**2020/03/13**

**PYTHON APRENDIZAJE SUPERVISADO**

**Toni Almagro**

También vamos a ver si da tiempo algo de data engineering.

**Tipos de Machine Learning:**

**Reinforcement Learning**: coches autónomos, juegos. No se hablará en el máster.

**Unsupervised Learning**:

* Clustering: se trata de que el modelo agrupe nuestros individuos, que les añada etiquetas. K-means típico modelo.
* Dimensionality reduction: no suele ir aisladamente. Es una herramienta que se usa para luego meterla en otro sitio. Si tengo por ejemplo 100 variables en mis datos, quiero reducirlas a 20

**Supervised Learning:** cuando tenemos una etiqueta en nuestros valores.

***Regression***: La etiqueta objetivo puede ser una variable continua (un número)

***Classification***: variable categórica o discreta .

Veremos:

* Features.
* Models,
* Metrics: sirven para comparar y evaluar el modelo.

No existe el mejor modelo, sino el mejor modelo para tu métrica o para tu objetivo

* Model interpretability.
* Salvar modelos: con pickle.

Aprendizaje supervisazo requiere un vector de salidad y otro de entrada (input y output)

El input seerá una matrix porque lo habitual es tener para cada item varias variables.

Nuestro modelo será la relación entre el input y el output.

Si la relación es un número, el modelo será un regresor.

Si la relación es una clase, el modelo será un clasificador.

**Qué tipos de relaciones hay?**

Regression:

* Regresion lineal
* k neighbour
* decision tree

Classification

* Logistic Regression
* k neighbour
* Support vector machine
* Decision Tree Classifier

Ojo, decision tree se puede usar en regresion y en clasificación.

**Métricas:**

Regression:

* RMSE
* MAE (Mean Absolute Error) and MAPE
* Correlation and Bias

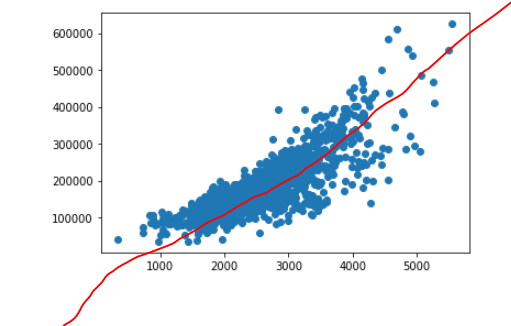
Classifcation:

* Accuracy
* Precision and Recall: número de acierto de acuerdo a algo
* AUC Curve

Lo importante es quedarnos que las métricas usadas para regresión y clasificación son diferentes.

**Regression**

Ejemplo: predecir el precio de una casa a partir de su superficie.



Con una regresion lineal el modelo falla para casa pequeñas, porque salen precios negativos.

Puedo hacer varias cosas:

* Decir que a partir de una punto los precios son una constante que yo defino: CUTRE.
* Partir los datos en 2 y crear un modelo para csas pequeñas y otro modelo para el resto de casa.
* Aumento la complejidad del modelo: hacer una regresión que en vez de una recta sea una parábola.

sklearn tienen una librería que nos hace el training set y el test set automáticamente.



**MAE**

Sus unidades son las mismas que la variable objetivo. En nuestro ejemplo de las casas sería dólares. Es mmuy fácil intuirlo pero pierdo la información relativa del error. Es decir, si el error es grande o pequeño.

En sklearn

**MAPE**

Es como el MAE pero divide por yi para convertirlo en un porcentaje, en algo relativo.

**K Nearest Neighbors**

Variables que necesitamos:

* k = número de vecinos
* Weight: que ponderen los puntos más cercanos al valor correcto.
  + Uniform: da el mismo valor a todos
  + Distance: da mas importancia cuando más cerca está el punto de la variable
  + Custom: meto los pesos como quiera.

NOTA: cuando tengo más de una variable de entrada (que es siempre) debemos normalizar las variables para que las distancias tenga una escala comparable de cálculo

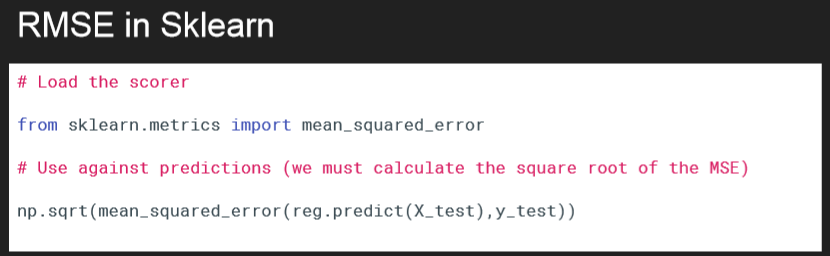
**RMSE**

Es la raíz cuadrada del MSE cuyas unidades son las de la variable objetivo (dolares) al cuadrado. Hacemos la raíz cuadrada y conseguimos que las unidades sean dólares, y podemos compararlo con el MAE.

El cuadrado de su fórmula minimiza mucho los errores < 1 y maximiza mucho los > 1.

*Significa que si tengo un MAE normalito y un RMSE más grande, es que hay poquitos puntos que se han ido mucho de la predicción.*

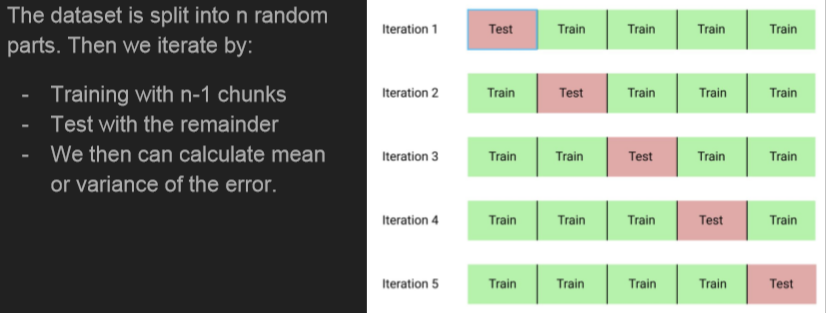
MORALEJA: el MAE y el RMSE **son magnitudes diferentes** y debemos calcular los dos para saber leer mejor nuestro modelo.



**CROSS VALIDATION**

Cuando sólo hago un split del data set mi métrica puede variar bastante dependiendo de cómo haga el split. Para conseguir métricas más robustas hacemos cross validation.

Dividimos el dataset en n trozos y hacemos el split n veces.



Porque sacaríamos una métrica para cada iteración y usaríamos la media.

Cross Validation NO ES UNA METRICA. Es una forma de obtener unas métricas más robustas.

Con sklearn y un par de calculos podemos sacar métricas calculadas con cross validation automáticamente

**Decision Tree**

Lo que hace es construir particiones homogenes. Particionar tus datos de manera que sean más homogéneos que antes.

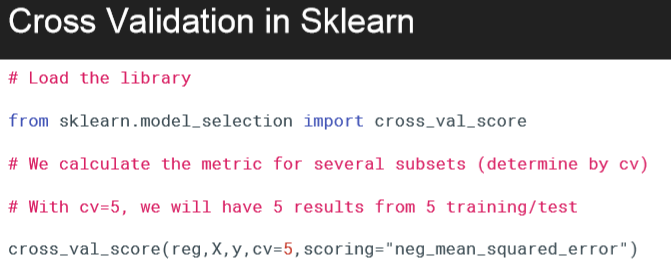
Cómo se mida la homogeneidad? Varianza (regresión) o entropía (clasificación.

Parámetros a definir:

* Max\_depth: Número de particiones.
* Min\_sample\_leaf: Minimo número de samples para que exista una partición

El decission tree hace como cajoncitos con valores. Cuando quiero predecir un valor, el decision tree sabe a qué cajón asignarlo

El arbol de decision parte el espacio en partes y asigna valores constantes por trozos



**GridSearchCV**

Cómo sabemos en k-vecinos cuál es el valor de K óptimo? Por prueba y error endría que probar con 5, 6, 7, hasta… por ejemplo 100.

Y luego

**RandomizeSearchCV**

Si quiero probar una malla muy grande y creo que me va a petar el ordena puedo pedirle que elija de manera aleatoria unos puntos de mi malla.

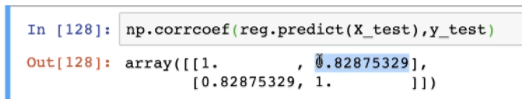
**2020/03/14**

**Correlación (otra métrica)**

Cómo de conectados están unos puntos de otros.

Como métrica no está definida en sklearn.

Pero sí lo están en numpy. corrcoef te devuelve la matriz de correlación

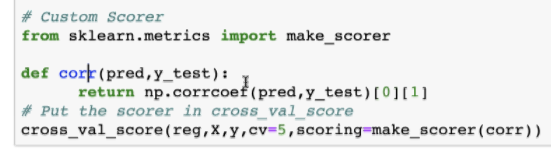


Cómo nos definimos esta métrica en sklearn?

Nos creamos una función:

* Cuyos inputs sean dos vectores, los valores predichosy ylos reales
* El output sea un escalar.

Usamos la función creada en cross\_val\_score:

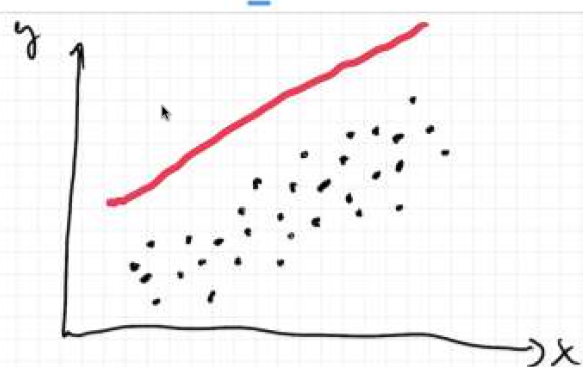


Cuanto más correlación haya entre mi predicción y me valor real mejor.

**Bias** (otra métrica)

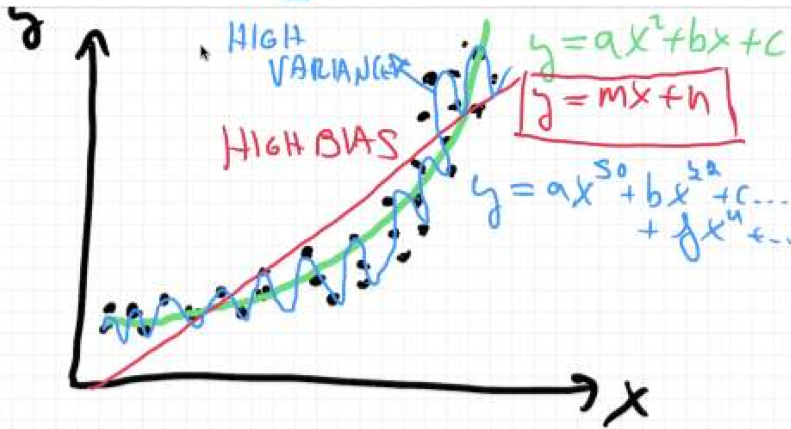
Mide la diferencia entre mi predicción y el valor real, sin corregir si la diferencia es negativa o positiva. De manera que los errores negativos con los positivos se van a corregir unos con otros. Otra manera de verlo es que es la media de los errores.

Ejemplo de tener una correlación alta y un bias alto:



El bias de arriba sale alto porque todos los errores tienen el mismo signo.

En general cuando tengo un high bias significa que no estoy capturando la complejidad del problema.



**Variance**: otra métrica. Es lo que varían los datos. El modelo azul tienen high variance.

Cómo reducimos un problema de high variance? Reduciendo la complejidad del modelo

Cómo reducirmos un problema de high bias? Aumentando la complejidad del modelo.

Cosas que pueden meter ruido en nuestro datos:

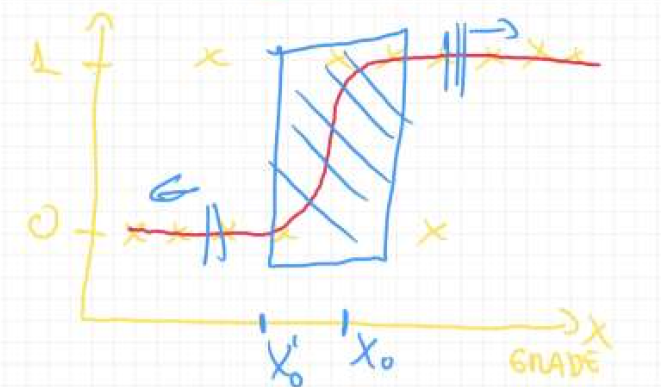
* Factores aleatorios que desconocemos
* Factores no aleatorios que no conocemos.

Overfitting significa que hemos modelado el ruido

El bias y la varianza se usan para controlar el tema del overfitting

**Classification**

**Logistic Regression**



Entendemos la curva sigmoide como la probabilidad de que sea 1 o 0.

Es un modelo muy sencillo.

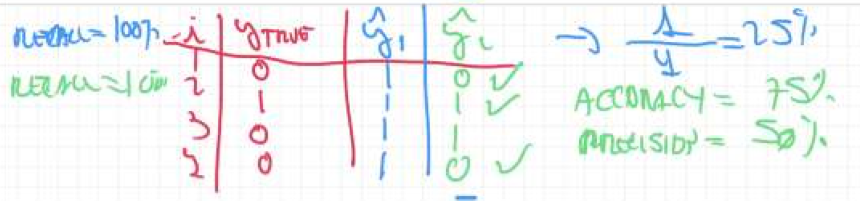
No hay que ajustar ningún parámetro.

**Precision vs Recall (metrica)**

**Precisión**: a la hora de predecir 1 o 0. Nos da igual medir los 1 o los 0. Vamos a usar pa precisión en la predicción de los 1 (por ejemplo). Por defecto siempre nos centramos en el 1.

NOTA: el accuracy mide todos los aciertos (unos y ceros). La precisión mide los aciertos de sólo unos o sólo ceros!

**Recall**: cuantos 1 de los datos reales he conseguido acertar.



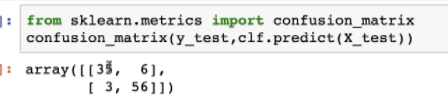
Cuál es mejor? no existe una mejor que otra. Depende de lo que quiera medir, etc.

En modelos médicos es mejor usar el recall. Ejemplo: 1 es que tengo peritonitis. Si predigo que lo tiene y no lo tiene no pasa nada, pero si predigo que no lo tiene, y SI lo tienen y le mando a casa se puede morir.

El **recall** es importante cuando el NO encontrar los positivos tiene un precio muy grande

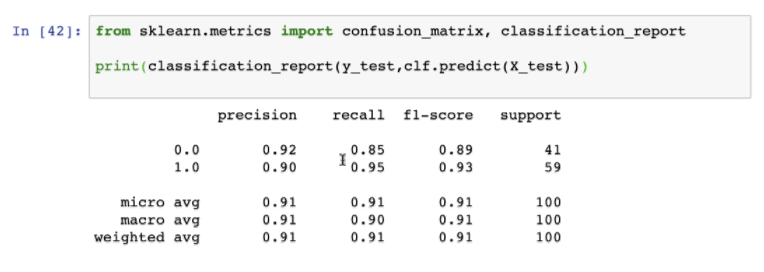
Precision y Recall están en sklearn

La **matriz de confusión** también está en sklearn



No es una métrica per se pero ayuda a ver cómo está repartido tu problema

**Classification report**



nos dice cómo ha funcionado este modelo de clasificación.

fl-score es la media de precision y recall

support: son los puntos que realmente avalan nuestra hipótesis, cuántos son unos y cuántos son 0. En este ejemplo tengo 41 cerosy 59 unos.

Las 3 lineas de abajo nos dan igual.

**Support Vector Machine**

No lo usaremos a menudo porque tarde muchísimo en entrenarse.

Lo usaríamos cuando el separador de las clases sea circular u otra forma compleja, ya que los modelos anteriores no me permiten esto.

No aporta en general gran cosa, teniendo otros modelos.

Intenta buscar una línea que separe las dos clases.

Se llama support porque parece que se está apoyando en la clase de abajo

**Parámetros**:

El parámetro con el que se juega se llama MARGIN: es un error, dice hasta dónde estamos dispuestos a permitir que las clases estén mezcladas.

C: sum of Error Margin

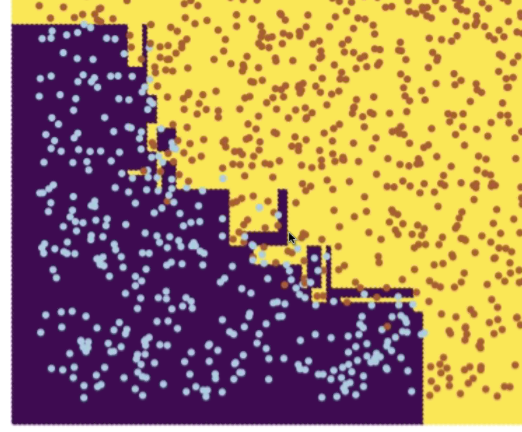
Kernel: es el tipo de separación

* lineal
* circle
* hay más…

**Decision Tree Classifier**

Medimos la entropía en lugar de la varianza, para buscar la homogeneidad.

La división que hace en los datos también es escalonada.



Cualquier modelo de clasificación por debajo lo que hace en realidad es calcular la probabilidad de que sea 0 o 1. Y él mismo me devuelve 1 si está por encima del 50% y 0 si está por debajo.

¿Qué pasa si yo altero ese threshold del 50%? Que cambian mis predicciones.

Puedo ir cambiando thresholds n veces, obteniendo diferentes predicciones y comparandolas con los datos reales. Y si pinto eso me sale la curva ROC!!

Esa curva nos da la confianza que teng aun clasificador en sí mismo. Que tenga mucha confianza en sí mismo significa que cuando predice unos las porbs son 0.01, 0.3,,, valores bajos. Y con ceros valores muy altos 0.9, 0.85… pero no encuentro 0.4, 0.6, etc.

AUC: Para medir esa confianza uso el área bajo la curva. Cuanto más cerca de 1 sea el área mejor.

Un vez hemos probado todos los modelos si queremos seguir mejorando qué hacemos???

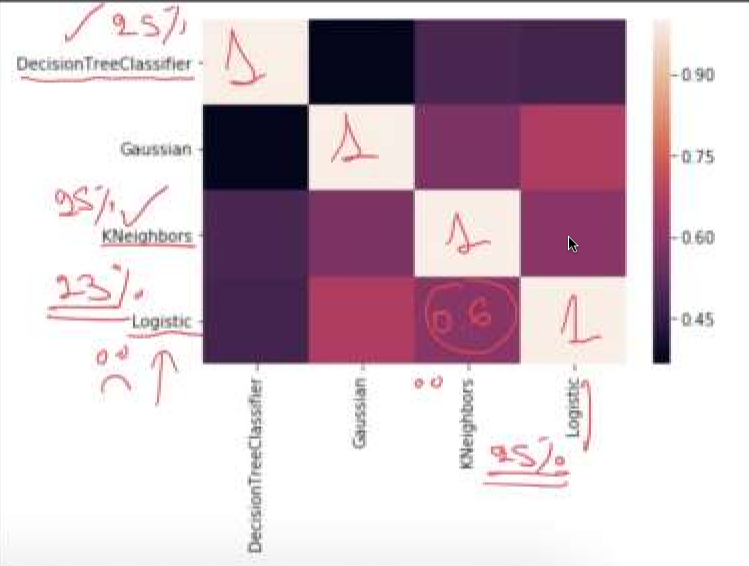
**Ensemble Learning**

combinar varios modelos sencillos en un intento de mejorar nuestro resultado.

Se puede usar tanto en regresión como en clasificación.

Por qué me planteo hacer ensemble learning?

Si dos modelos tienen muy buenas métricas pero sus respuestas no tienen buena correlación entre sí significa que donde un modleo lo hace muy bien el otro lo hace mal y viceversa.



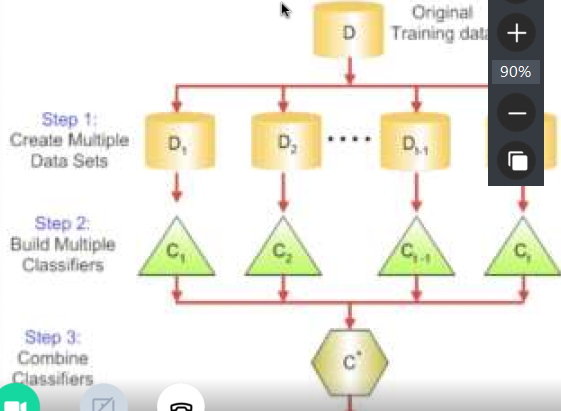
**Tipos de combinación:**

**Sequential**: creas un modelo, sacas sus residuos, creas un modelo corrigiendo esos errores y así varias veces.

* **Stacking**: meter las predicciones de un modelo en otro modelo nuevo.

**Parallel**: se crean varios modelos, se comparan los resultados y se elige:

* **Voting**/Averaging: el resultado que tenga mayoría. Da resultados bastante malos!
* O se ponderan los resultados de un modelo en concreto.
* **Bagging**: ni secuencial ni parallel.
  + Hago diferentes bolsitas / particiones de mi data set, con repetición o no, de forma heterogénea. N de bolsitas es el **número de estimadores, empezar siempre con 100.** Si mejora el modelo simple (sin ensemble) probamos con más, 500 o 1000.
  + Creo un modelo SENCILLO por cada bolsita. Puedo mezclar tipos de modelos pero en la práctica lo normal es que sea el mismo tipo de modelo. Hay un bagging muy típico que es hacerlo con Decision Trees: a esto se le llama RANDOM FOREST.
  + Combino el resultado de cada modelo.

–

Random Forest : se da por explicado….

**Gradient Boosted Trees** (stacking)

Es un ensemble learning

Se usa la función **GradientBoostingRegressor** de sklearn

Con el notebook de House Pricing

La primera predicción (o modelo) va a aser siempre la media.

Construyo un data frame nuevo con dos columnas: el valor real y la predicción.

Calculo el residuo (valor real menos predicción) y lo guardo en una tercera columna.

Entreamos un modelo para encontrar la relación entre los residuos y el valor real.

La variable alpha= 0.1 es el **learning rate (**con cuánto me quedo de la corrección del modelo anterior**).** Hay que empezar con un valor pequeñito para que el valor no se vuelva inestable.

El experto 2 dice que su modelo es el modelo 1 más el modelo del residuo.

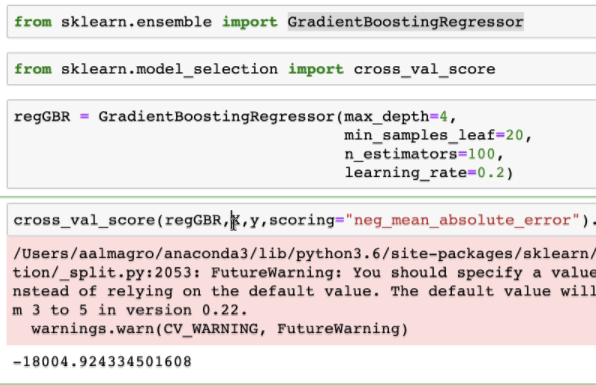
El experto 3 calcula los residuos entre el valor real y los residuos 2. y crea otro modelo entrenando con los residuos 2

Esto sería un gradient boosted tree con 2 estimadores.

Todo esto ha sido para comprenderlo, en la práctica va a ser una caja negra.

Lo parámetros que necesitamos decidir son:

* Propios del decision tree:
  + Max depth
  + Min sample leaf
* Propios del ensemble:
  + Num de estimadores
  + Learning Rate: con cuánto me quedo de la corrección del modelo anterior. Cómo de rápido aprender. Si vas muy rápido la cagas.



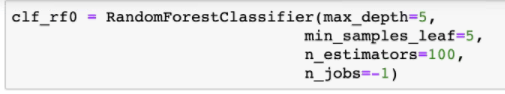
El cómo se combinan los modelos, cuales se eligen y tal es una caja negra, no lo vamos a ver.

**Random Forest**

Parametros:

Los mismos que el decision tree

Y además el número de estimadores.



Si no sé qué parametros poner, como siempre, puedo hacer un GridSearchCV



Cómo compartir un modelo con otra persona?

**Pickle**

Guarda cualquier arhcivo binario de python y también pipelines.

Open() para guardar un archivo

La extensión que me de la gana (pongo pickle)

wb = write no sé qué

pickle.load() para abrir el archivo que me hanmandado

rb = read no sé qué