



Curso Profesional de Scikit-learn

Ariel Ortiz Beltrán

¿Por qué Scikit-learn?

Origen de la librería

Scikit-learn comienza su historia en 2007 por David Cournapeau como un proyecto dentro del Google Summer of Code.

Matthieu Brucher continúa como parte de su tesis.



Scikit-learn en la actualidad


- La versión estable actual es la **v0.22**
- Está siendo usada por organizaciones como Spotify, J.P Morgan, Betaworks, entre muchas otras.
- Cuenta con buena documentación y una comunidad bastante activa a nivel mundial, lista de correo, Quora, Stackexchange.



¿Por qué Scikit-learn?

- Curva de aprendizaje suave.
- Es una librería muy versátil.
- Comunidad de soporte.
- Usado en producción.
- Integración con librerías externas.

<https://scikit-learn.org/>

 [Install](#) [User Guide](#) [API](#) [Examples](#) [More ▾](#)

scikit-learn

Machine Learning in Python

[Getting Started](#) [What's New in 0.22](#) [GitHub](#)

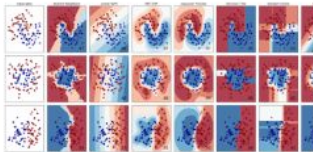
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



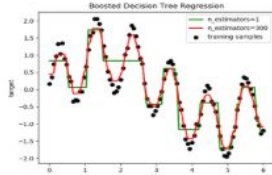
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...




Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



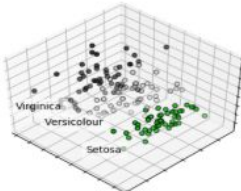
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



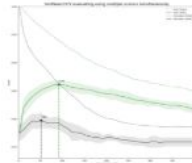
Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation, metrics, and more...



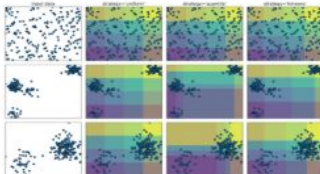
Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

Visión general del curso



Módulos de Scikit-learn

1. Clasificación.
2. Regresión.
3. Clustering.
4. Preprocesamiento.
5. Reducción de la dimensionalidad.
6. Selección del modelo.

Algunas magias de Scikit-learn

- ¿Cómo nos puede ayudar en el preprocesamiento de los datos?
- ¿Qué modelos podemos usar para resolver problemas específicos?
- ¿Cuál es el procedimiento a seguir para optimizar nuestros modelos?

¿Cómo aprenden
las máquinas?

Las máquinas aprenden de los datos disponibles

Desde el punto de vista de los datos, podemos aplicar tres técnicas según la naturaleza y disponibilidad de los mismos:

1. Aprendizaje supervisado
2. Aprendizaje no supervisado
3. Aprendizaje por refuerzo

Aprendizaje Supervisado (por observación)

Si de nuestro conjunto de datos podemos extraer con anticipación información precisa del resultado que esperamos.

Usaremos entonces,
Aprendizaje supervisado.



Manzana



Uvas

| Aprendizaje Por Refuerzo (prueba y error)

Si no tenemos información precisa sobre lo que esperamos, pero sí podemos evaluar si una decisión tomada por la máquina es buena o mala.

Usaremos entonces,
Aprendizaje por refuerzo.



Aprendizaje No Supervisado (por descubrimiento)

Finalmente, si no sabemos qué esperar de nuestros datos y queremos explorar, la estructura o las relaciones de nuestro dataset.

El camino es,

Aprendizaje no supervisado.



NOTA

El Machine Learning es solamente una de las posibles ramas que tiene la Inteligencia Artificial, para otro tipos de problemas existen:

1. Algoritmos evolutivos.
2. Lógica difusa.
3. Agentes.
4. Sistemas expertos.

¿Qué tipos
de problemas
podemos resolver
con Scikit-learn?

Algunas limitaciones

1. No es herramienta de Computer Vision.
2. No se puede correr en GPUs.
3. No es una herramienta de estadística avanzada.
4. No es muy flexible en temas de Deep Learning.

Clasificación

Necesitamos etiquetar nuestros datos para que encajen en alguna de ciertas categorías previamente definidas.

¿Es cáncer o no es cáncer?

¿La imagen pertenece a un Ave, Perro o Gato?

¿A qué segmento de clientes pertenece determinado usuario?

Regresión

Cuando necesitamos modelar el comportamiento de una variable continua, dadas otras variables correlacionadas.

Predecir el precio del dólar para el mes siguiente.

El total de calorías de una comida dados sus ingredientes.

La ubicación más probable de determinado objeto dentro de una imagen.

Clustering

Queremos descubrir subconjuntos de datos similares dentro del dataset.

Queremos encontrar valores que se salen del comportamiento global.

Identificar productos similares para un sistema de recomendación.

Descubrir el sitio ideal para ubicar paradas de buses según la densidad poblacional.

Segmentar imágenes según patrones de colores y geometrías.

¿Qué matemáticas
necesitamos?



La cortina de fondo

Varias de las técnicas que usamos para que los computadores aprendan están inspiradas en el mundo natural.

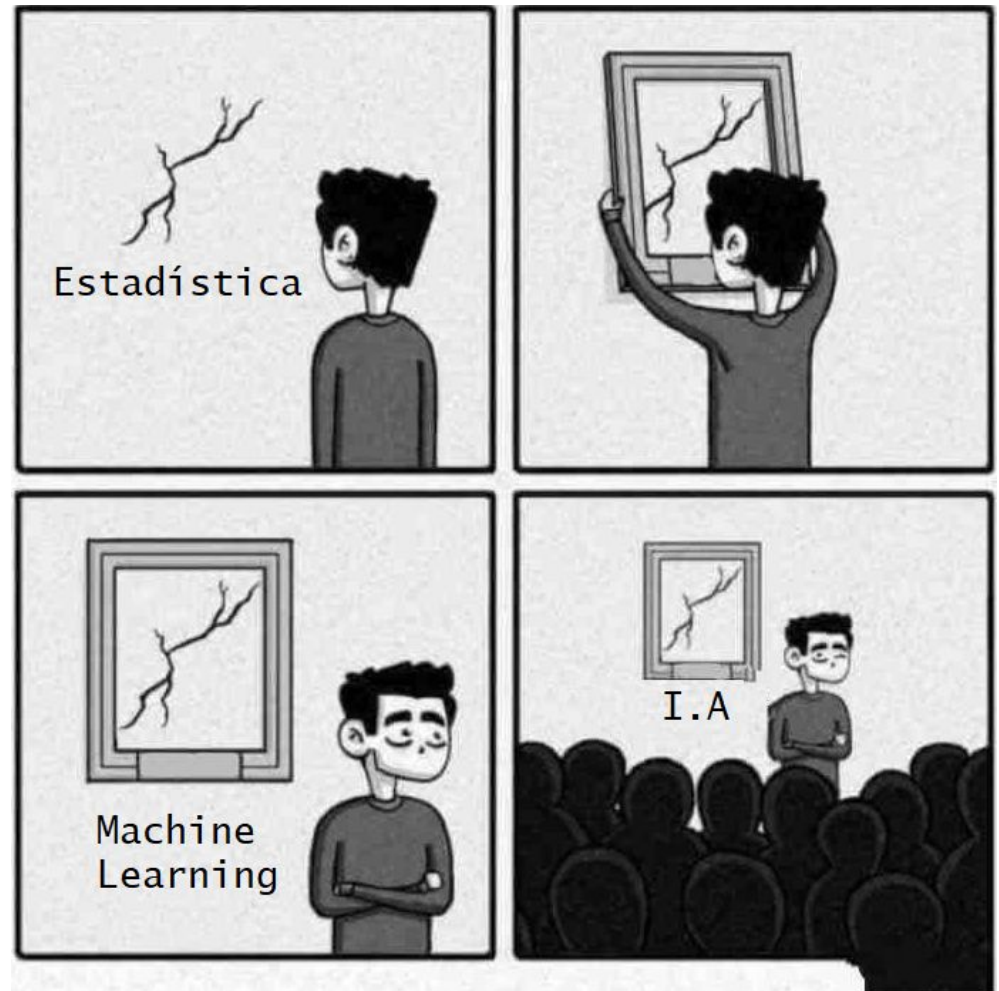
- Redes neuronales,
- Aprendizaje por refuerzo,
- Algoritmos evolutivos...

La cortina de fondo

Sin embargo,

Son modelos simplificados que funcionan gracias a la matemática que hay de fondo.

El Machine learning es en esencia matemática y estadística.



De Padawans a Maestros

Para obtener un buen resultado en un proyecto de ML **NO NECESITAMOS** ni matemáticas ni estadística avanzada.

Pero,

Para entender ¿Por qué lo obtuvimos? Y proyectar esa solución a otros contextos. **SÍ.**





Matemáticas a repasar:

1. Funciones y trigonometría
2. Álgebra Lineal
3. Optimización de funciones
4. Cálculo básico



Estadística a repasar:

1. Probabilidad básica
2. Combinaciones y Permutaciones
3. Variables Aleatorias y distribuciones
4. Teorema de Bayes.
5. Pruebas estadísticas.



NOTA

La estrategia para este curso será desde el desarrollo de software y la computación.

Usaremos Scikit-learn para ayudarnos a cubrir algunos de estos vacíos conceptuales de una manera que beneficie a nuestro modelo.

Configuración de nuestro entorno

Código

Datasets que usaremos en el curso

Data Science para la Felicidad y el Bienestar





World Happiness Report 2019

Es un dataset que desde el 2012 recolecta variables sobre diferentes países y las relaciona con el nivel de felicidad de sus habitantes.

Vamos a usar este dataset para trabajar algunos temas especiales de regresiones.

<https://www.kaggle.com/unsdsn/world-happiness>

The ultimate halloween candy power ranking

Un estudio online con 269 mil votos desde más de 8371 IPs diferentes. Para 85 tipos de dulces diferentes se evaluaron tanto las características del dulce como la opinión y satisfacción para generar comparaciones.

Vamos a usar este dataset para trabajar algunos temas especiales de clustering.

<https://www.kaggle.com/fivethirtyeight/the-ultimate-halloween-candy-power-ranking>



Heart disease Dataset

Subconjunto de variables de un estudio realizado en 1988 en diferentes regiones del planeta para predecir el riesgo de sufrir una enfermedad relacionada con el corazón.

Vamos a usar este dataset para trabajar algunos temas especiales de clasificación.

<https://www.kaggle.com/johnsmith88/heart-disease-dataset>



Material adicional

¿Cómo afectan
nuestros features
a los modelos de
machine learning?

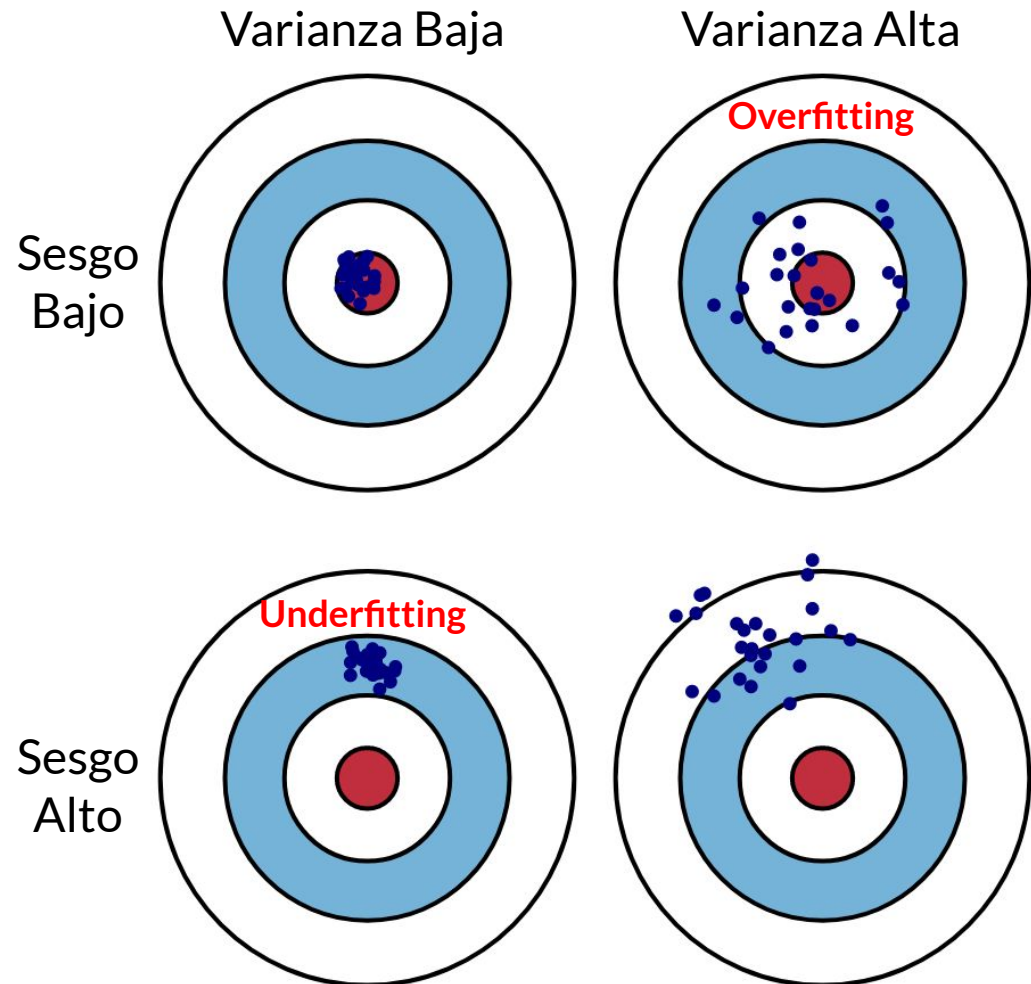
Más features siempre es mejor ¿Verdad?

Respuesta corta: **NO**

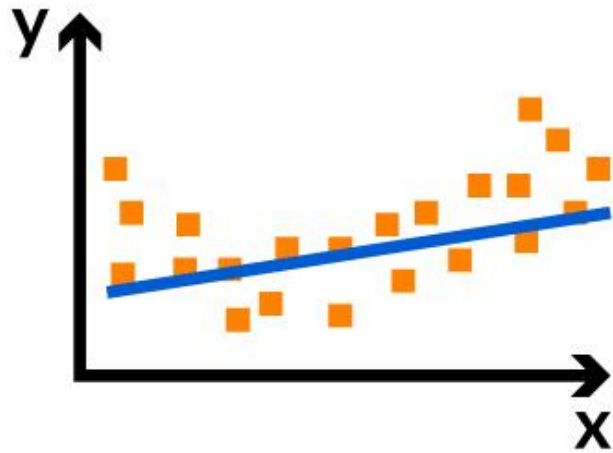
- Features irrelevantes para inferencia.
- Valores faltantes.
- Introducción de ruido.
- Costo computacional.

Sesgo y Varianza (Bias & Variance)

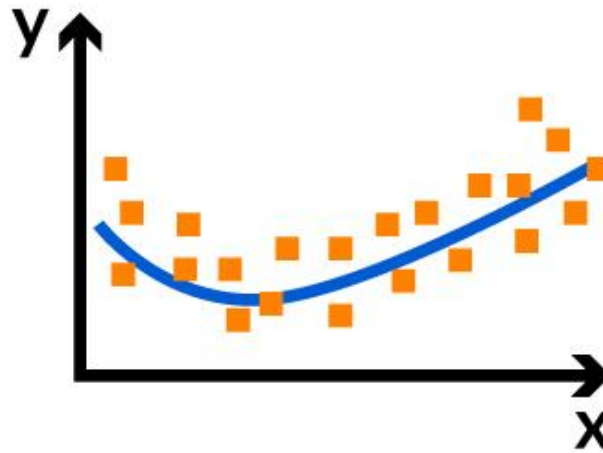
Una mala selección de Features nos puede llevar a alguno de estos dos escenarios indeseados.



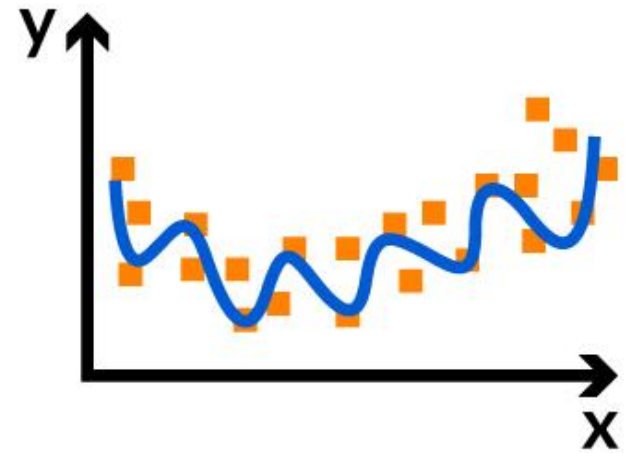
RECORDATORIO



Underfitting



Balanceado



Overfitting

Fuente: Docs de Amazon sobre Machine Learning



¿Qué podemos hacer?

- Técnicas de feature Selection y Feature Extraction (PCA).
- Regularización.
- Balanceo: Oversampling y Undersampling.



Introducción a PCA



¿Qué son los features?

- Atributos de nuestro modelo que usamos para realizar una inferencia o predicción.
- Variables de entrada.



¿Problema?

En machine learning es normal encontrarnos con problemas donde tengamos una enorme cantidad de features relaciones complejas entre ellos y con la variable que queremos predecir.



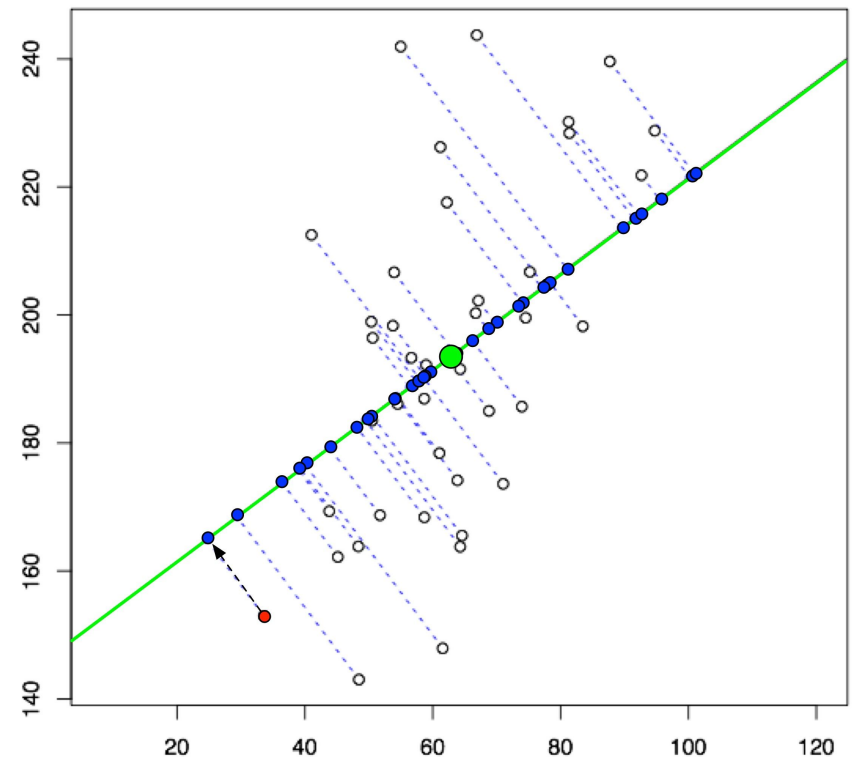
¿Problema?

1. Nuestro dataset tiene un número alto de features.
2. Hay una alta correlación entre los features.
3. Overfitting.
4. Alto coste computacional.

Solución: PCA

Reducir la complejidad del problema:

1. Seleccionando solamente las variables relevantes.
2. Combinandolas en nuevas variables que mantengan la información más importante. (Varianza de los features)



Proyección de los puntos sobre un plano que mantiene su varianza.

Análisis de Componentes Principales

1. Calculamos la matriz de Covarianza para expresar las relaciones entre nuestros features.
2. Hallamos los vectores propios y valores propios de esta matriz, para medir la fuerza y variabilidad de estas relaciones.
3. Ordenamos y escogemos los vectores propios con mayor variabilidad, esto es, que aportan más información.

Adicionalmente...

Si tenemos un dataset demasiado exigente o un equipo de bajos recursos, podemos usar IPCA.

Si nuestros datos no tienen una estructura separable linealmente, y encontramos un KERNEL que pueda mapearlos podemos usar KPCA.

Implementación de PCA e IPCA.

Código

En nuestro código

1. Escalamos con *StandardScaler()*
2. Declaramos un objeto *PCA()*
3. Identificamos componentes *pca.explained_variance_ratio_*
4. Extraemos componentes *pca.components_*

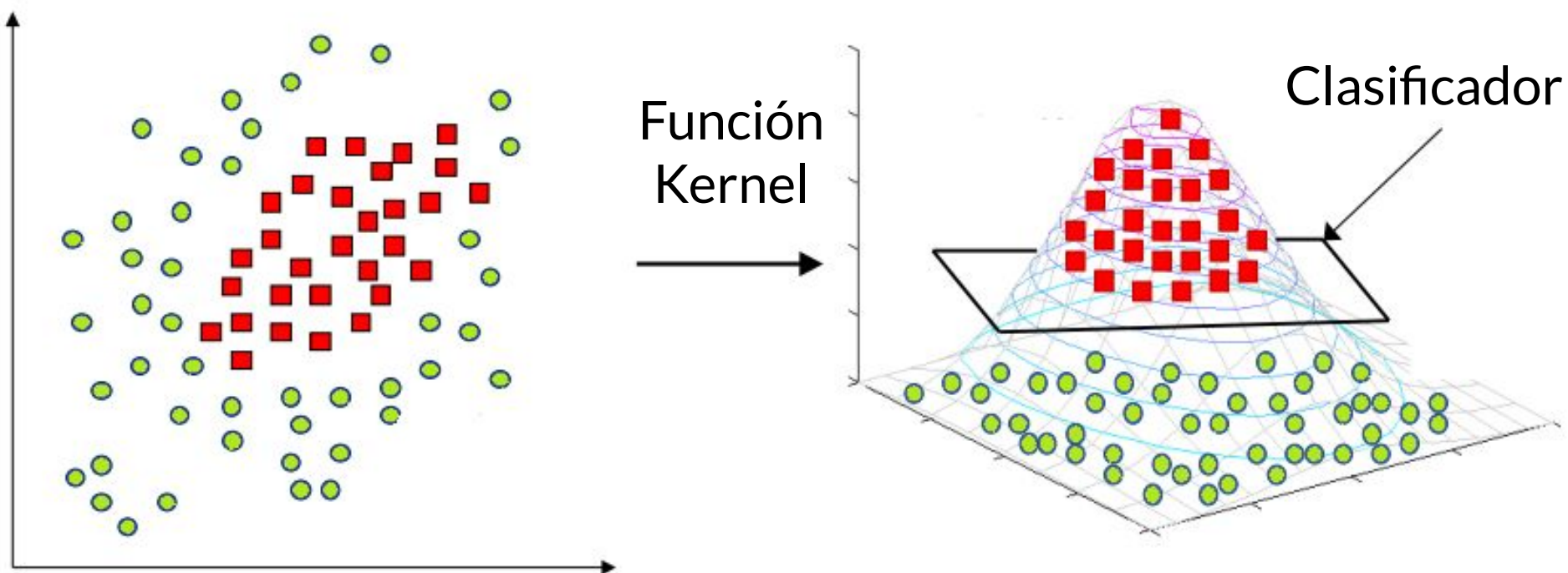
Funciones de Kernel y KPCA



Kernels

Un Kernel es una función matemática que toma mediciones que se comportan de manera no lineal y las proyecta en un espacio dimensional más grande donde sean linealmente separables.

Kernels



Kernels comunes

Lineales

$$k(x, y) = x * y$$

Polinomiales

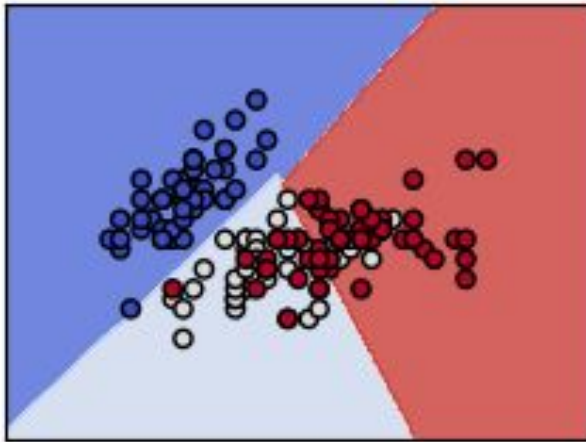
$$k(x, y) = (x * y)^p$$

Gaussianos
(RBF)

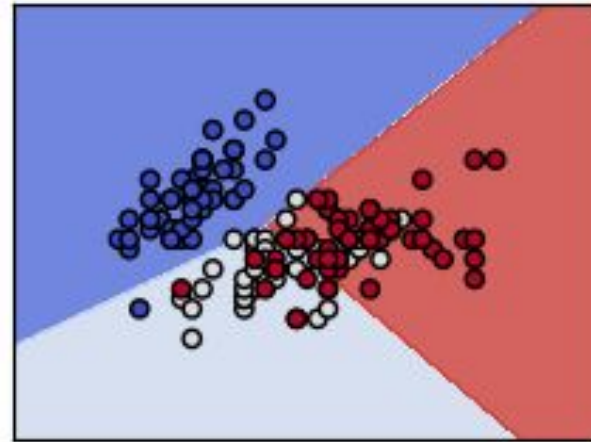
$$k(x, y) = e^{-\frac{\|x - y\|^2}{2\sigma^2}}$$

Clasificador con diferentes Kernels

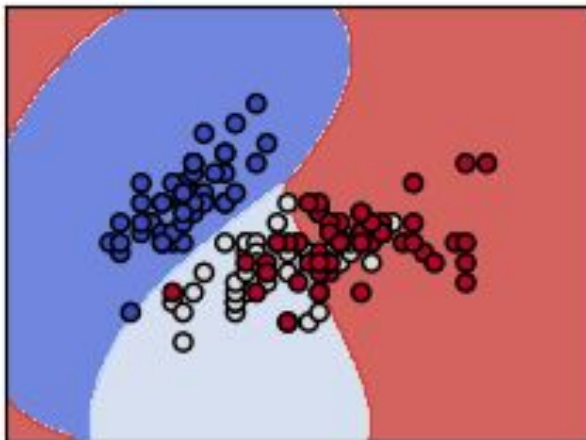
SVC: Kernel Lineal



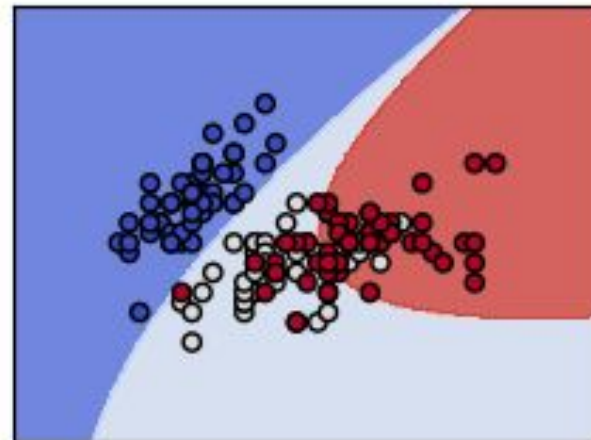
LinearSVC: Kernel Lineal



SVC: Kernel RBF



SVC: Kernel Polinomial



Implementación de KPCA

Código

¿Qué es la Regularización?



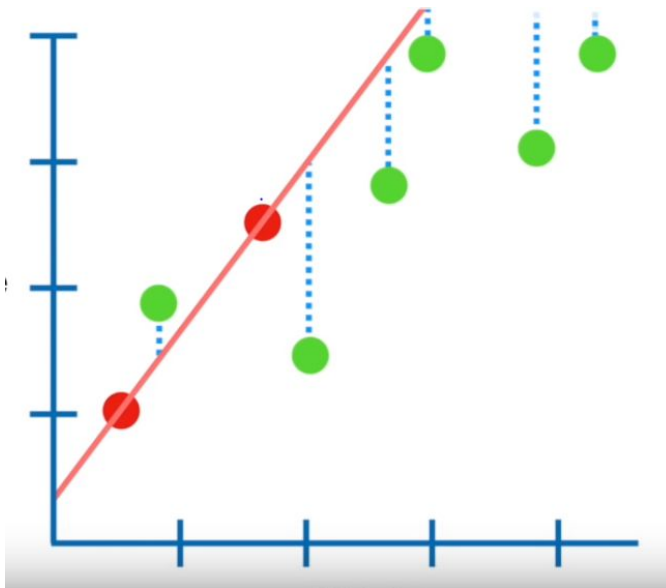
Regularización

La regularización consiste en disminuir la complejidad del modelo a través de una penalización aplicada a sus variables más irrelevantes.

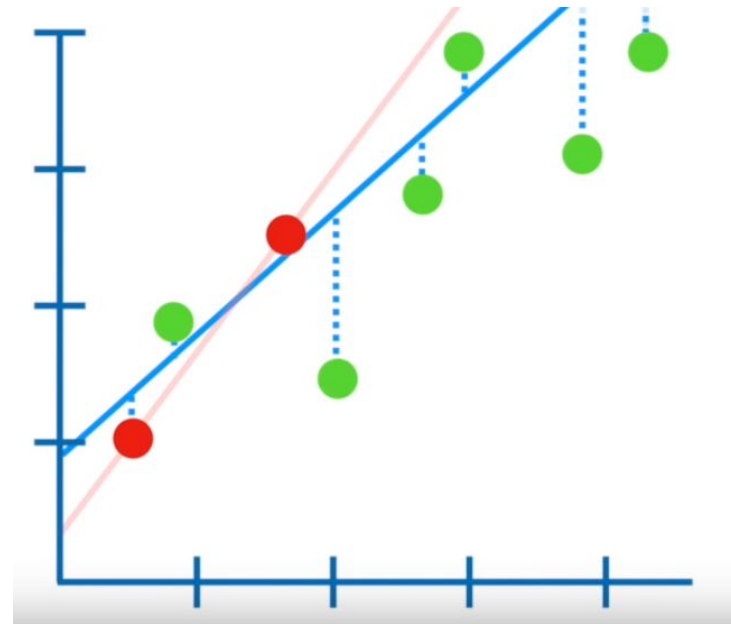
Cambiar: +Sesgo por -Varianza

Entrenamiento

Test

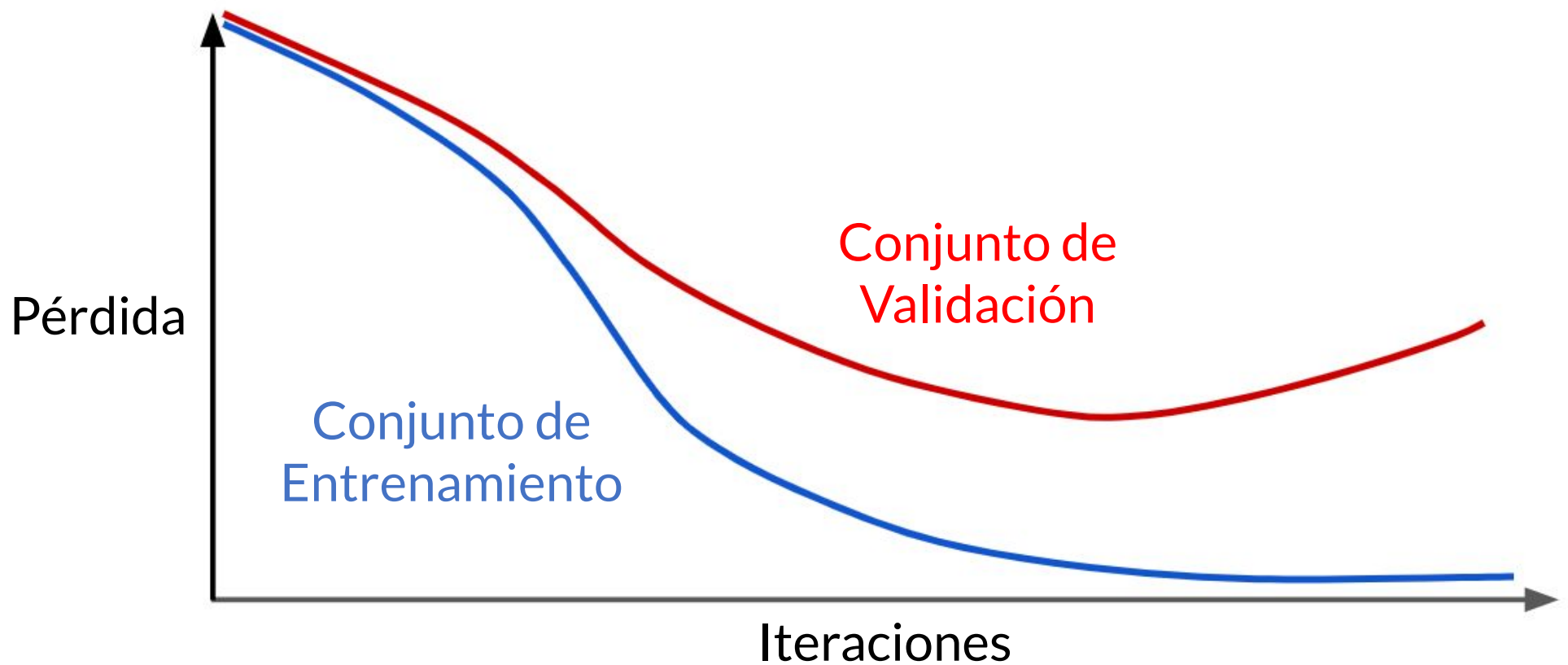


Overfitting en
regresión simple



Se introduce bias para
mejorar la generalización

Pérdida en entrenamiento y en validación



Fuente: Crash course Aprendizaje automático de Google



Regularización

- **L1 Lasso:** Reducir la complejidad a través de la eliminación de features que no aportan demasiado al modelo.
- **L2 Ridge:** Reducir la complejidad disminuyendo el impacto de ciertos features a nuestro modelo.
- **ElasticNet:** Es una combinación de las dos anteriores.

Regularización Lasso

También llamada L1

- Penaliza a los features que aportan poca información ***volviéndolos cero***, eliminando el ruido que producen en el modelo.

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

Penalización

Regularización Ridge

También llamada L2

- Penaliza los features poco relevantes, pero no los vuelve cero. Solamente limita la información que aportan a nuestro modelo.

$$\hat{\beta} = \arg \min_{\beta} \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Penalización



Lasso vs Ridge

1. No hay un campeón definitivo para todos los problemas.
2. Si hay pocos features que se relacionen directamente con la variable a predecir: **Probar Lasso.**
3. Si hay varios features relacionados con la variable a predecir. **Probar Ridge.**

Implementación de Lasso y Ridge

Código

El problema de los valores atípicos

¿Qué es un valor atípico?

- Un valor atípico es cualquier medición que se encuentre por fuera del comportamiento general de una muestra de datos.
- Pueden indicar variabilidad, errores de medición, o novedades.



¿Por qué son problemáticos?

1. Pueden generar sesgos importantes en los modelos de Machine Learning.
2. A veces contienen información relevante sobre la naturaleza de los datos.
3. Detección temprana de fallos.

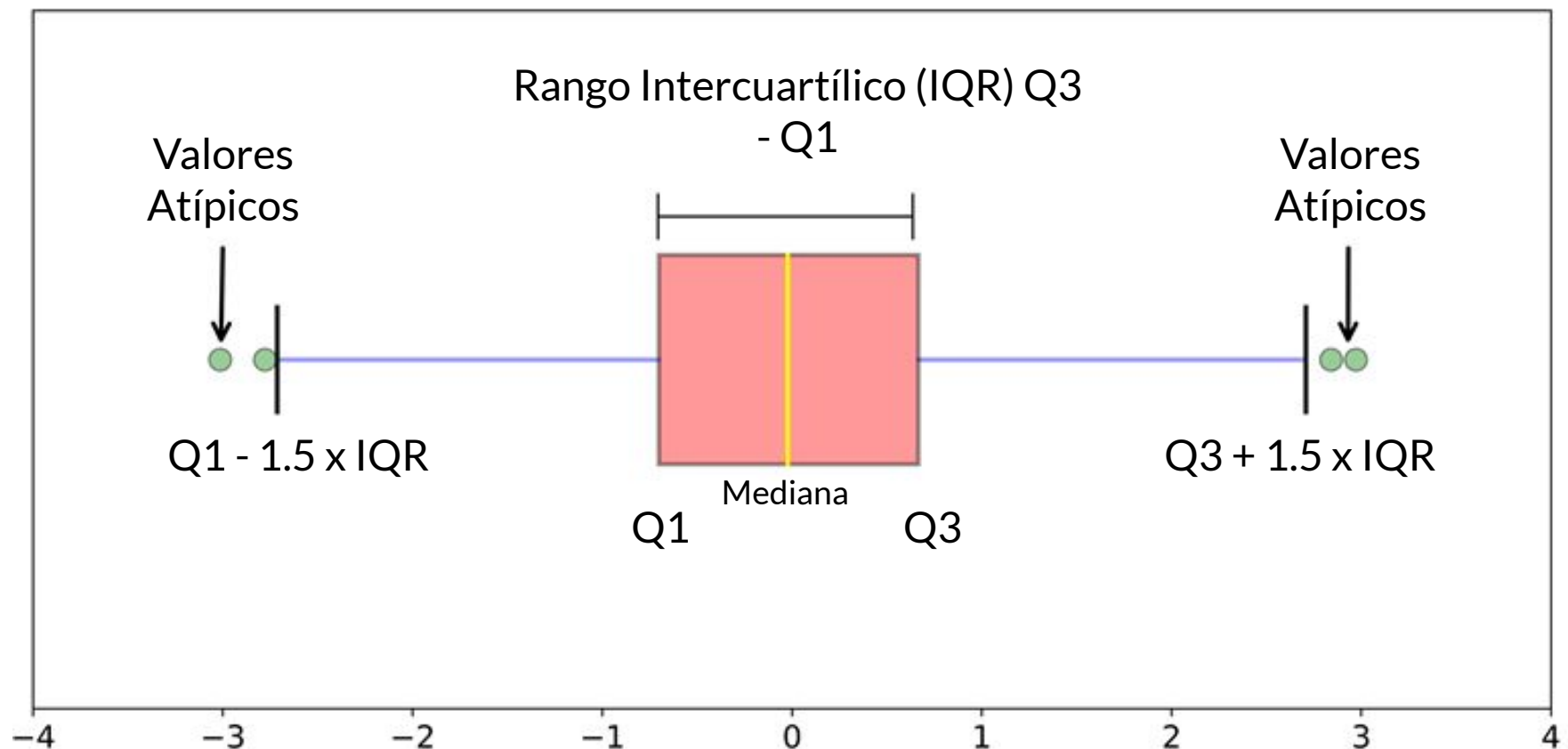
¿Cómo identificarlos?

A través de métodos estadísticos:

1. **Z – Score:** Mide la distancia (en desviaciones estándar) de un punto dado a la media.
2. Técnicas de clustering como **DBSCAN**.
3. Si $q < Q1 - 1.5 * IQR$ ó $q > Q3 + 1.5 * IQR$

¿Cómo identificarlos?

Gráficamente a través de Boxplots



Regresiones Robustas en Scikit-learn



Regresiones Robustas

Sci-kit learn nos ofrece algunos modelos específicos para abordar el problema de los valores atípicos:

1. RANSAC
2. Huber Regressor



RANSAC

Random Sample Consensus

Usamos una muestra aleatoria sobre el conjunto de datos que tenemos, buscamos la muestra que más datos “buenos” logre incluir.

El modelo asume que los “malos valores” no tienen patrones específicos.

Huber Reggresor

No ignora los valores atípicos, disminuye su influencia en el modelo.

Los datos son tratados como atípicos si el error absoluto de nuestra pérdida está por encima de un umbral llamado *epsilon*.

Se ha demostrado que un valor de *epsilon* = 1.35 logra un 95% de eficiencia estadística.

Implementación de regresores robustos

Código



Métodos de ensamble



Métodos de ensamble

1. Combinar diferentes métodos de ML con diferentes configuraciones y aplicar un método para lograr un consenso.
2. La diversidad es una muy buena opción.
3. Los métodos de ensamble se han destacado por ganar muchas competencias de ML.

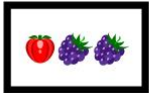
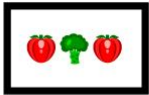


Dos estrategias: Bagging

¿Qué tal si en lugar de depender de la opinión de un solo “experto” consultamos la opinión de varios expertos en paralelo e intentamos lograr un consenso?

Dos estrategias: Bagging

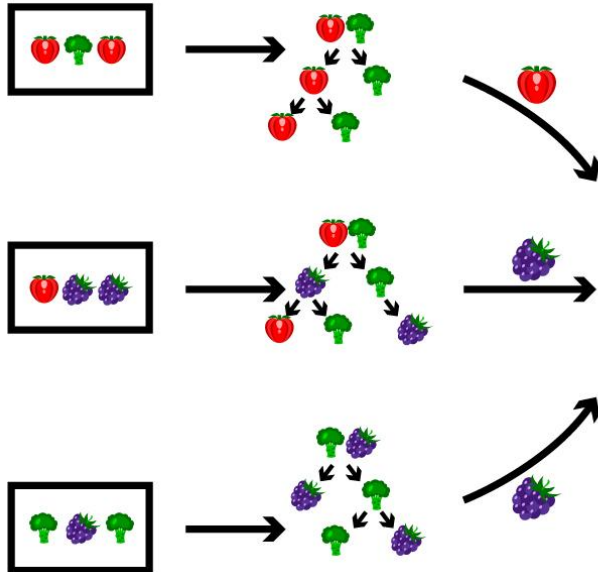
Bagging viene de : *Bootstrap AGG*gregation.



- Creamos particiones aleatorias (uniforme y con reemplazo) del conjunto de datos original.

Dos estrategias: Bagging

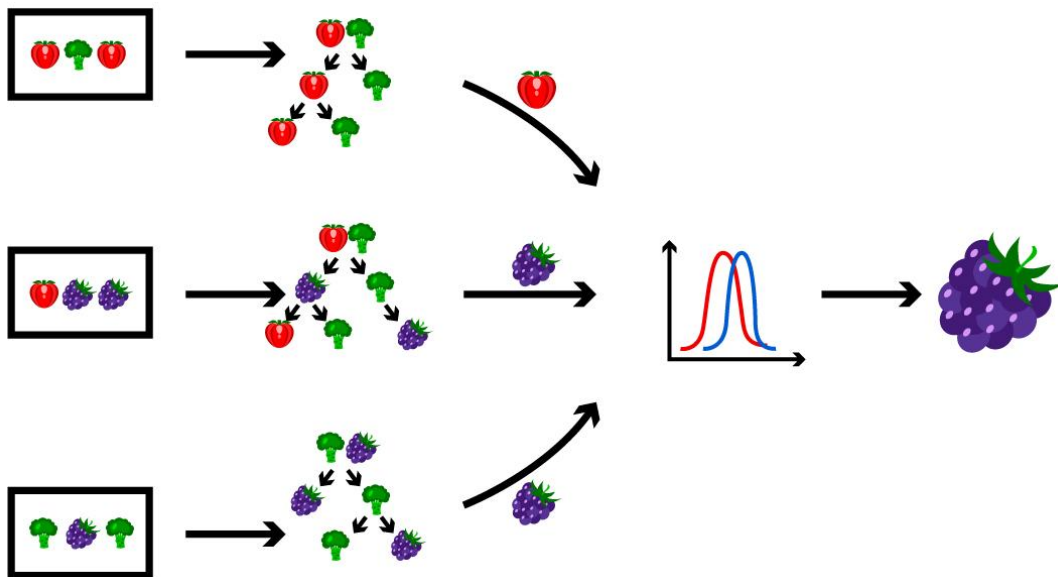
Bagging viene de : *Bootstrap AGG*gregation.



- Se construyen modelos de aprendizaje para cada uno de los subconjuntos por aparte.

Dos estrategias: Bagging

Bagging viene de : *Bootstrap AGG*gregation.



- La respuesta final es la combinación (Por ejemplo por votación) entre las respuestas individuales.

Dos estrategias: Bagging

Modelos Ensamblados basados en Bagging

1. Random Forest.
2. Voting Classifiers/Regressors.
3. En general se puede aplicar sobre cualquier familia de modelos de Machine Learning.

Dos estrategias: Boosting

¿Y si probamos otro enfoque?

Le pediremos a un experto su criterio sobre un problema. Medimos su posible error, y luego usando ese error calculado le pedimos a otro experto su juicio sobre el mismo problema.

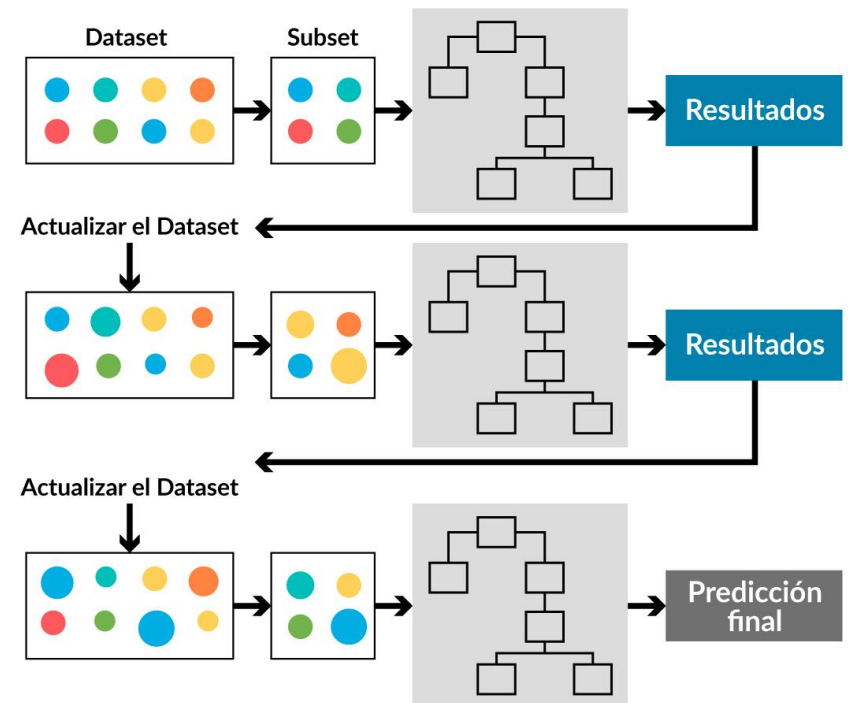
Dos estrategias: Boosting

Boosting: Impulsar / Propulsar

Boosting es un método secuencial

Busca fortalecer gradualmente un modelo de aprendizaje usando siempre el error residual de las etapas anteriores.

El resultado final también se consigue por consenso entre todos los modelos.



Dos estrategias: Boosting

Modelos Ensamblados basados en Boosting

1. AdaBoost.
2. Gradient Tree Boosting.
3. XGBoost.

Implementación de Bagging

Código

Implementación de Boosting

Código

Estrategias de Clustering

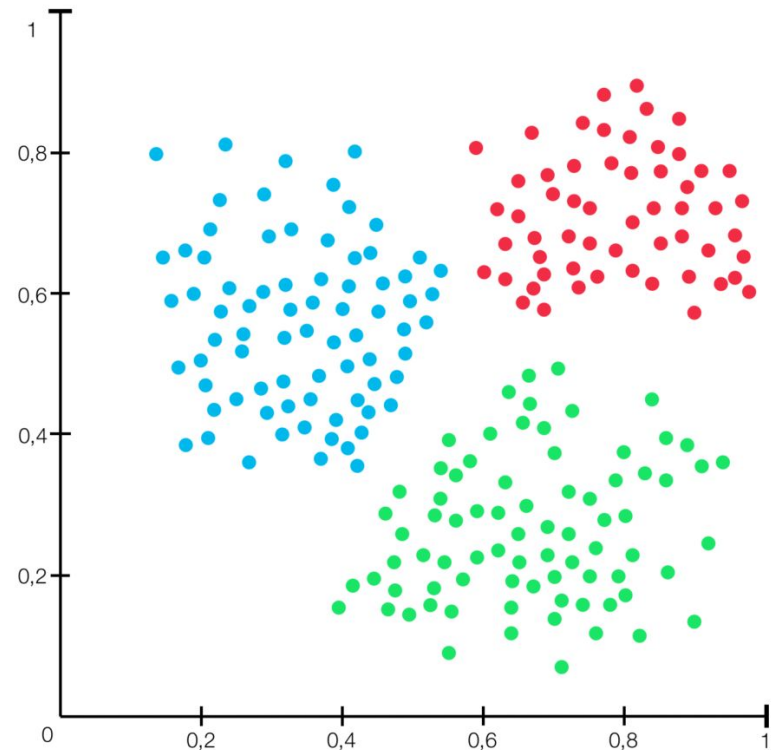


Clustering

Los algoritmos de clustering son las estrategias que podemos usar para agrupar los datos de tal manera que todos los datos pertenecientes a un grupo sean lo más similares que sea posible entre sí, y lo más diferentes a los de otros grupos.

El problema del Clustering

1. No conocemos con anterioridad las etiquetas de nuestros datos (Aprendizaje no supervisado).
2. Queremos descubrir patrones ocultos a simple vista.
3. Queremos identificar datos atípicos.





Dos casos de aplicación

1. Cuando sabemos cuántos grupos “ k ” queremos al final en nuestro resultado.
2. Cuando queremos que el algoritmo descubra la cantidad de grupos “ k ” óptima según los datos que tenemos.



Clustering

Si conocemos k ,

Usaremos ***k -means***, o bien, ***Spectral Clustering***.



Clustering

Si no conocemos k ,

Usaremos *Meanshift*, *Clustering jerárquico*,
o bien *DBScan*.

Implementación de Batch k-means

Código

Implementación de Mean-Shift

Código

Validación de nuestro modelo usando Cross Validation



Validación de modelos

1. La última palabra la tienen los datos.
2. Necesitamos mentalidad de testeo.
3. Todos los modelos son malos, solamente que algunos resultan útiles.



Tipos de validación

1. Dividir nuestros datos en Entrenamiento / Prueba (Hold-On).
2. Usar Validación Cruzada (K-Folds).
3. Validación Cruzada (LOOCV).



HOLD-ON

1. Se requiere un prototipado rápido.
2. No se tiene mucho conocimiento.
3. No se cuenta con abundante poder de cómputo.

K-FOLDS CV

1. Recomendable en la mayoría de los casos.
2. Se cuenta con un equipo suficiente para desarrollar ML.
3. Se requiere la integración con técnicas de optimización paramétrica.
4. Se tiene más tiempo para las pruebas.





LOOCV

1. Se tiene gran poder de cómputo.
2. Se cuentan con pocos datos como para dividir por Training / Test.
3. Personas que sufren de TOC y quieren probar todos los casos posibles.

Implementación de K-Folds Cross Validation

Código



Optimización Paramétrica con CV

Problema

¡Finalmente
encontramos un
modelo de aprendizaje
de máquina que
parece funcionar!



Problema

Ahora debemos enfrentarnos a la optimización de este modelo para descubrir sus mejores parámetros y subir hasta el cielo nuestras métricas.



Optimización paramétrica

1. Es fácil perderse entre los conceptos de tantos parámetros.
2. Es difícil medir la sensibilidad de los mismos manualmente.
3. **ES COSTOSO.**

```
SVC(C=1.0, kernel='rbf', degree=3, gamma='auto  
_deprecated', coef0=0.0, shrinking=True, probabilit  
y=False, tol=0.001, cache_size=200, class_weight=  
None, verbose=False, max_iter=  
1, decision_function_shape='ovr', random_state=N  
one)
```




Tres enfoques

1. Optimización manual.
2. Optimización por grilla de parámetros.
GridSearchCV.
3. Optimización por búsqueda Aleatorizada.
RandomizedSearchCV.



Búsqueda Manual

1. Escoger el modelo que queremos ajustar.
2. Buscar en la documentación de scikit-learn.
3. Identificar los posibles ajustes.
4. Probar combinaciones una por una iterando a través de listas.




Búsqueda por grilla

1. Definir una o varias métricas que queramos optimizar.
2. Identificar los posibles valores que pueden tener los parámetros.
3. Crear un diccionario de parámetros.
4. Usar Cross Validation.
5. ¡Entrenar el modelo e ir por un café!

Búsqueda por grilla

La grilla de parámetros nos define GRUPOS DE PARÁMETROS que serán probados en todas sus combinaciones (Un grupo a la vez)

```
param_grid =  
[  
    {  
        'C':[1,10,100], 'kernel':['linear','rbf']},  
    {  
        'C':[1,10,100,1000], 'gamma':[0.001,0.0001], 'kernel':['rbf']  
    },  
]
```



SVC (C=1, 'kernel' = 'linear')
SVC (C=1, 'kernel' = 'linear')
SVC (C=1, 'kernel' = 'linear')
SVC (C=1, 'kernel' = 'rbf')
SVC (C=1, 'kernel' = 'rbf')
SVC (C=1, 'kernel' = 'rbf')



Búsqueda aleatorizada

1. Definir una o varias métricas que queramos optimizar.
2. Identificar los rangos de valores que pueden tomar ciertos parámetros.
3. Crear un diccionario de rangos de valores.
4. ¡Usar Cross Validation!
5. ¡Entrenar el modelo e ir por un café!

Búsqueda aleatorizada

En este método, definimos escalas de valores para cada uno de los parámetros seleccionados, el sistema probará varias iteraciones (Configurables según los recursos) y mostrará la mejor combinación encontrada.

```
parametros = {  
    'C': scipy.stats.expon(scale=100),  
    'gamma': scipy.stats.expon(scale=1),  
    'kernel': ['rbf', 'linear'],  
    'class_weight': ['balanced', None],  
}
```



```
SVC (C=1, gamma=0.001, 'kernel'='linear',  
class_weight='balanced')  
  
SVC (C=5, gamma=0.3, 'kernel'='rbf',  
class_weight=None)
```



¿Cuando usar cada cuál?

GridSearchCV

Cuando se quiera realizar un estudio a fondo sobre las implicaciones de los parámetros, y además se tenga el tiempo y el poder de procesamiento requerido.

Randomized SearchCV

Cuando se quieran explorar posibles optimizaciones, cuando haya poco tiempo o poco poder de procesamiento.

Implementación GridSearchCV y RandomizedCV

Código

Revisión de nuestra arquitectura de código

Código

Exportar e importar modelos con Scikit-Learn

Código

Creación de una API REST con FLASK

Código