

DigitalHouse >
Coding School
DATA SCIENCE

MÓDULO 4

Dimensionalidad / Clases desbalanceadas/Feature Selection

1

Entender cómo afecta la incorporación de muchos features la resolución de un problema de clasificación

2

Adquirir herramientas para resolver un problema de clasificación de clases desbalanceadas

3

Presentar un esquema de técnicas de feature selection. Introducir nociones básicas de optimización con algoritmos genéticos.

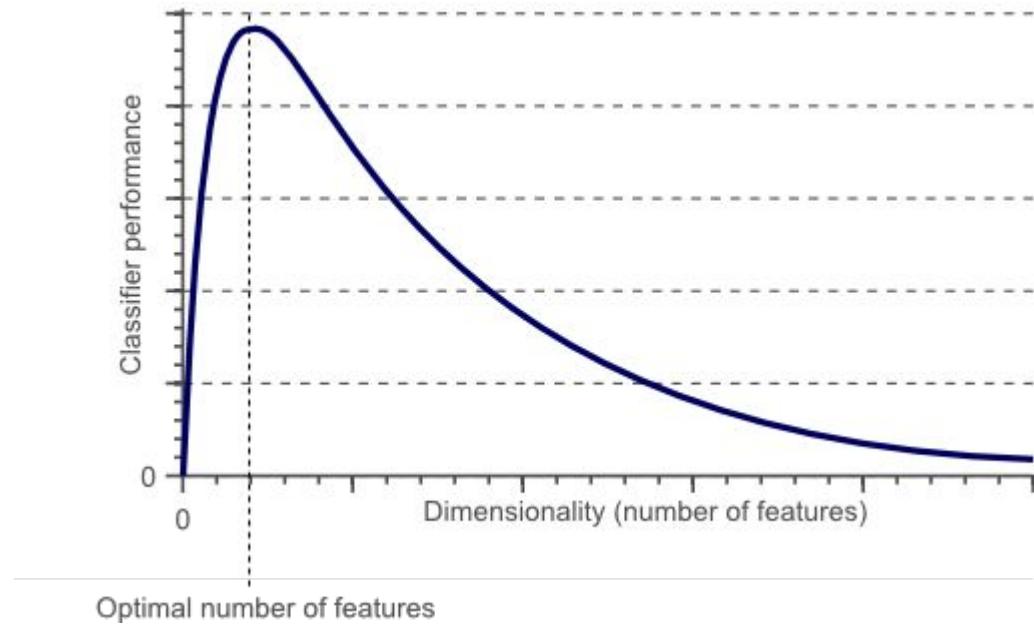
La maldición de la dimensionalidad



- El problema de la “**maldición de la dimensionalidad**” se refiere al problema de encontrar “estructura” dentro de datos en muchas dimensiones
- A priori, podría pensarse que una mayor cantidad de features es mejor... no necesariamente

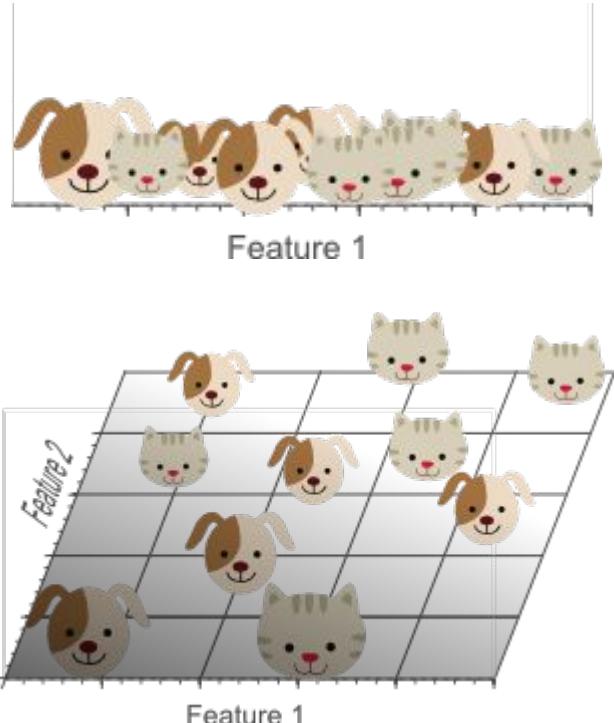
- Problema de clasificación: determinar si una imagen es un perro o un gato
- Features: promedio de cada color (RGB) de todos los pixels de una imagen
 - o Rojo
 - o Verde
 - o Azul
- Clasificador de imágenes
 - o IF $0.5 * \text{rojo} + 0.33 * \text{verde} + 0.2 * \text{azul} > 0.6 \Rightarrow \text{gato}$
 - ELSE perro;
- ¿Por qué no agregar más features? Saturación, tonos de grises, histogramas de color, etc.
- Es más, ¿por qué no agregar 100 o 200 de estos features para tratar de asegurarnos un clasificador perfecto?

- En general, al agregar features en un problema el error mejora hasta cierto punto.
- Pero a partir de ese umbral, el clasificador comienza a empeorar su performance

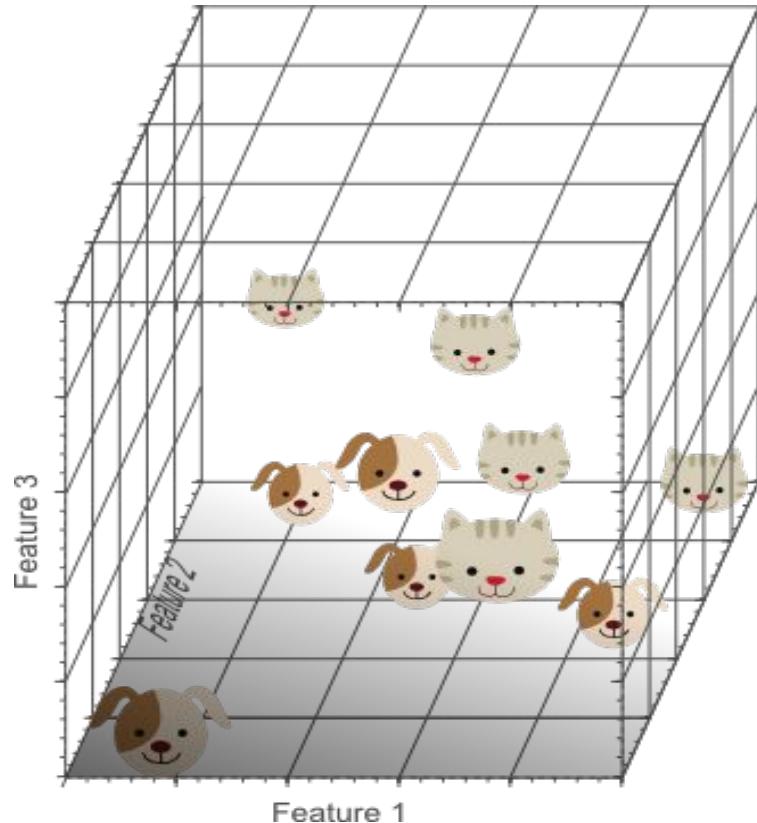


- Tenemos diez imágenes de gatos y perros y tenemos que construir un clasificador en base a estas imágenes que sirva para clasificar cualquier imagen de perros o gatos
- Usemos un clasificador lineal que haga la discriminación de forma lineal.

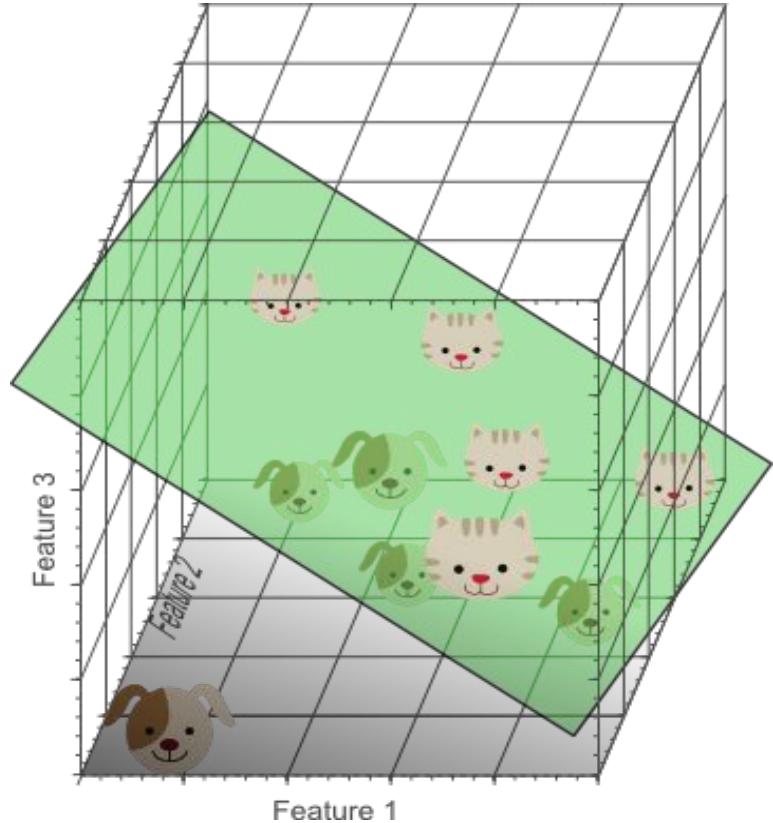
- Si usamos el feature “rojo” nuestro clasificador no funciona bien
- Agregamos la variable “verde” y no parece observarse la posibilidad de un clasificador lineal



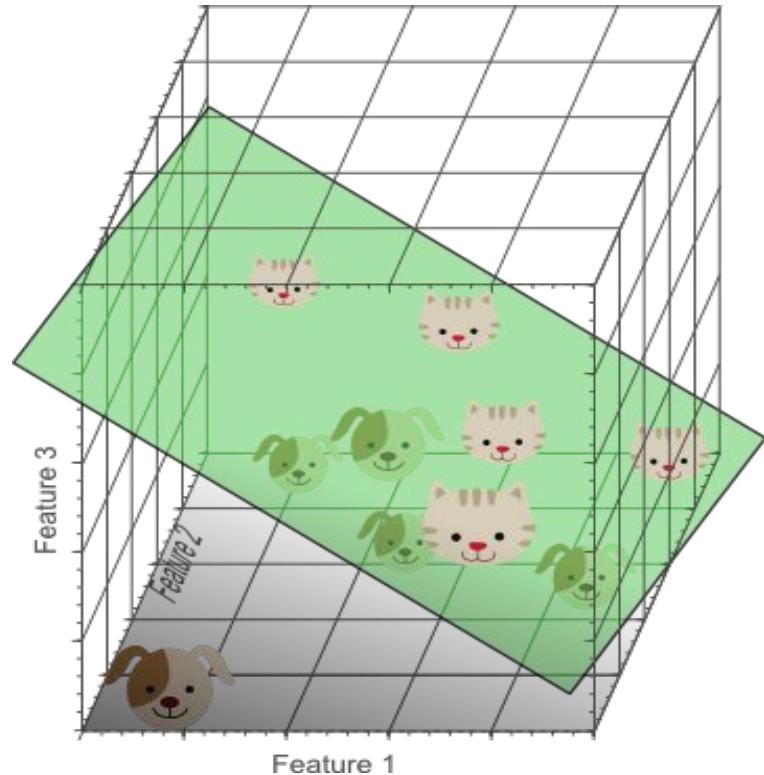
- Si agregamos una tercera dimensión (por ejemplo, "azul") la cosa parece mejorar.



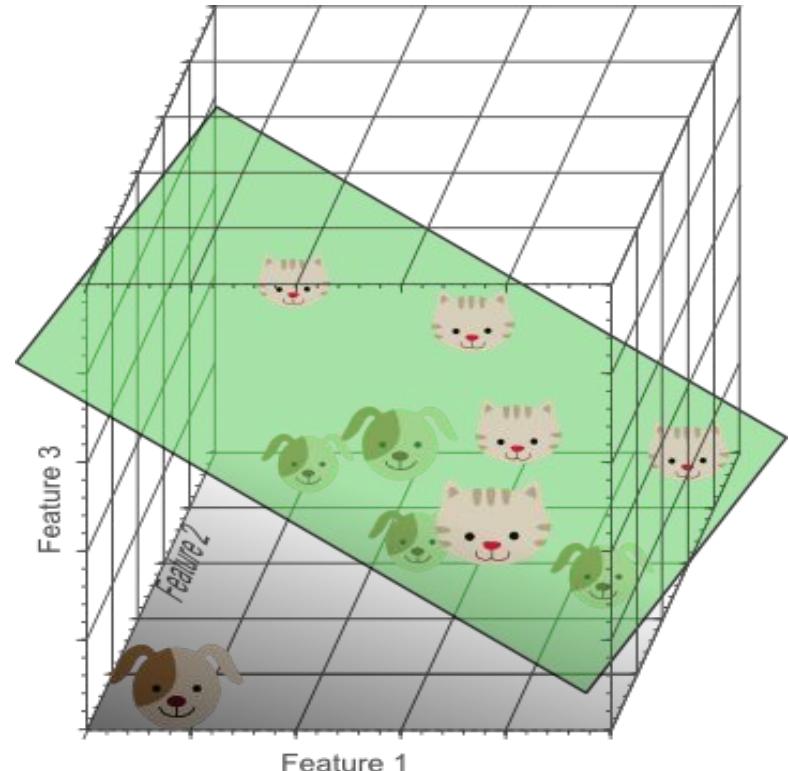
- Podemos encontrar un plano que separa perfectamente gatos y perros y que es la combinación lineal de "rojo", "verde" y "azul" en cada pixel
- Hasta acá pareciera que agregar features mejora la capacidad de clasificación del modelo.
- Sin embargo, notar cómo la densidad de las observaciones fue descendiendo al ir agregando features.



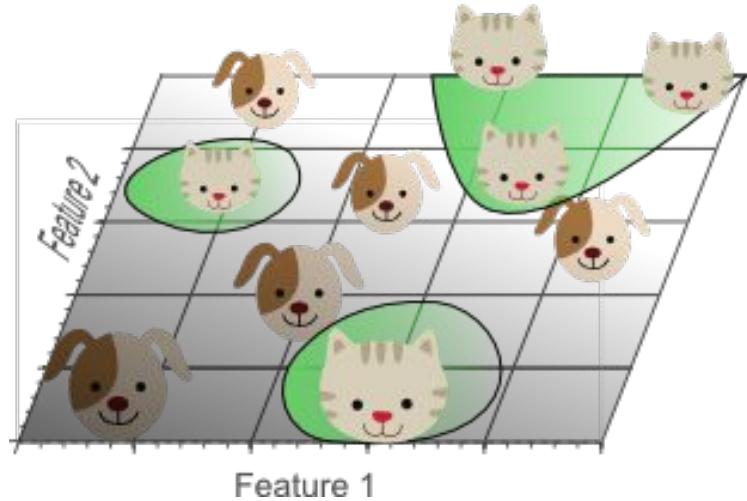
- **Caso 1:** los 10 casos cubrían el espacio de predictores que tenía una longitud de 5. => $10 / 5 = 2$ muestras por intervalo
- **Caso 2:** tenemos 10 casos (igualmente) pero en un espacio de 5×5 (25 posiciones) => $10 / 25 = 0.4$ muestras / intervalo.
- **Caso 3:** 10 muestras en un espacio de $5 \times 5 \times 5$ => $10 / 125 = 0.08$ muestras / intervalo



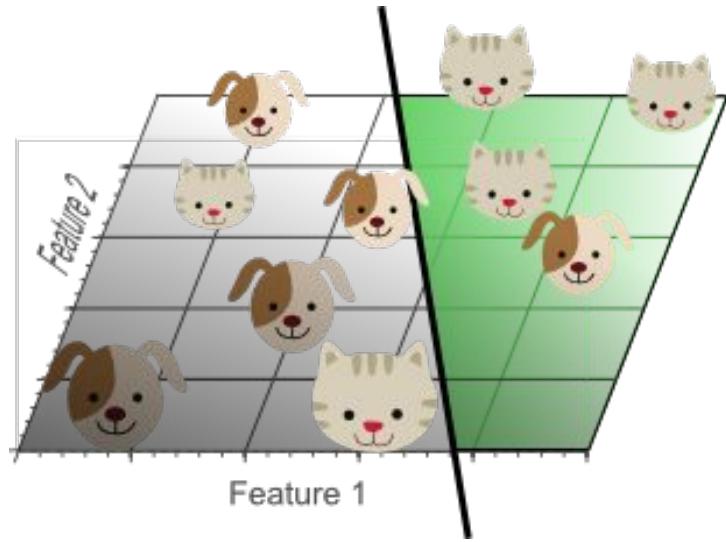
- Al **agregar features** la **dimensionalidad del espacio de predictores se incrementa** y los datos se hacen cada vez más dispersos.
- Debido a esta dispersión se hace cada vez **más fácil encontrar un hiperplano que separe a las clases** porque la probabilidad de que un dato de **training** se encuentre del lado equivocado del hiperplano se hace infinitamente pequeña
- Proyectemos este hiperplano a las dos primeras dimensiones



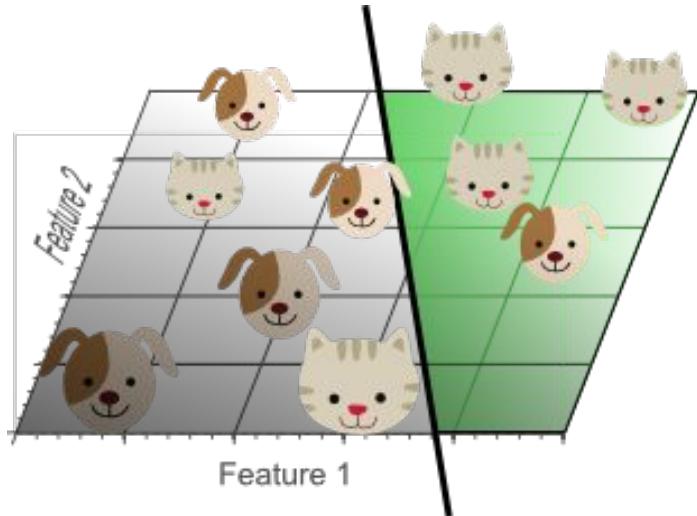
- Se ve que entrenar el clasificador con tres features es equivalente a entrenar un clasificador extremadamente complejo en un espacio de menor dimensionalidad.
- Entonces corremos el riesgo de caer en el **overfitting**.



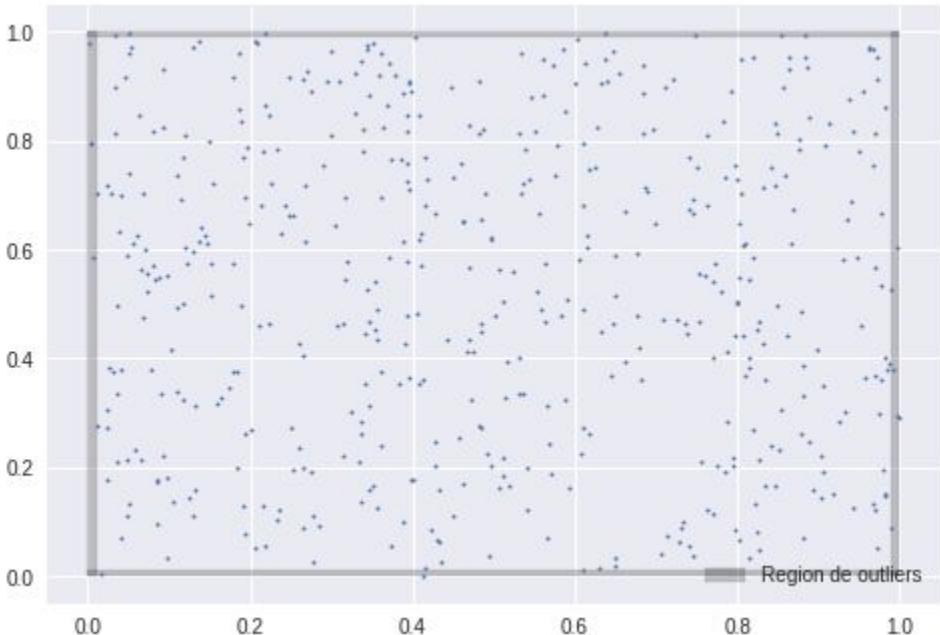
- Un clasificador lineal en dos dimensiones (aunque más simple) es más generalizable a casos “nuevos” que el clasificador lineal en tres dimensiones.
- En otras palabras: en muchos casos, al usar una **menor cantidad de features** podemos obtener un modelo con **mayor capacidad de generalización**, dado que se ve potencialmente **menos afectado por el overfitting**.



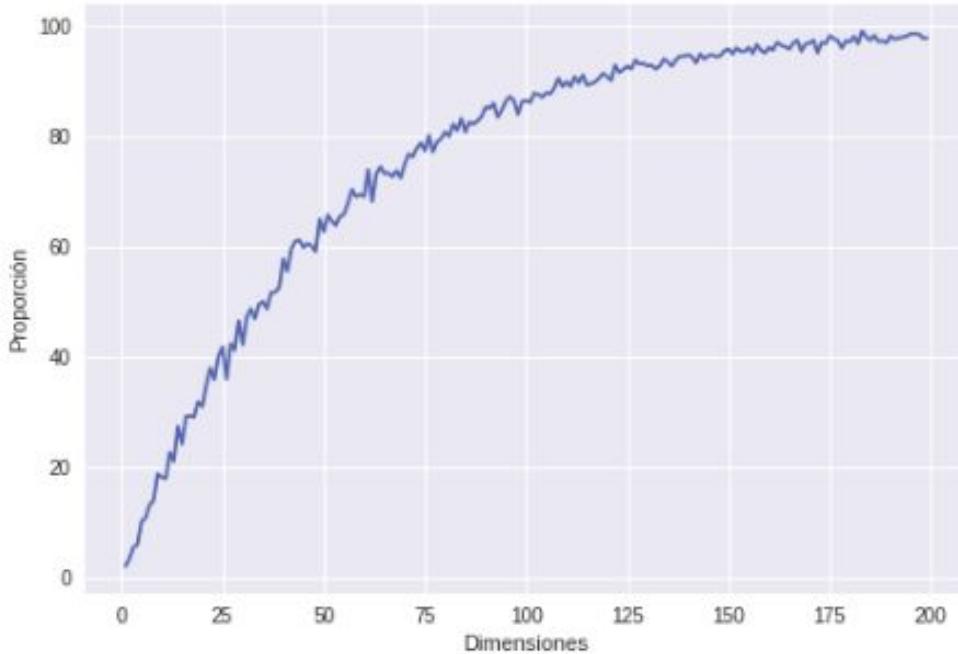
- Entonces, a mayor cantidad de features (e igual cantidad de casos) se incrementa la dispersión de los casos en el espacio de predictores,
- Además, esa mayor dispersión no se distribuye de forma homogénea en el espacio.
- En general, al incrementarse la dimensionalidad del problema, cada vez más muestras se ubican en los extremos del espacio, por lo cual, se hace más difícil construir un buen clasificador.



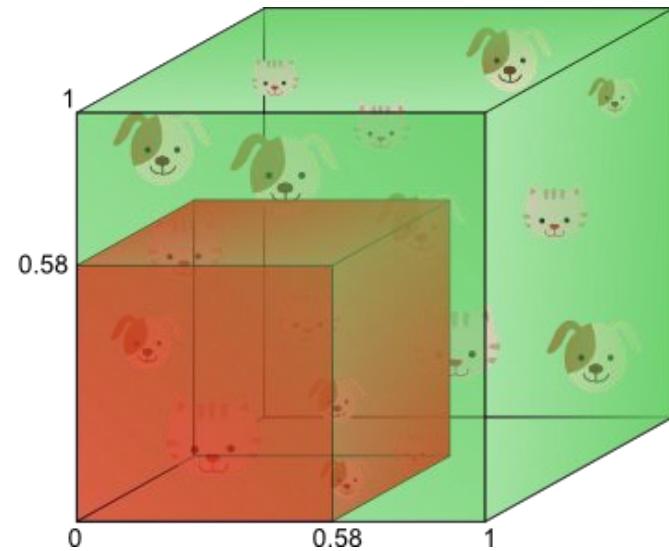
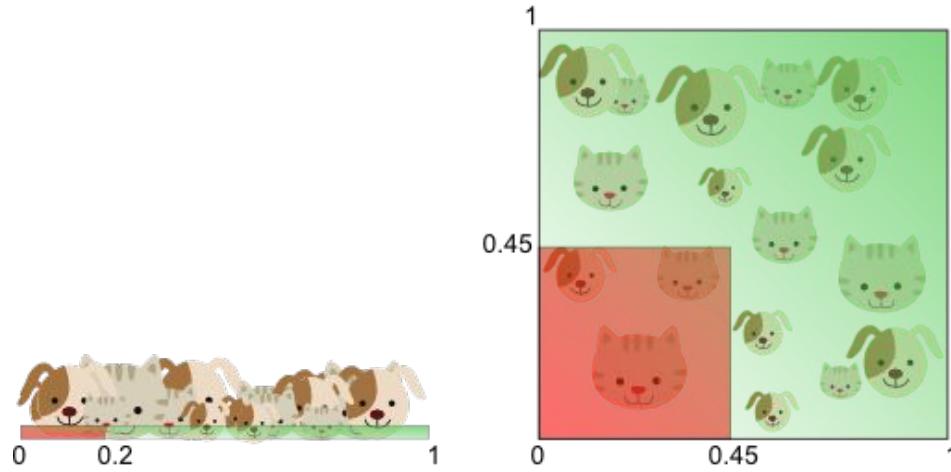
- Si tenemos una variable distribuida uniformemente en un hipercubo, ¿qué proporción de outliers encontramos?
- Podemos definir outliers como aquellos puntos que toman valores extremos en alguna de las "d" dimensiones.



- Ahora bien, ¿qué pasa a medida que aumentamos la dimensionalidad del dataset?
- Dado que no podemos graficar en más de tres dimensiones, lo que haremos será graficar la evolución de la proporción de outliers para cada nivel de dimensionalidad.



- En el ejemplo de gatos y perros la cosa se vería así:



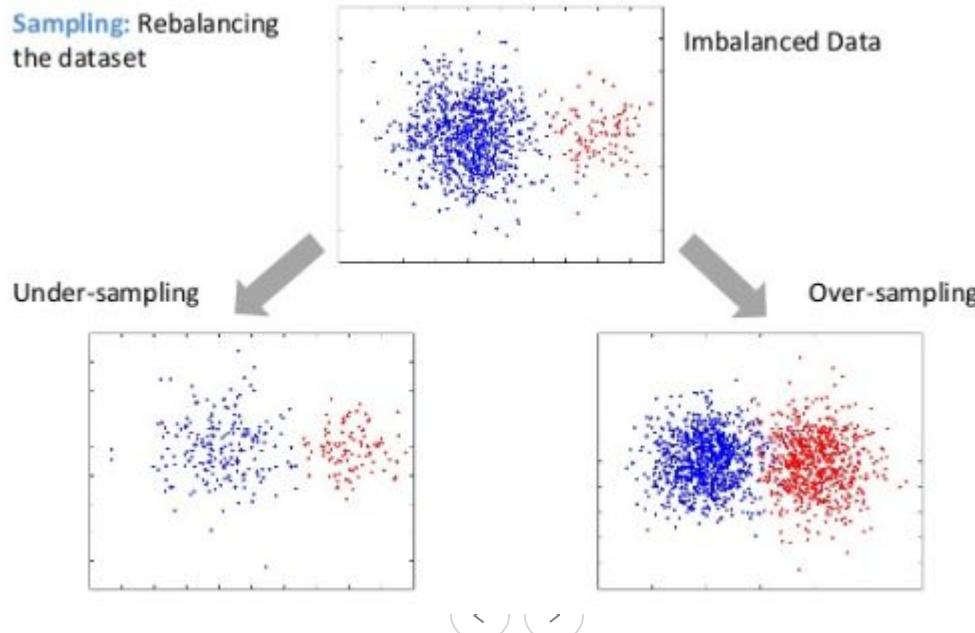
- En general, cuanto menor sea la cantidad de casos de entrenamiento, menor será la cantidad de features que podrán usarse
- Uso de **algoritmos de selección de features** (lasso, best subset selection, forward selection, backward selection, etc.)
- Uso de algoritmos de **reducción de dimensionalidad** (por ejemplo PCA).

Datasets desbalanceados



- Los datasets del mundo real suelen presentar **clases desbalanceadas**.
- ¿Qué pasa en un dataset desbalanceado con el accuracy?
El accuracy suele ser muy alto, pero esto generalmente es porque el modelo responde la “solución trivial” o algo muy cercano.
- Algunos casos comunes de datasets desbalanceados son:
 - Estudios sobre fraude
 - Diagnóstico de enfermedades

- Una de las técnicas para combatir el desbalance del dataset es el **resampling**: aumentar los casos de la clase minoritaria o descartar algunos casos de la clase mayoritaria.
- Estas técnicas se conocen como **oversampling** y **undersampling**



Existe un paquete de Python específicamente diseñado para abordar el problema del oversampling: imbalanced-learn. El paquete no forma parte de sklearn, pero adopta sus estándares para ser compatible con los demás modelos.

Implementa, entre otras técnicas **SMOTE: Synthetic Minority Oversampling Technique**.

Consiste en los siguientes pasos:

1. Se elige un punto al azar de la clase minoritaria y sus K vecinos más cercanos.
2. Se elige al azar uno de esos vecinos.
3. Se calcula el vector entre el punto seleccionado y el vecino seleccionado al azar y se lo multiplica por un número aleatorio entre 0 y 1.
4. El punto aleatorio dentro del vector es el nuevo dato para el oversampling.

- Otra técnica es tomar en cuenta el desbalance en la función de costos del algoritmo.
- Algunos algoritmos de SKlearn tienen implementada la posibilidad de incluir pesos en su función de costos. Por ejemplo:
 - o SVM
 - o Regresión Logística
- La matriz de pesos también puede ser una decisión de implementación que conviene explorar utilizando cross validation.

Práctica Guiada: Clasificación con clases desbalanceadas



- El problema de las clases desbalanceadas en tareas de clasificación debe ser tenido en cuenta dado que afecta la performance de/los modelo/s utilizados
- Es necesario tener en cuenta que no debería agregarse de forma indefinida features a los modelos dado que se incrementa la dimensionalidad del problema y junto con ello, el riesgo de overfitting.

Selección de Características

Feature Selection



1

Describir qué es la selección de características y por qué es importante

2

Reconocer varias técnicas de selección

3

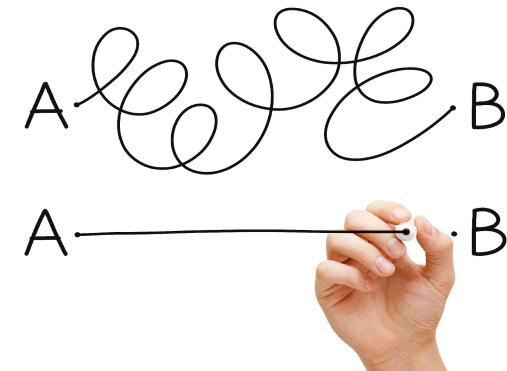
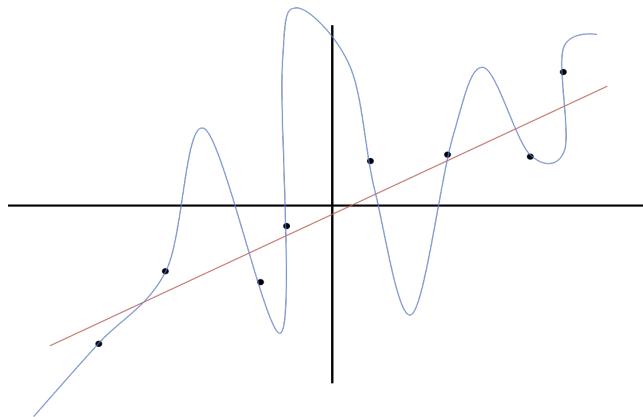
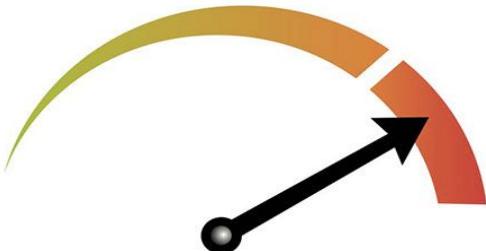
Aplicar algunos de los métodos en base a Scikit Learn



Selección de características o feature selection es una manera de reducir la dimensionalidad de nuestro dataset y de nuestro modelo para simplificarlo mientras mantiene su poder de predicción.

Es una parte importante del proceso de construcción de un modelo.

Para qué sirve?



Los problemas de machine learning parten de un vector de características que encapsula diferentes aspectos de nuestro dataset y es utilizado para entrenar un modelo que prediga una variable objetivo.

En general los datasets no vienen tabulados, limpios y definidos. Por eso es que como paso previo a construir un modelo, se deben extraer las características.

Sobre qué tipos de datos se pueden extraer características?



EJEMPLOS

Extracción y Selección



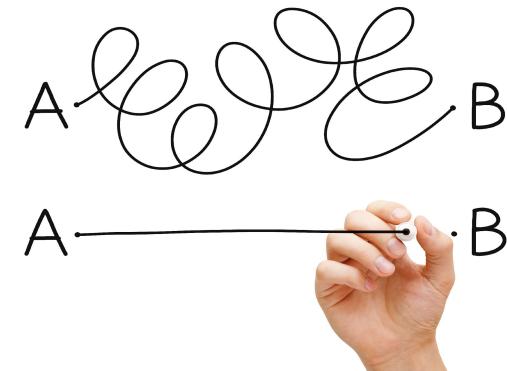
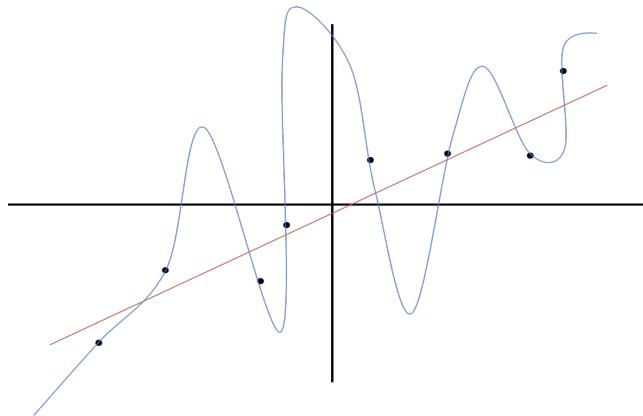
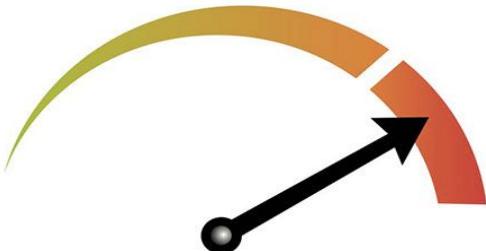
- 1 Describir qué es la selección de características y por qué es importante
- 2 Reconocer varias técnicas de selección
- 3 Aplicar algunos de los métodos en base a Scikit Learn



Selección de características o feature selection es una manera de reducir la dimensionalidad de nuestro dataset y de nuestro modelo para simplificarlo mientras mantiene su poder de predicción.

Es una parte importante del proceso de construcción de un modelo.

Para qué sirve?



Los problemas de machine learning parten de un vector de características que encapsula diferentes aspectos de nuestro dataset y es utilizado para entrenar un modelo que prediga una variable objetivo.

En general los datasets no vienen tabulados, limpios y definidos. Por eso es que como paso previo a construir un modelo, se deben extraer las características.

Sobre qué tipos de datos se pueden extraer características?



Para el caso de texto, una unidad de datos podría corresponderse con un documento, o una oración, o un párrafo.

Debemos aplicar técnicas para extraer de ahí las características que nos interesen y mapearlas en un vector numérico. Por ejemplo: la función CountVectorizer.

Text	the	cat	is	on	table	blue	sky	...
The cat is on the table	2	1	1	1	1	0	0	...
The table is blue	1	0	1	0	1	1	0	...
The sky is blue	1	0	1	0	0	0	1	...
...

Podemos observar que el elemento (característica) "the" del vector obtenido no tendrá valor predictivo, por lo que podría descartarse en la selección.

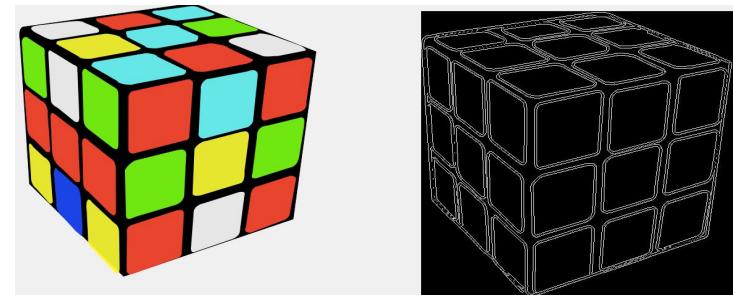
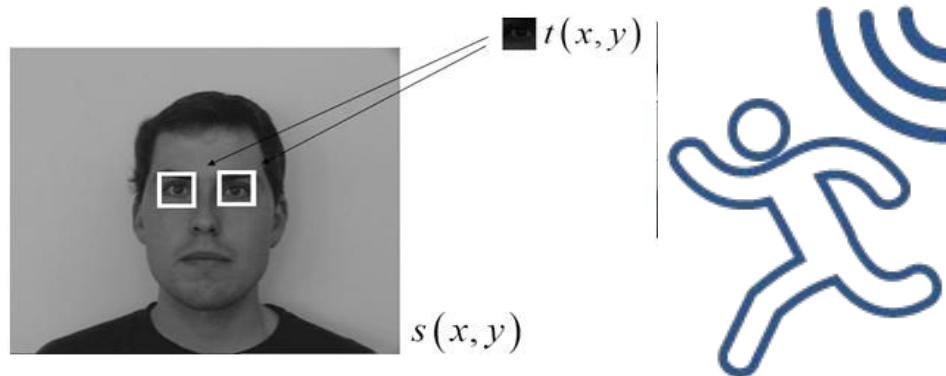
Otro caso frecuente es cuando obtenemos un dataset extraído desde una base de datos. El mismo puede ser resultado de una consulta o un dump de tabla. Normalmente cada columna sería una característica de nuestro modelo

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin		12209	Germany	030- 0074321	030- 0076545
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México D.F.		05021	Mexico	(5) 555- 4729	(5) 555- 3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos2312	México D.F.		05023	Mexico	(5) 555- 3932	
...

Podemos observar que hay un número importante de columnas que podrían no ser relevantes para nuestro problema.

En procesamiento de imágenes y video se aplican otras técnicas tales como

- Detección de Bordes y Esquinas
- Histograms of Gradients (HOG)
- Template Matching
- Detección de Movimiento



- Parte de lo que habitualmente se llama proceso de “**Feature Engineering**”:
 - **Feature Extraction** (como por ejemplo PCA)
 - **Feature Selection** (lo que vamos a ver ahora)
- El proceso de “**Feature Selection**” **busca identificar y seleccionar las variables más relevantes** para el entrenamiento de los modelos
- ¿Por qué realizar feature selection?
 - potenciales **problemas computacionales**: si contamos con muchos predictores el tiempo de cómputo podría incrementarse demasiado (obviamente, esto dependerá de cada algoritmo)
 - la presencia de **predictores irrelevantes** podría afectar la performance los modelos.

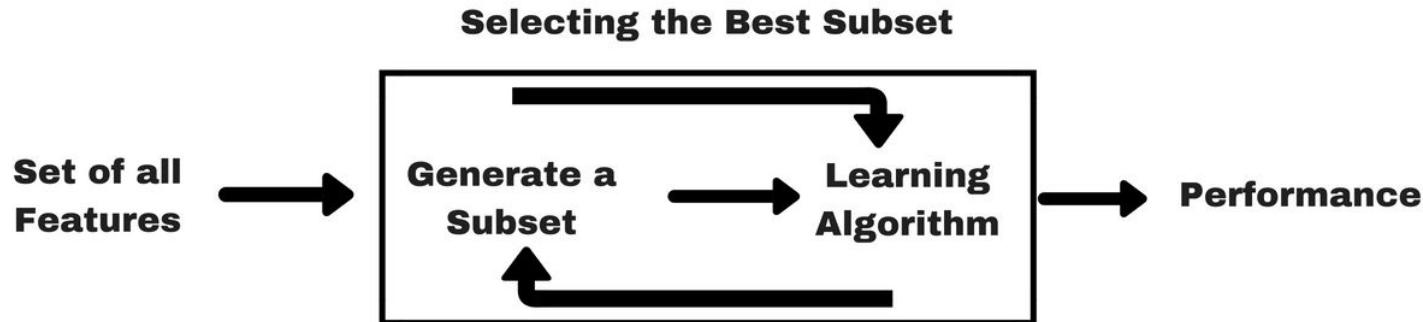
- Un enfoque de **fuerza bruta** sería realizar una **búsqueda exhaustiva** de todas las combinaciones posibles de features y encontrar el mejor conjunto (método llamado “best subset”).
- Tal enfoque resulta **impracticable en términos computacionales**
 - solamente en un modelo de regresión considerando solamente las interacciones entre features (es decir, sin considerar términos polinómicos) tenemos 2^p modelos posibles: en una regresión con 20 predictores tenemos aproximadamente 1.050.000 modelos posibles
- Es por eso que suelen utilizarse otros **métodos** llamados, a veces, “**greedy**” para llevar a cabo los procesos de feature selection.
 - Filter Methods
 - Wrapper Methods
 - Embedded Methods

- Buscan **rankear las variables en función de su “importancia”**. Aquellas menos importantes son eliminadas del algoritmo. **Se aplican “antes” de entrenar los modelos.**
- Habitualmente, se define algún umbral por debajo del cual las variables son consideradas poco importantes y por ello eliminadas (filtradas).



- ¿Cómo medir la relevancia de una variable?
 - Una variable puede considerarse irrelevante si es condicionalmente independiente de las etiquetas de clase. Básicamente, se establece que una variable para ser relevante, no puede ser independiente de las etiquetas de clase, es decir, la variable que no tiene ninguna influencia en las etiquetas de clase puede ser descartada.
- Dos indicadores habitualmente utilizados (no son los únicos...)
 - Correlation criteria:
$$R(i) = \frac{cov(x_i, Y)}{\sqrt{var(x_i) * var(Y)}}$$
 - Mutual Information:
$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right),$$

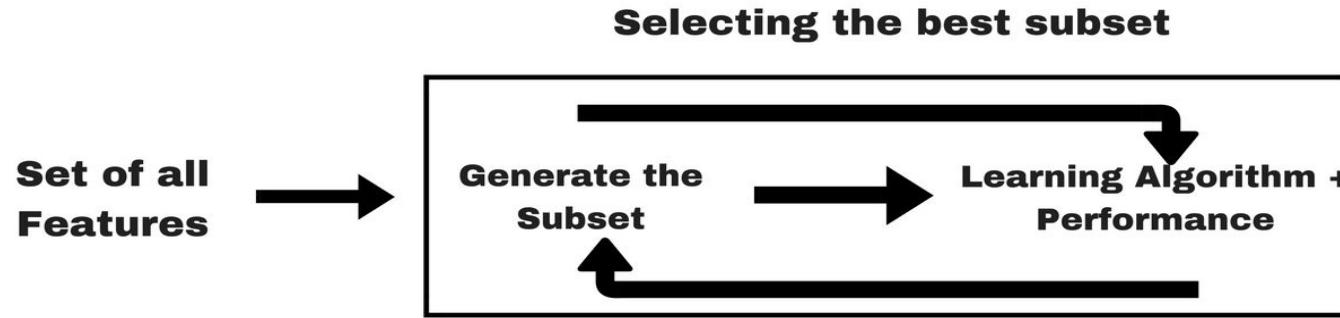
- Estos métodos evalúan múltiples modelos usando procedimientos que agregan o quitan predictores para encontrar una combinación óptima que maximiza la performance predictiva del modelo.
- En esencia, son algoritmos de búsqueda que tratan a los predictores como los inputs y utilizan la performance de los modelos como el output a ser optimizado.



- Uno de los más utilizados
 - Recursive Feature Elimination: a veces llamada "Backward Selection":
 - Se fitea un modelo con todos los predictores. Cada predictor se rankea en función de su importancia (medida de alguna forma)
 - En cada iteración el conjunto de predictores mejor rankeados son retenidos y el modelo se fitea nuevamente y se evalúa su performance.
 - Se repite recursivamente hasta llegar a un criterio de finalización
- Tienden a ser más intensivos computacionalmente.

	FILTER	WRAPPER
Evaluación	Relevancia de cada variable en función de la correlación entre X's e y	"Utilidad" de un subset de variables en función de la performance que tienen al entrenar un modelo
Técnicas de validación	Métodos estadísticos (p-valores, betas, etc.)	CrossValidation, Bootstrap. etc.
Velocidad	No entranan modelos => más rápidos	Algoritmos de búsqueda: computacionalmente más intensivos

- Combinan algo de los métodos de filtro y los wrappers. Suele decirse que van detectando las mejores features a medida que el modelo va siendo creado.



- Se trata de métodos que tienen "incorporados" métodos de selección de features.
- Ejemplo: regresión LASSO

Algoritmos Genéticos

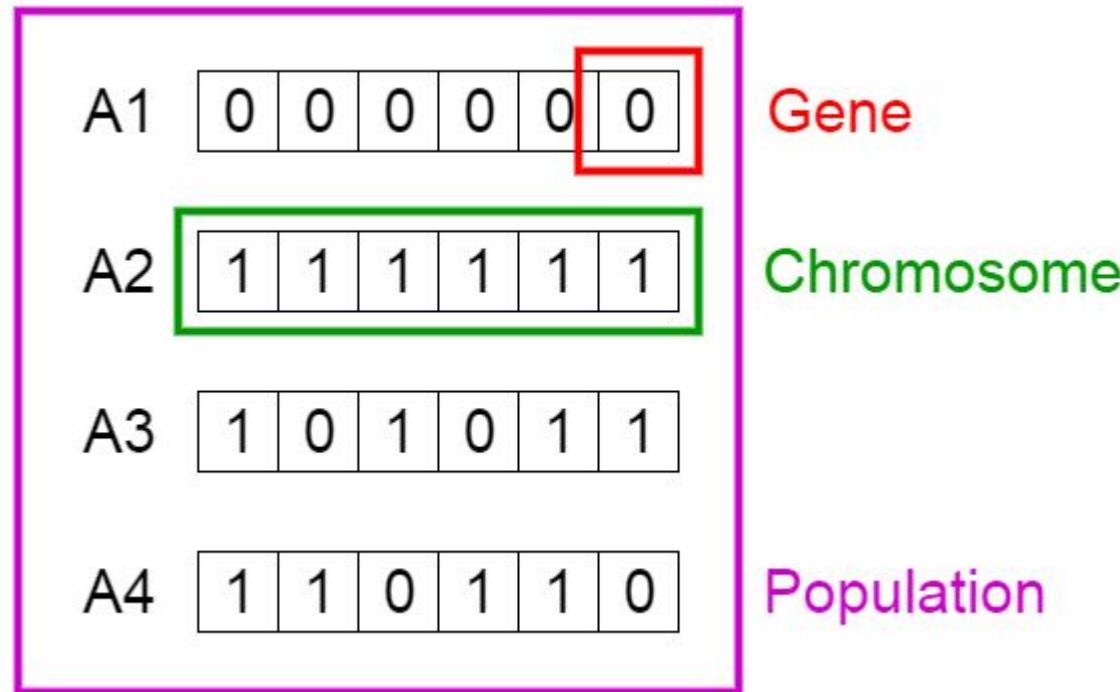


- Son heurísticas de búsqueda.
- Inspirados en “selección natural”.
- Nos ayudan a evitar el problema del sobreajuste al conjunto de validación en la búsqueda de hiper parámetros óptimos.
 - Este sobreajuste ocurre pese al uso de validación cruzada.
 - El problema surge cuando se prueban demasiadas combinaciones de hiper parámetros dada la cantidad de datos.
 - Information leaking.
 - La selección de features constituye un caso de selección de hiperparámetros.

— Un algoritmo genético se puede caracterizar por las siguientes fases

- Población inicial
- Evaluación de la función de fitness
- Selección
- Crossover
- Mutación

- Población inicial.
 - Un individuo se caracteriza por un conjunto de parámetros (variables) conocidos como genes.
 - Los genes se unen en una cadena para formar un cromosoma (solución).
 - En un algoritmo genético, el conjunto de genes de un individuo se representa mediante una cadena, en términos de un alfabeto. Usualmente, se usan valores binarios (cadena de 1s y 0s).
 - Decimos que codificamos los genes en un cromosoma.

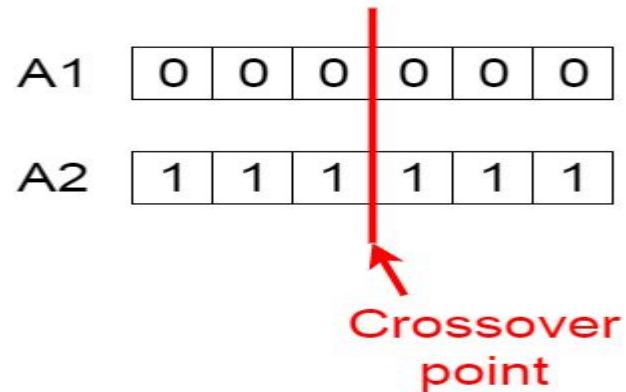


- Función de fitness.
 - La función de fitness determina qué tan adecuado es un individuo (la capacidad de ese individuo para competir con otros).
 - Da una puntuación de fitness a cada individuo. La probabilidad de que un individuo sea seleccionado para la reproducción se basa en su puntaje de condición física.

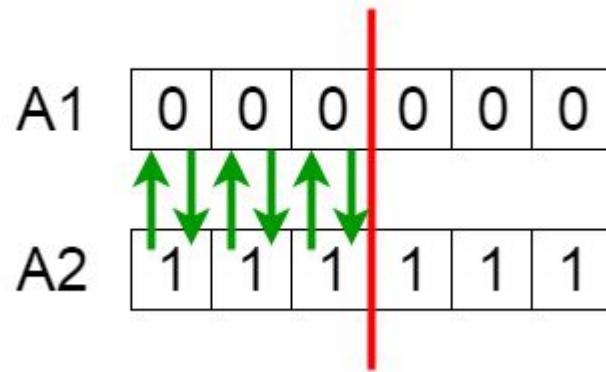
— Selección.

- La idea de la fase de selección es seleccionar a los individuos más aptos y dejar que pasen sus genes a la siguiente generación.
- Se seleccionan dos pares de individuos (padres) en función de sus fitness scores.
- Las individuos con mejor score tienen más posibilidades de ser seleccionadas para su reproducción.

- Crossover.
 - Para cada par de padres se eligen un punto de crossover aleatoriamente dentro de los genes.



- Crossover.
 - La descendencia es creada intercambiando los genes de los padres entre ellos hasta que el punto de crossover es alcanzado.



- Crossover.
 - Los nuevos descendientes son agregados a la población.

A5

1	1	1	0	0	0
---	---	---	---	---	---

A6

0	0	0	1	1	1
---	---	---	---	---	---

— Mutación.

- En ciertos individuos entre los descendientes formados algunos de sus genes pueden estar sujetos a mutación con una baja probabilidad.
- Esto implica que algunos de los bits del string puede ser rotados.
- La idea es mantener la diversidad dentro de la población y prevenir una convergencia prematura.

- Mutación.

Antes de la mutación

A5	1	1	1	0	0	0
----	---	---	---	---	---	---

Después de la mutación

A5	1	1	0	1	1	0
----	---	---	---	---	---	---

- Terminación. El algoritmo termina si la población ha convergido, cuando no produce descendientes con performance significativamente diferente a la de sus predecesores.

IMPORTANCIA DE CARACTERÍSTICA



Es importante poder identificar cuáles son las características con mayor valor predictivo en un modelo.

Esto permite:

- Entender y explicar mejor el modelo y sus componentes.
- Concentrar y optimizar esfuerzos en recolección y data-quality

Cómo determinar la importancia de una característica?

- Hay modelos, como los árboles de decisión, que permiten obtener esta información de manera intrínseca
- En los modelos lineales, como la Regresión Logística, la importancia está dada por el tamaño de cada coeficiente del modelo, con una condición...

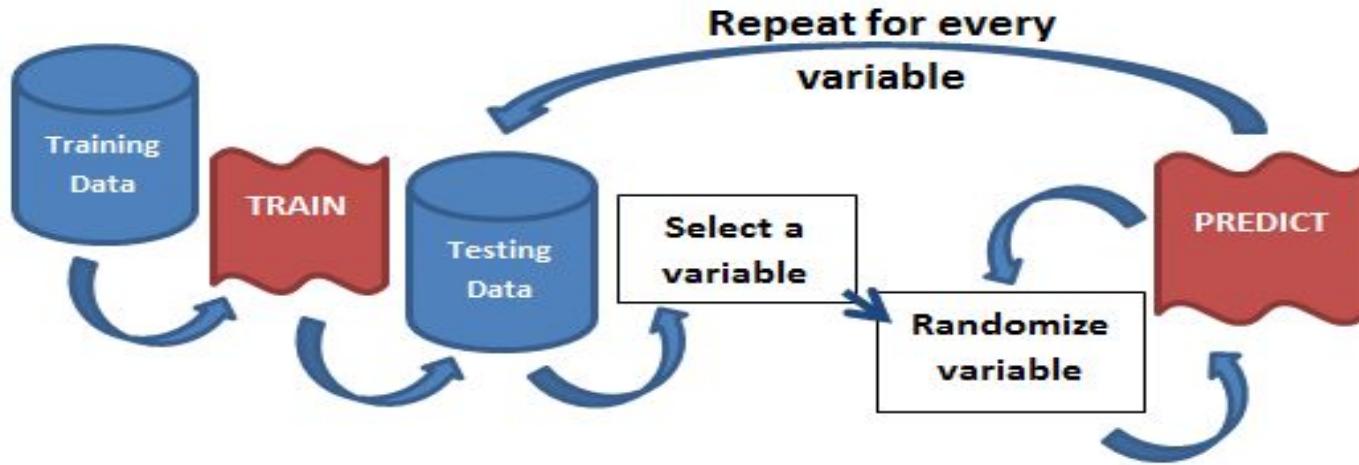
Para poder comparar los valores de coeficientes dentro de un modelo lineal, las características pertenecientes al mismo deben encontrarse en la misma escala.

De lo contrario, una diferencia unitaria en un coeficiente, tendrá un impacto distinto a la misma variación para una característica en una escala distinta.

Métodos de Escalamiento:

- **Estandarización:** Imita a la distribución normal (media=0 y varianza=1).
$$x' = (x - \text{mean}) / \text{std_dev}$$
- **Min-Max:** Conociendo el intervalo de los datos u obteniéndolo del dataset se aplica $x' = (x - \text{min}) / (\text{max} - \text{min})$
- **Normalización:** Se divide x por la suma de todas las características. Es similar a obtener el vector normal o versor (de longitud 1).

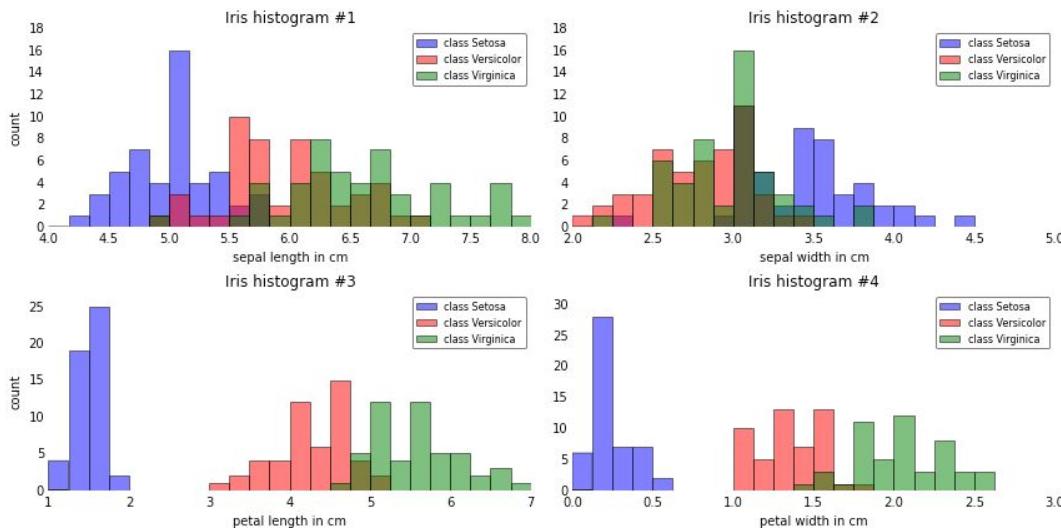
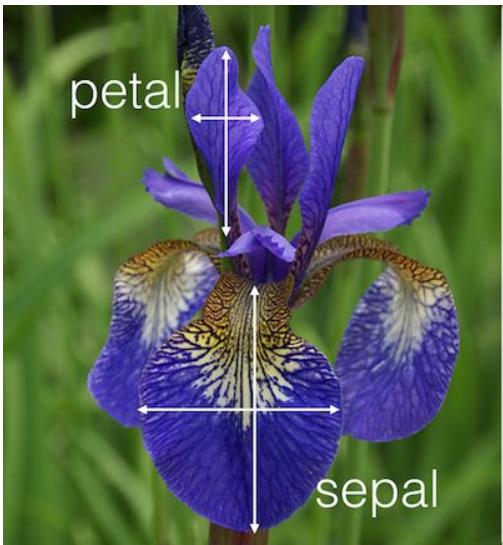




DEMO: IMPORTANCIA DE CARACTERÍSTICA



Para esta demo, vamos a utilizar el dataset “Iris”



Cargamos la información:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

iris = load_iris()

X = iris.data
y = iris.target
```

```
iris.feature_names
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

Inicializamos el modelo de Regresión Logística:

```
model = LogisticRegression()  
model.fit(X, y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,  
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,  
verbose=0, warm_start=False)
```

Exponemos el atributo “coef_” del modelo ():

```
model.coef_
```

```
array([[ 0.41498833,  1.46129739, -2.26214118, -1.0290951 ],  
       [ 0.41663969, -1.60083319,  0.57765763, -1.38553843],  
      [-1.70752515, -1.53426834,  2.47097168,  2.55538211]])
```

Formateamos el atributo “coef_” para una salida más amigable:

```
coeffs = pd.DataFrame(model.coef_, columns = iris.feature_names, index =iris.target_names)  
coeffs
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
setosa	0.414988	1.461297	-2.262141	-1.029095
versicolor	0.416640	-1.600833	0.577658	-1.385538
virginica	-1.707525	-1.534268	2.470972	2.555382

Mirando los coeficientes obtenidos, y siendo el modelo una Regresión Logística, podemos concluir por ejemplo que “petal length” es la característica más importante para determinar “setosa”, no?

NO!!! Dada que las características no están normalizadas, no podemos aseverar dicha conclusión.

Entonces: apliquemos normalización y luego volvamos a correr el modelo:

```
from sklearn.preprocessing import StandardScaler  
X_norm = StandardScaler().fit_transform(X)  
  
model.fit(X_norm, y)  
  
coeffs = pd.DataFrame(model.coef_, columns = iris.feature_names, index =iris.target_names)  
coeffs
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
setosa	-0.810166	1.393699	-1.687386	-1.518991
versicolor	0.130380	-1.246338	0.789195	-0.889440
virginica	0.012990	-0.144535	1.863173	2.698873

Ahora los coeficientes reflejan la importancia relativas de sus características asociadas.

PRÁCTICA GUIADA: Feature Selection



PRÁCTICA INDEPENDIENTE:

Documentación de Feature Selection



- Formar grupos
- Ir a la documentación de la API: http://scikit-learn.org/stable/modules/feature_selection.html
- Consultar la información de los siguientes métodos
 - `feature_selection.GenericUnivariateSelect(...)`
 - `feature_selection.SelectPercentile(...)`
 - `feature_selection.SelectKBest([score_func, k])`
 - `feature_selection.SelectFpr([score_func, alpha])`
 - `feature_selection.SelectFdr([score_func, alpha])`
 - `feature_selection.SelectFromModel(estimator)`
 - `feature_selection.SelectFwe([score_func, alpha])`
 - `feature_selection.RFE(estimator[, ...])`
 - `feature_selection.RFECV(estimator[, step, ...])`
 - `feature_selection.VarianceThreshold([threshold])`

Investigar y responder:

- Breve explicación del método
- En qué situación es recomendable?
- Se puede usar en clasificación y en regresión?
- Se puede usar en un pipeline?

CONCLUSIONES

SELECCIÓN DE CARACTERÍSTICAS

Los conceptos vistos hoy nos permiten:

- Reducir la dimensionalidad de nuestro dataset a través de la selección de ciertos features
- Se busca eliminar las características menos relevantes nuestro modelo, buscando mejorar su performance predictiva
- Esta selección puede tomar diferentes formas:
 - Filtros
 - Wrappers
 - Métodos “embedidos”