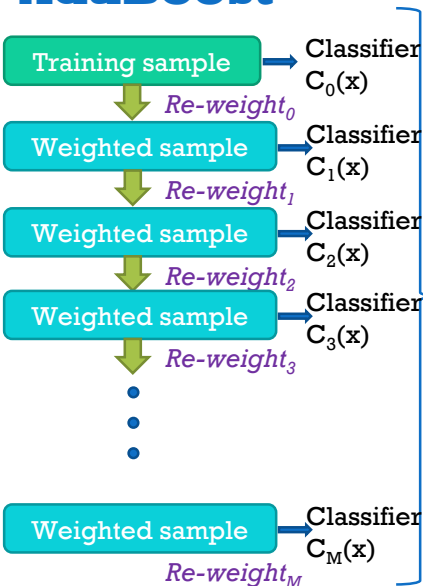
Fuente: Joseph Rocca. Ensemble methods: bagging, boosting and stacking, 2019

Boosting

- ❑ Centrado principalmente en reducir el sesgo (aunque también puede reducir la varianza)
- ❑ Secuencial
- ❑ Idea: ajustando modelos secuencialmente de una manera muy adaptativa
 - ❑ Dar más importancia a las observaciones del conjunto de datos que fueron mal predichas por los modelos anteriores en la secuencia.
 - ❑ Cada nuevo modelo concentra sus esfuerzos en las observaciones más difíciles de predecir hasta ahora



Boosting (ii): Adaptive boosting = AdaBoost



- AdaBoost vuelve a ponderar las instancias clasificadas erróneamente por el clasificador anterior :

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

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

- AdaBoost pondera los clasificadores utilizando la tasa de error del clasificador individual de acuerdo con :

$$y(x) = \sum_i^{M_{\text{classifier}}} \log(re - weight_i) C_i(x)$$

Boosting (iii): Gradient boosting

El aumento de gradiente actualiza los valores de las observaciones en cada iteración. Los modelos débiles están entrenados para ajustarse a los pseudo-residuales que indican la dirección para encontrar la predicción correcta del modelo de conjunto actual y reducir el error.

El término pseudo-residual se basa en la regresión lineal donde la diferencia entre los valores observados y los predichos son residuos



train a weak model
and aggregate it to
the ensemble model



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



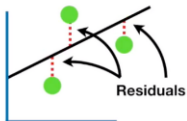
dataset values



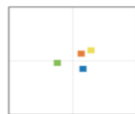
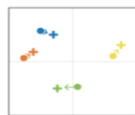
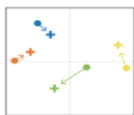
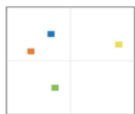
predictions of the current ensemble model



pseudo-residuals (targets of the weak learner)



pseudo-residuals
(\rightarrow) are the
targets (\blacksquare)
of the weak
learner



...

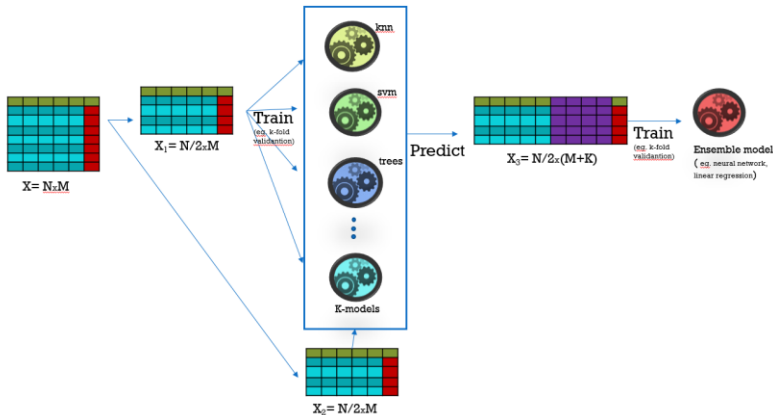
Stacking

Diferencias con bagging and boosting

- Se combinan diferentes algoritmos de aprendizaje, i.e. antes combinábamos modelos que usan la misma técnica, ahora combinamos diferentes técnicas
- Entrena un metamodelo para la combinación, antes de que usáramos algoritmos deterministas

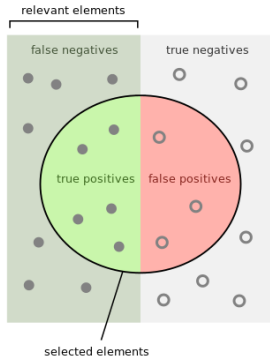
Método: dividir los datos de entrenamiento en dos pliegues (o k-fold)

- elegimos L modelos débiles y los ajustamos a los datos del primer pliegue
- para cada uno de los L modelos débiles, hacemos predicciones para las observaciones en el segundo pliegue
- ajustar el metamodelo en el segundo pliegue, usando predicciones hechas por los modelos débiles como entradas



Métricas (4)

- **F1 (F-score)** . Se emplea en la determinación de un valor único ponderado de la precisión y la recall.



How many selected items are relevant?

Precision =



How many relevant items are selected?

Recall =



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

Curva ROC: Receiver Operating Characteristic

- ▶ Son curvas en las que se presenta la sensibilidad en función de los falsos positivos para distintos puntos de corte.
- ▶ TPR mide hasta qué punto un clasificador es capaz de detectar los casos positivos correctamente, de entre todos los casos positivos. La FPR define cuántos resultados positivos son incorrectos de entre todos los casos negativos.
- ▶ Un espacio ROC se define por FPR y TPR como ejes x e y respectivamente

