Aprendizaje Automático, Regresores y Clasificadores

Inteligencia Artificial e Ingeniería del Conocimiento

Constantino Antonio García Martínez

Universidad San Pablo Ceu

Bibliografía

PMLR Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

AIMA Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Pearson, 2016.

Regresión Lineal, Regresión Logística y Descenso de Gradiente

Linear Regression

x1 altura | x2 sexo --> y peso

- El modelo es una combinación lineal de características:

formula de linea recta +-

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n = \boldsymbol{\beta}^T \boldsymbol{x}.$$

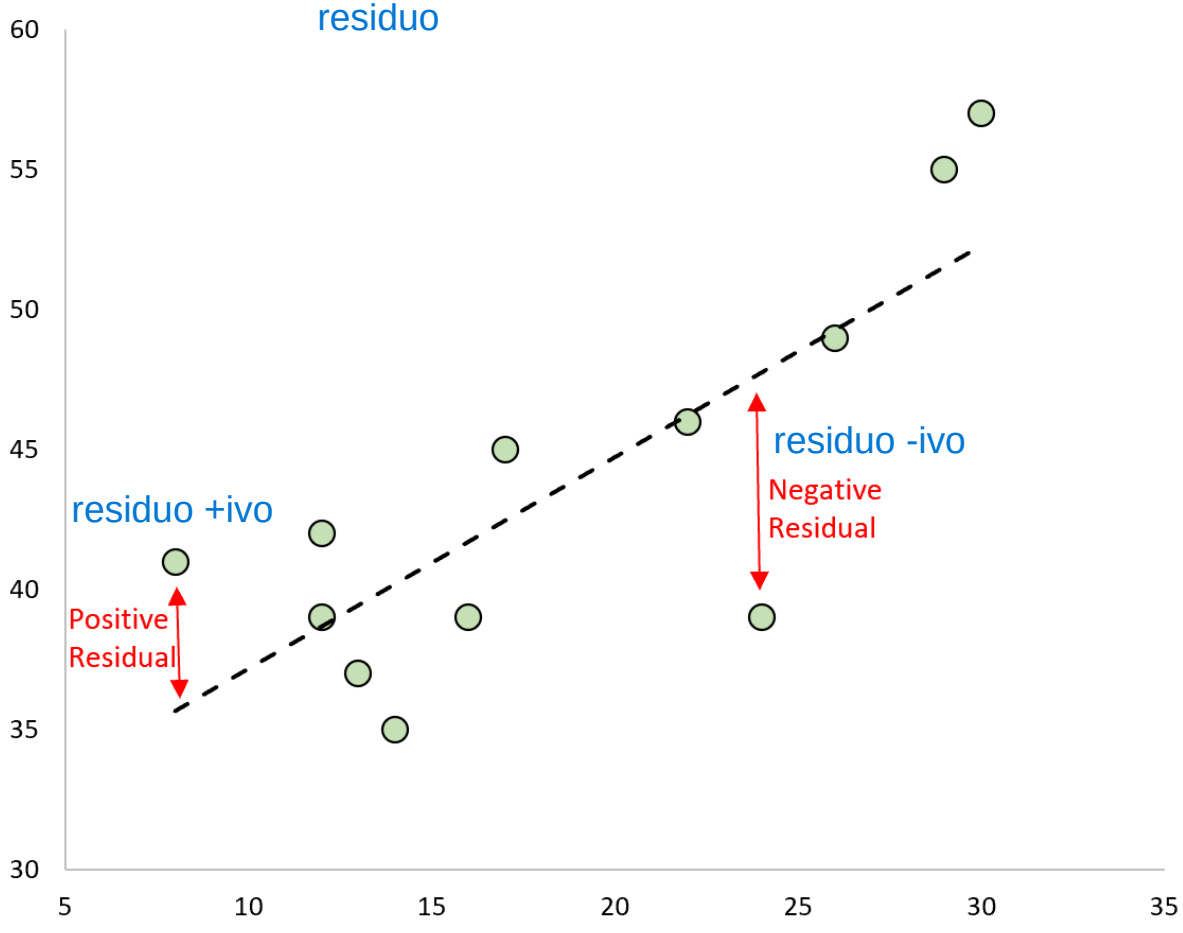
= [B0, B1, ...] * [x1,...,xn]

- Objetivo: Minimizar el Error Cuadrático Medio (MSE). MSE es un ejemplo de **función de pérdida**.
- $\mathcal{L} = MSE = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$

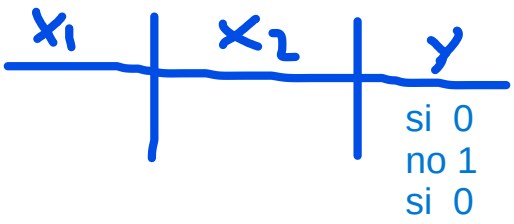
parametros libres que queremos que aprenda nuestra regresion

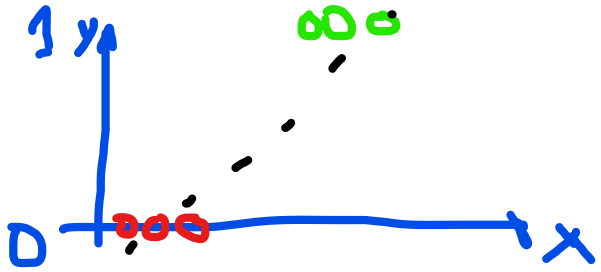
queremos que la funcion de perdida sea lo mas pequena que se pueda

residuo



Logistic Regression



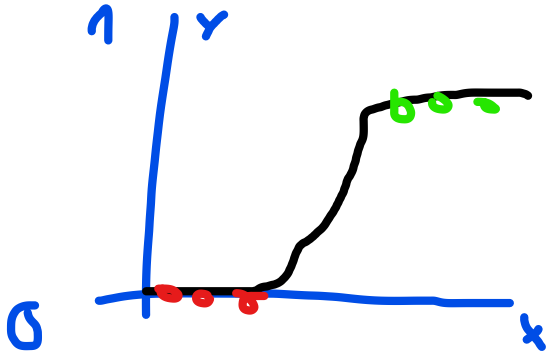


usar regresion lineal para solo dos opciones.

una primera idea que no es optima

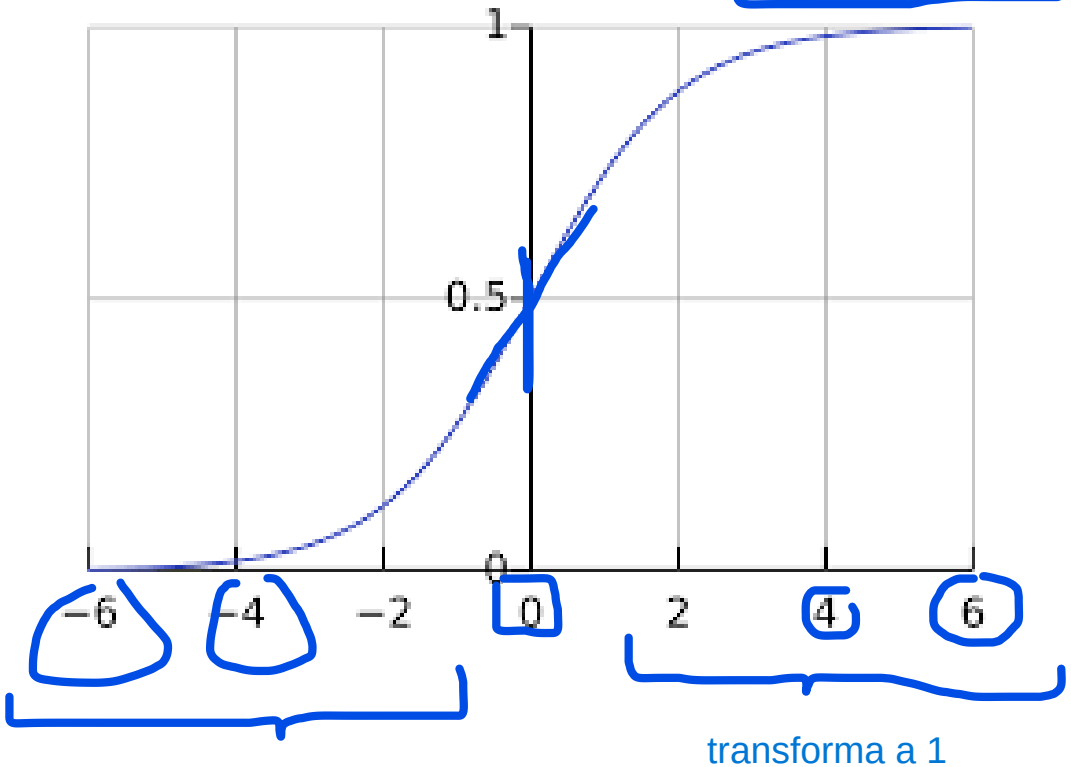
- ¡Recuerda que los Regresores Logísticos son clasificadores!
- Idea básica: Usar regresión para ajustar los identificadores de clase (0 y 1, en un problema binario).
- Problema: Salidas no interpretables. ¿Qué pasa si el modelo produce $\hat{y} = 1.9$ o $\hat{y} = -5$?
- La solución: permitir que el regresor produzca cualquier número en $(-\infty, \infty)$ y luego comprimirlo al rango $(0, 1)$ para interpretarlo como la probabilidad de pertenecer a la clase 1.

Los valores muy pequenos transformarlos en 0 y los grandes en 1



Logistic Regression

- Si la probabilidad predicha de pertenecer a la clase 1 es $\hat{p}$, la salida sin comprimir puede interpretarse como un logaritmo de ratio de probabilidades $\log \frac{\hat{p}}{1-\hat{p}}$.
- La operación de compresión se llama sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$



Logistic Regression

- Si la probabilidad predicha de pertenecer a la clase 1 es $\hat{p}$, la salida sin comprimir puede interpretarse como un logaritmo de ratio de probabilidades $\log \frac{\hat{p}}{1-\hat{p}}$.
- La operación de compresión se llama sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$
- El modelo completo es:

1. regresion
$$z = log \frac{\hat{p}}{1-\hat{p}} = \beta^T x$$
z ∈ (-inf, inf)

2. compresion
$$\hat{y} = \hat{p} = \sigma(z)$$
transforma en una probabilidad con sigmoide

funcion de perdida

β ∈ [0,1]

- Objetivo: **entropía cruzada binaria** (**binary cross-entropy**).

$y_i = 1$

$y_i = 0$

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

L de loss

deaparece

desaparece

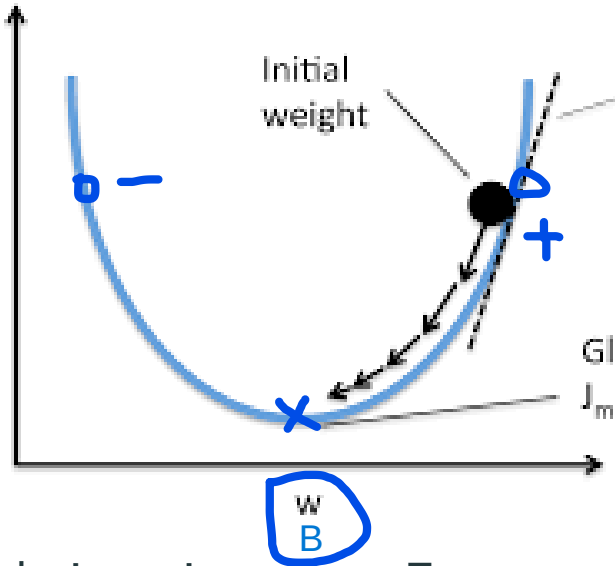
- Intuición de la entropía cruzada binaria: Mide la disimilitud entre la distribución verdadera (etiquetas reales) y la distribución predicha (salidas del modelo).

Como ajusto los B para hacer que L sea minima → Optimizando -> Derivando

Descenso de Gradiente (Gradient Descent, GD)

- Problema: Dado un conjunto de datos y un modelo de regresión lineal/logística, ¿cómo *aprendemos* los mejores parámetros/pesos β?
- Solución: ¡Simplemente minimizar $\mathcal{L}(\beta)$!

derivada > 0 , creciente
derivada < 0 , decreciente



solo un beta

Para probar la grafica hay que probar todos los valores posibles. Computacionalmente imposible

buscar el mejor beta para que sea el minimo posible

si derivada +iva -> izquierda
si derivada -iva -> derecha

- ¿Cómo? Hagámoslo iterativamente. Empezamos en un β inicial. Encontramos la dirección que minimiza $\mathcal{L}(\beta)$ y nos movemos un pequeño paso en esa dirección. Es decir:

Se hace por intervalos hasta llegar al 'fondo del cuenco'

$$\beta = \beta - \alpha \nabla_{\theta} \mathcal{L}(\beta)$$

gradiente $\nabla_\beta \mathcal{L}(\beta)$

- α es la llamada **tasa de aprendizaje**.

valores pequenos

Code Exercise: Regression from scratch

actualizar la beta hasta que sea muy proxima a 0

Variantes de Regresión Lineal: Lasso, ElasticNet, y Ridge

Técnicas de Regularización

L = MSE

L = BCE

Existen muchas variantes de modelos lineales (tanto para clasificación como regresión) que utilizan **regularización**:

si lambda es muy grande, el resto tiene mas peso

L2 viene de ^2

- Ridge Regression (L2): $\mathcal{L}' = \mathcal{L} + \lambda \sum_{j=1}^{n} \beta_j^2$ coge todos los coef y los ^2
- Lasso (L1): $\mathcal{L}' = \mathcal{L} + \lambda \sum_{j=1}^{n} |\beta_j|$
- ElasticNet: Combinación de L1 y L2: $\mathcal{L}' = \mathcal{L} + \lambda_1 \sum_{j=1}^{n} |\beta_j| + \lambda_2 \sum_{j=1}^{n} \beta_j^2$
- Objetivo: Prevenir el sobreajuste, manejar la multicolinealidad

Exercise: Sklearn

Localiza en Sklearn el nombre de los regresores y clasificadores Lasso, Ridge y ElasticNet. ¿Cómo se llaman λ, λ_1 y λ_2 en el código?

u termino es el doble de otro

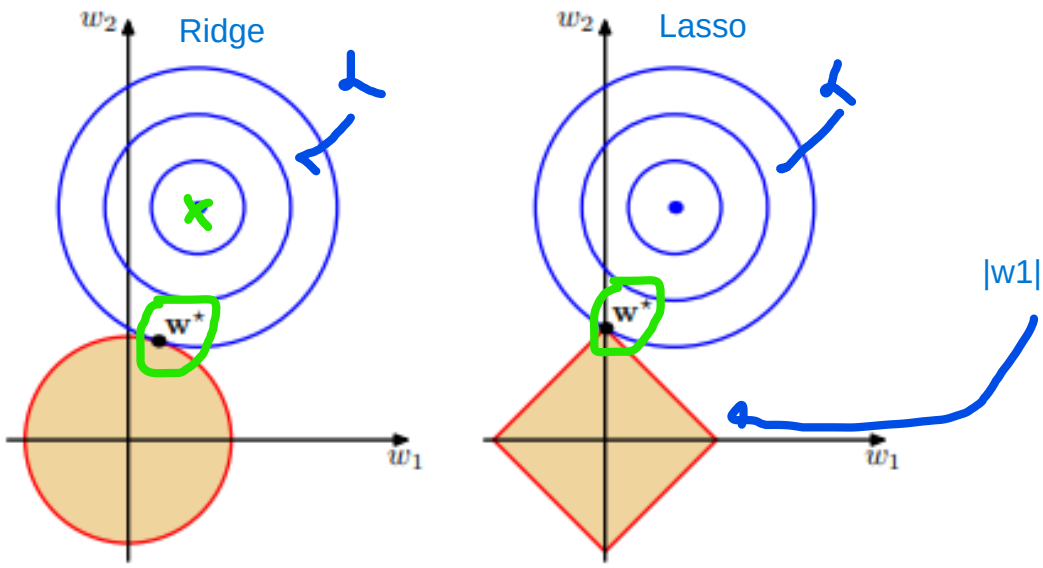
x1	x2
2	4
3	6
1	2
-2	-4

si se hace regresion lineal y existe multicolinealidad, algun termino se ira al infinito

Comparación de Técnicas de Regularización

Lo que hace ridge es hacer sus coef cercanos a 0

vamos a comparar:



Regularizacion

w1^2 + w2^2

eq de un circulo

punto optimo es el centro. pero ahora con la regularizacion el punto optimo es la interseccion. (esta rodeado en verde

|w1| + |w2|

- Ridge (L2):
  - Mejor manejo de la multicolinealidad
  - Reduce los coeficientes hacia cero (pero nunca exactamente a cero)
- Lasso (L1):
  - Mejor manejo de la multicolinealidad
  - Puede llevar coeficientes exactamente a cero (selección de características)

se cepilla coef que no parecen relevantes

Lasso si puede hacer sus coeficientes a 0, Ridge los acerca pero nunca exacto 0

- ElasticNet (L1 + L2): Combina los beneficios de Ridge y Lasso
  - Maneja mejor los siguientes escenarios:
    - Cuando n < p (más características que muestras) muchas col pero pocas rows
    - Cuando las características están correlacionadas en grupos

el que se suele usar en la practica

Imagen de PRML

Lo malo es que la validacion cruzada sera mas dificil (por lam1 y lam2)

ej: cuando el genA se expresa, tambien se expresa el genB. seria optimo que si tienen que ver con la enfermedad, saber ambos es importante, pero Ridge y Lasso se cepiillaria a una de ellas. Pero ElasticNet mantiene caracteristicas que estan relacionadas

Decision Trees

El segundo gran algoritmo a lo mejor

Decision Trees: Visión General

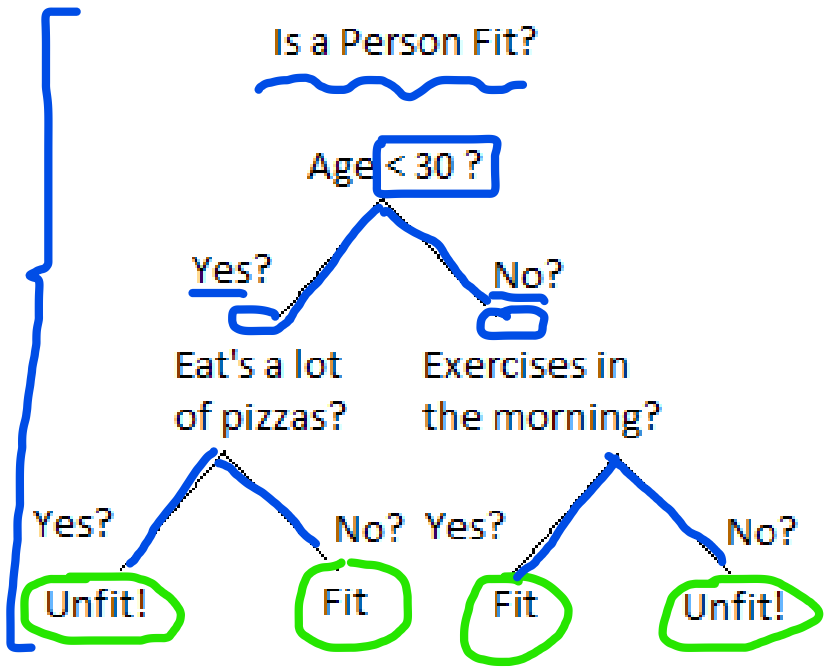
- Modelo en forma de árbol de decisiones y sus consecuencias
- Usado tanto para clasificación como regresión (`DecisionTreeClassifier` y `DecisionTreeRegressor`)
- Ventajas: Interpretabilidad, maneja relaciones no lineales
- Desafíos: Sobreajuste, inestabilidad

Ser interpretable es importante en muchos campos como en la medicina

Con validacion cruzada + otros conceptos se puede esquivar

Para llegar a la solucion se van haciendo preguntas, y segun las preguntas se hacen decisiones

Nodos hoja, nodos terminales de un arbol



Algoritmo de Decision Trees

Algorithm 1 Algoritmo de Decision Trees

1: Comenzar con todo el conjunto de datos
2: Para cada característica, encontrar la división que minimiza la función de costo
3: Elegir la mejor división entre todas las características
4: Dividir el nodo y crear nodos hijos
5: Repetir pasos 2-4 para cada nodo hijo hasta cumplir los criterios de parada

hasta la profundidad deseada

- Algoritmos históricos:
  - ID3 (Iterative Dichotomiser 3) por Ross Quinlan (1986)
  - C4.5: Sucesor de ID3, maneja atributos continuos
  - C5.0: Sucesor comercial de C4.5, más rápido y eficiente en memoria
  - CART: Soporta tanto clasificación como regresión, usa divisiones binarias

CART es el mas usado hoy en dia.

CART usa **impureza de Gini** para clasificación como función de costo, mientras que los otros algoritmos usan **ganancia de información**.

Selección de la Mejor División

1) Seleccionar la división para una característica dada:

probamos todos umbrales posibles

- Para cada posible umbral t en la característica f, dividir los datos en dos grupos:

pero si tenemos datos categoricos, podria ser en mas de 2 grupos

$$S_{left} = \{x_i | x_i[f] \leq t\}, \quad S_{right} = \{x_i | x_i[f] > t\}$$

asociamos coste

- Calcular la función de costo (ej., impureza Gini/Ganancia de Información, MSE) para la división.

elegimos el mejor

- Elegir el umbral t^* que minimiza el costo:

151

$$t^* = \arg\min_t \left(\frac{|S_{left}|}{|S|} \cdot \text{Costo}(S_{left}) + \frac{|S_{right}|}{|S|} \cdot \text{Costo}(S_{right}) \right)$$

2) Elegir la mejor división entre todas las características:

- Repetir el proceso anterior para cada característica.
- Elegir la característica y el umbral correspondiente que resulten en el menor costo general.

Detour: Entropía a través de Longitudes de Código

Ejemplo: Frecuencias de Palabras

Palabra	Frecuencia
"cat"	$50\,\% = \frac{1}{2}$
"dog"	$25\,\% = \frac{1}{4}$
"stats"	$12.5\,\% = \frac{1}{8}$
"AI"	$12.5\,\%$ $\frac{1}{8}$

hacer uso de la frecuencia de las palabras

asociar menos bits a lo mas frecuente

Codificación Fija vs. Óptima

Longitud Fija (2 bits):

Palabra	Código
cat	00
dog	01
stats	10
AI	11

a cada palabra se les asocia dos bits

pero gasta bits inecesariamente

Codificación Óptima:

Palabra	Código	Longitud
cat	0	1 bit
dog	10	2 bits
stats	110	3 bits
AI	111	3 bits

Longitud Media del Código

- Fija: 2 bits siempre
- Óptima: $0.5 \cdot 1 + 0.25 \cdot 2 + 0.125 \cdot 3 + 0.125 \cdot 3 = 1.75$ bits

Una reduccion substancial

Detour: Entropía a través de Longitudes de Código

log(2^x) = x

Como se hace el calculo para saber cuantos bits necesitan

Idea Clave: Bits Necesarios vs. Probabilidad

- Si $p = \frac{1}{2}$ Necesita 1 bit $-\log_2(\frac{1}{2}) = 1$
- Si $p = \frac{1}{4}$ Necesita 2 bits $-\log_2(\frac{1}{4}) = 2$
- Si $p = \frac{1}{8}$ Necesita 3 bits $-\log_2(\frac{1}{8}) = 3$

Patrón General
Cuando la probabilidad es $p = (\frac{1}{2})^n$, necesitamos n bits
Resolviendo para n: $n = -\log_2(p)$

Entropía
Entropía = Bits promedio necesarios $= \sum p(x) \cdot (-\log_2(p(x)))$

la esperanza

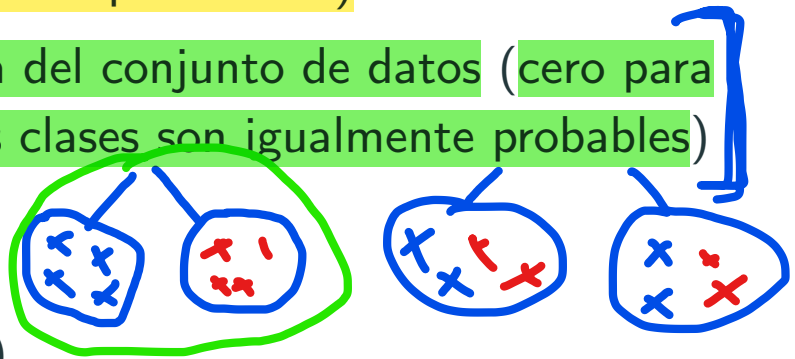
Detour: Entropía y Ganancia de Información

Diferentes Interpretaciones de la Entropía

Este concepto ayuda a medir la impureza en un conjunto de datos

- **Teoría de Codificación**: Número mínimo promedio de bits necesarios para codificar eventos en una distribución
- **Teoría de la Información**: Contenido de información esperado o sorpresa de los eventos (eventos raros son más sorprendentes)
- **Decision Trees**: Medida de impureza del conjunto de datos (cero para conjuntos puros, máximo cuando las clases son igualmente probables)

este es el mejor, menos entropia



Formulación Matemática

- Entropía: $H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$
- Ganancia de Información: $IG(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} H(S_v)$

Entendiendo la Ganancia de Información

- Representa la reducción en entropía después de dividir por el atributo A
- $H(S)$: Entropía (incertidumbre) original del conjunto de datos
- $\sum \frac{|S_v|}{|S|} H(S_v)$: Promedio ponderado de entropía después de la división
- Mayor ganancia = Mayor reducción en incertidumbre = Mejor división

Decision Trees: Criterios de Parada y Selección de Profundidad

¿Cuándo Dejar de Hacer Crecer el Árbol?

- Se alcanza la profundidad máxima
- El nodo se vuelve puro (todas las muestras de la misma clase)
- Muy pocas muestras en el nodo

El Compromiso de la Profundidad

Uno de los hiperparametros clave -> Depth

- **Muy Superficial** → Subajuste (demasiado simple)
- **Muy Profundo** → Sobreajuste (demasiado complejo)

Code Exercise: Visualización de Decision Trees

Code Exercise: Decision Trees y Sklearn

Lee los parámetros de DecisionTree y relaciónalos con todos los conceptos explicados.

Research Project: Decision Trees from Scratch

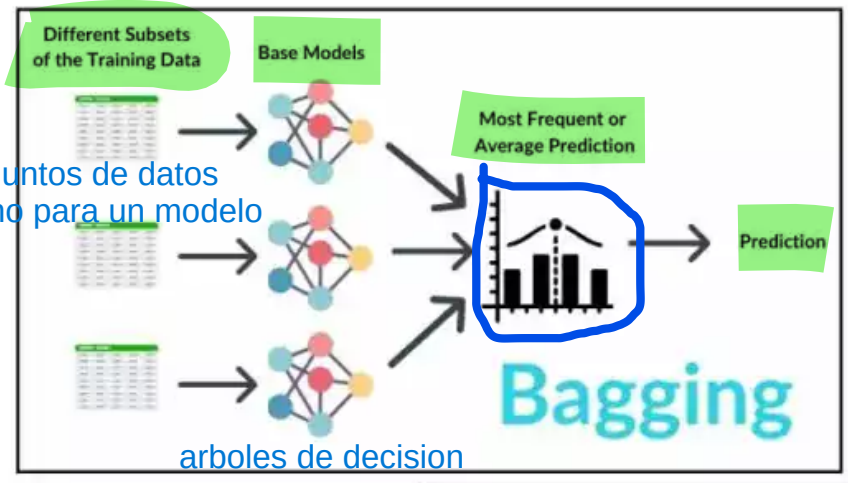
Ganan muchas competiciones

Ensembles y Random Forest

Preguntan a varios modelos y compara

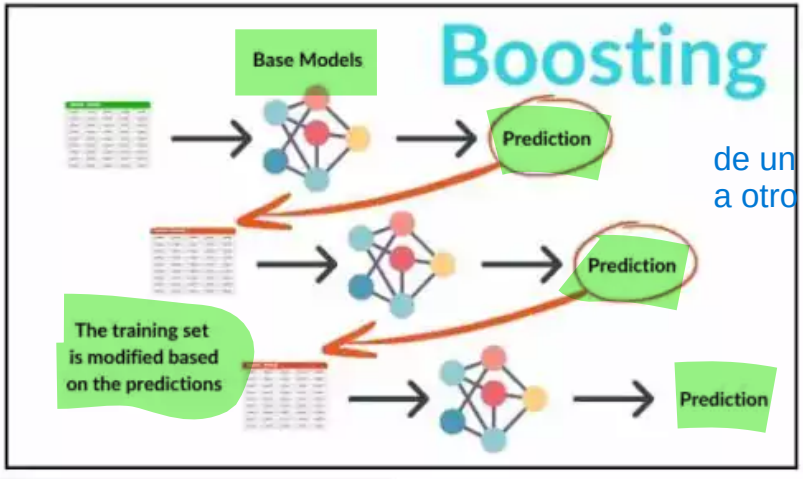
Métodos de Ensemble: Visión General

- Combinan múltiples modelos para mejorar el rendimiento
- Tipos: Bagging, Boosting, Stacking
- Objetivo: Reducir la varianza y el sesgo, mejorar la generalización



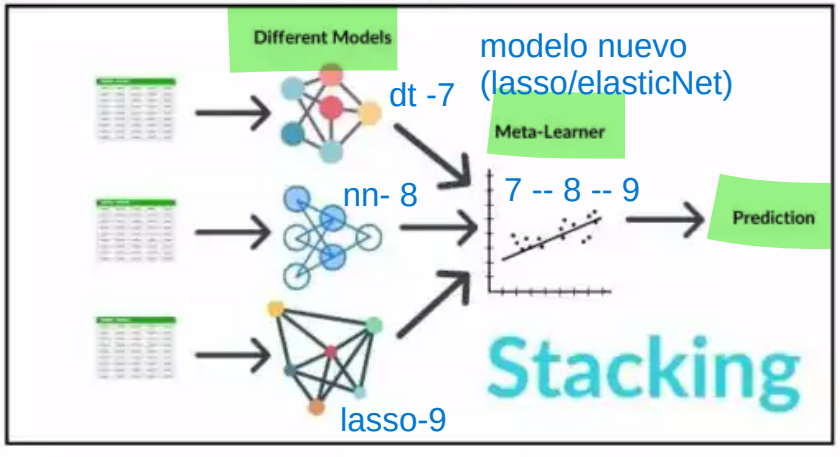
subconjuntos de datos
cada uno para un modelo

arboles de decision



de un modelo a otro

como una cadena (como balance cascade)
partes del conjunto original, entreno modelo, hace preds. Cogemos ejemplos que fallan mucho y se los pasa a un segundo modelo (cadena de modelos).
Se puede meter submuestreo.
Suelen ser modelos del mismo tipo.



Empieza igual que bagging, pero tenemos distintos modelos en cada subtarea.
Para combinar las predicciones se crea un ultimo modelo (meta-learner) que recibe todas las predicciones (e info extra a veces).
Como le puedo pasar contexto, puede discernir que tareas hace mejor cada prediccion. Y usar esa.

Bagging y Boosting

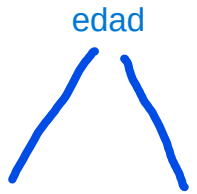
- **Bagging (Bootstrap Aggregating)**
  - Crear múltiples subconjuntos de datos con reemplazamiento
  - Entrenar un modelo en cada subconjunto
  - Combinar predicciones (votación para clasificación, promedio para regresión)
  - Reduce la varianza, ayuda a prevenir el sobreajuste
- **Boosting**
  - Método de ensemble secuencial
  - Cada modelo intenta corregir los errores del modelo anterior
  - Algoritmos populares: AdaBoost, Gradient Boosting
  - Reduce el sesgo y la varianza

Stacking

- También conocido como *Stacked Generalization*
- Combina predicciones de múltiples modelos usando un *meta-learner*
- Proceso:
  - Entrenar varios *modelos base* ***diversos*** en el conjunto de datos original
  - Usar modelos base para hacer predicciones en un conjunto de validación
  - Entrenar un *meta-learner* con las predicciones de los modelos base
- El meta-aprendiz aprende cómo combinar mejor las predicciones de los modelos base
- Puede (debe) combinar diferentes tipos de modelos (ej., decision trees, SVMs, redes neuronales)
- A menudo usado en competiciones de machine learning para lograr alta precisión

Research Project: Stacking

Algoritmos de Ensemble



- **Random Forest**
  - Ensemble de decision trees
  - Ideas clave:
    - Muestreo Bootstrap (bagging)
    - Selección aleatoria de características en cada división
  - Ventajas: Robusto, maneja datos de alta dimensionalidad
  - Desventajas: Menos interpretable que un único decision tree
- **XGBoost (Extreme Gradient Boosting)**
  - Implementación avanzada de gradient boosting
  - Características clave:
    - Regularización para prevenir el sobreajuste
    - Maneja valores faltantes
    - Procesamiento paralelo eficiente
  - Alto rendimiento en muchas competiciones de machine learning

coge por muestreo los que estan disponibles

para un arbol y repito para un segundo arbol/

para cosas como para medicos no les gusta

Boosting

paquete xgboost

Research Project: XGBoost

Interpretabilidad de Modelos

Interpretabilidad de Modelos: De Modelos Simples a Complejos

¿Qué Hace que un Modelo sea Interpretable?

- **Interpretabilidad Global**: Entender cómo el modelo toma decisiones en general
- **Interpretabilidad Local**: Entender predicciones individuales
- **Importancia de Características**: Una forma más débil de interpretabilidad, común en modelos complejos

en base a columnas

decisiones especificas

Modelos Verdaderamente Interpretables

- **Modelos Lineales**:
  - Interpretación directa de coeficientes
  - Contribución clara de características
- **Árbol de Decisión Individual**:
  - Caminos de decisión explícitos
  - Representación visual

Modelos Complejos (ej., Ensembles)

- Solo ofrecen importancia de características
- Ocultan interacciones complejas
- Sacrifican interpretabilidad por rendimiento

No podemos ver como ha tomado la decision, como ver el camino como arbol individual

Los datos a veces pueden tener sesgos peligrosos. Como aprende de datos, si los datos originales, por ejemplo, son discriminatorios, podria tomar decisiones muy polemicas.

Punto Clave

Importancia de características $\neq$ Verdadera interpretabilidad. Mientras que los métodos ensemble pueden decirnos qué características son importantes, no pueden explicar cómo estas características interactúan o por qué se hacen predicciones específicas.

Problemas de Investigación

Problema de Investigación: Entropía y Teoría de la Información

Lectura de Artículo:

- Leer el trabajo original “A Mathematical Theory of Communication"de Claude Shannon.

Tarea de Escritura:

- Escribir un blog explicativo sobre entropía, entropía cruzada y ganancia de información, incluir visualizaciones y ejemplos de ciencia de datos y decision trees.

Problema de Investigación: Decision Trees desde Cero

Lectura de Artículo:

- Leer el Capítulo 9 de “The elements of statistical learning".

Tarea de Programación:

- Implementar un clasificador decision tree desde cero.
- Probar el modelo en un conjunto de datos de clasificación.

Problema de Investigación: Stacking

Lectura de Artículo:

- Leer “Stacked Generalization"de David H. Wolpert.

Tarea de Programación:

- Implementar un ensemble stacking usando múltiples aprendices base (ej., regresión logística, decision trees).
- Probar el rendimiento en una tarea de regresión o clasificación.

Problema de Investigación: XGBoost

Lectura de Artículo:

- Leer “XGBoost: A Scalable Tree Boosting System"de Chen y Guestrin.

Tarea de Programación:

- Escribir código que demuestre el uso de XGBoost.
- Escribir código para optimizar los hiperparámetros de XGBoost usando búsqueda Bayesiana.

Problema de Investigación: Support Vector Machines (SVM)

Lectura de Artículo:

- Leer el Capítulo 12 de “The elements of statistical learning".

Tarea de Programación:

- Probar SVMs en conjuntos de datos linealmente separables y no linealmente separables.