

Resumen organización de datos

¿Por qué es necesario graficar?

- Las técnicas de visualización de datos son muy importantes para nuestro trabajo como para comunicarlo.
- La cantidad de tipos de gráficos disponibles es enorme y es importante entenderlos y saber para qué es útil cada uno.
- Entender de forma eficiente los datos.
- Comunicar de forma concisa y clara.
- Encontrar patrones/relaciones.
- El análisis descriptivo es uno de las partes principales de cualquier análisis relacionado con un proyecto de ciencia de datos o de una investigación específica.
- La agregación de datos, el resumen, y la visualización son algunos de los pilares principales que respaldan esta área.
- La visualización de datos es una herramienta poderosa y ampliamente adoptada debido a su efectividad para extraer la información correcta, comprender e interpretar los resultados de manera clara y fácil.
- Tratar con conjuntos de datos multidimensionales con más de una variable o atributo comienza a causar problemas, y que estamos restringidos a comunicar dos dimensiones (a lo sumo 3).
- Los gráficos no son simplemente: “imágenes bonitas”.
- No toda la información importante se puede adivinar a través del análisis estadístico...
- Todos estos gráficos tienen la misma medida y desvío estándar.

Visualización para machine learning

- En aprendizaje automático, la visualización se utiliza para:
 - Análisis de los datos:
 - Para examinar si los datos satisfacen los supuestos requeridos para el método.
 - Tienen complicaciones inesperadas como valores atípicos o no linealidad.
- Evaluar el ajuste del modelo:

- Predicho vs observado.

Visualización de datos

Gráfico de distribución continua

En estos tipos de gráficos en el eje “y” (eje vertical) se tiene una métrica de cantidad o algo similar, análogo según el tipo de problema analizado. En el eje “x” se puede tener un soporte continuo o discreto. Un error muy común confundir entre soporte continuo y soporte discreto.

Histograma

En los gráficos de tipo histogramas, se obtiene una agrupación en barras (bins) y se puede elegir la cantidad de barras en la que uno quiere visualizar el gráfico.

Un histograma es útil por varias razones, se puede visualizar la distribución de los datos que pueden ser de diferentes tipos (symmetric/unimodal, skew left, skew right, uniform, bimodal, multimodal).

Algunos errores comunes de los histogramas son:

- No empezar desde el cero en cualquier eje.
- Una mala elección del tamaño de barras.
- Elegir border apropiados para que el gráfico sea fácil de visualizar.

El histograma puede verse anormalmente “desigual” simplemente debido a la cantidad de valores que posiblemente podría tomar cada contenedor (elección de tamaño de barras y bordes apropiados).

Density plot

Este gráfico es una variación del histograma, mostrando un **contorno suavizado**, son mejores para determinar la forma de distribución porque **no se ven afectados por el número de contenedores**. Pero también tienen sus **desventajas**, en el eje “y” no se muestra una cantidad, sino una densidad (una representación porcentual del agrupamiento de los datos) que es **poco interpretable**, por otro lado, la **representación suavizada puede traer inconsistencias** en ciertas partes del gráfico (ej.: **extremos**).

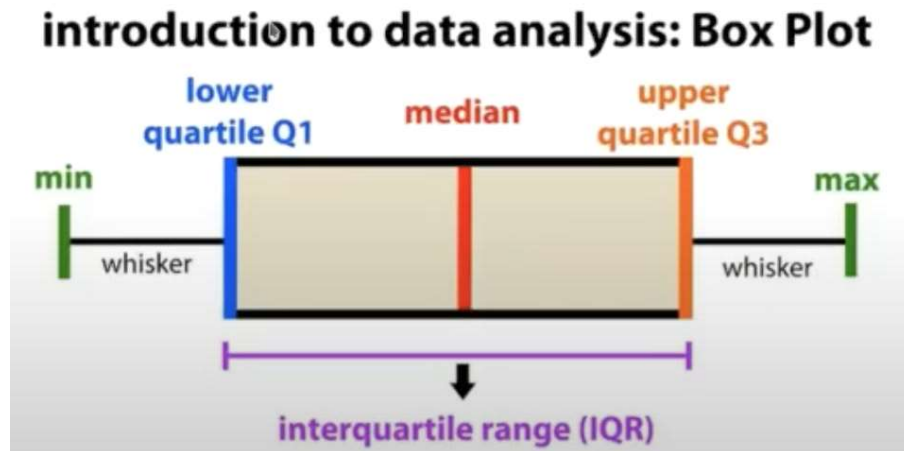
Algunos números útiles

- **Media:** es el promedio.

- **Mediana:** es el valor que está en la mitad de la población.
- **Cuartil:** son los valores límite que dejan al 25% de la población entre ellos.
 - Para calcular el cuartil hay varias posibilidades, en todas se debe cumplir que se descarta la misma porción de la población. En **numpy**, si queremos calcular el cuartil 1 se hace la siguiente cuenta:
 - $(N-1) * 0.25$.
 - Luego se devolvería: $\text{array}[1] + (\text{array}[2] - \text{array}[1]) * 0.75$.
- **Rango intercuartílico:** el rango entre el cuartil 1 y el cuartil 3.

Box plot

Un diagrama de caja o Box plot muestra visualmente la distribución de los datos numéricos y la asimetría mediante la visualización de los cuartiles (o percentiles) y los promedios de los datos.



De distribución discreta

Gráfico que muestra cantidades y valores discretos, en forma de torta o barra y con la posibilidad de modelizarlo en porcentajes.

Bar plot

Es similar al histograma, pero con la diferencia de que se muestra rangos discretos y no continuos. Con el bar plot se pueden poner dos barras o más para hacer una comparación de dos o más variables.

Stacked bar plot

Este gráfico consiste en apilar las barras para comparar múltiples variables, es una forma diferente de representación de lo mismo igual al bar plot.

Treemap

Es un gráfico en el cual que se lo puede pensar como un árbol, tiene subdivisiones y las divisiones pueden contener unos a otros.

De relación

Gráfico que permite medir relaciones o correlaciones entre gráficos, para eso, lo que se hace es superponer en un solo gráfico dos medidas o más.

Scatter plot (de dispersión)

Hablando de correlaciones, una forma de ver si hay dos variables en un set de datos que están relacionadas o correlacionadas de alguna manera, es mostrar un gráfico de dispersión. Este gráfico lo que hace es mostrar como punto las variables en el eje cartesiano, donde en un eje representa un set de datos y en el otro eje las demás.

Correlación de Pearson

Los diagramas de dispersión son útiles para ver si dos variables están correlacionadas

Para 2 variables podemos medir su correlación lineal con el coeficiente de correlación (Pearson). Este coeficiente, es una función que mide cuán relacionada están 2 variables de forma línea.

- Si da 0 NO existe correlación
- Si da 1 Están relacionadas linealmente de forma perfecta (todos los puntos están en una línea)
- Si da -1 Existe una correlación negativa perfecta.

La correlación de Pearson no necesariamente nos indica si hay relación o no, ya que existen otros tipos de correlaciones (polinómicas, exponenciales, etc.). Pearson solo nos dice si existe una correlación lineal.

Regression plot

En el gráfico se incluye una guía visual que muestra la relación entre las variables y eso es lo representativo de este gráfico de regresión.

Heatmap

Es un gráfico que permite representar en tres dimensiones incluyendo el eje z que es una escala de color. Sirve para comparar distribuciones en donde ambos ejes son discretos y un tercero de “profundidad” numérico.

Generalmente surge de calcular algún agregado del grupo al que corresponde el rectángulo.

Series de tiempo

En este gráfico, el eje “x” siempre va a representar el transcurso del tiempo y en el eje “y” se puede visualizar de diferentes formas (ej.: box plots y bar plot superpuestos a lo largo del tiempo).

Lineplot

Este gráfico se caracteriza por representarse con una línea continua a lo largo del tiempo (eje “x”).

Violin plots

Es un diagrama de caja con un diagrama de densidad de kernel rotado en cada lado.

El diagrama de violín es similar a los diagramas de caja, excepto que también muestran la densidad de probabilidad de datos en diferentes valores.

Falacias con los datos

Paradoja de simpson

Es una paradoja en la cual una tendencia que aparece en varios grupos de datos desaparece cuando otros grupos se combinan y en su lugar aparece la tendencia contraria para los datos agregados. Cuando la asociación entre dos variables cambia completamente cuando se tiene en cuenta el efecto de una tercera variable que no se había tenido en cuenta.

Sesgo de supervivencia

Explicándolo a través de un ejemplo, en Estados Unidos, durante la segunda guerra mundial, analizan los aviones que vuelven del combate para poder reforzarlos y aumentar el porcentaje de supervivencia de estos. Luego de analizar los aviones que volvieron de la guerra, concluyen que se debía reforzar los lugares donde más se

dañaron. Pero resulta que la lógica es todo lo contrario, justamente los lugares dañados de los aviones que volvieron, no eran críticos, ya que sobrevivieron. Por lo tanto, se debía reforzar los lugares que no estaban dañados.

Esto nos plantea que es necesario ver la totalidad de un problema, y no estar cegado solo con los datos que obtiene uno porque puede no estar completa. Siempre hay que preguntarse el origen de los datos.

Introducción a la ciencia de datos

Cuando hacemos ciencia de datos, estamos aplicando en general machine learning (aprendizaje automático) y se trata de ciertas técnicas y algoritmos para resolver problemas complejos que quizás no tengan una solución única u óptima o es muy costosa computacionalmente. Estas técnicas y algoritmos se aplican a un conjunto de datos de grandes volúmenes.

La ciencia de datos no es algo que antes no se practicaba, solo que se hacía de otra forma y con otro nombre llamado **datamining** que se puede resolver con modelos numéricos matemáticos sin necesidad de machine learning.

Variables

En los sets de datos para analizar, están representados en filas y columnas, cada fila es una observación y cada columna es un dato, es un atributo de la observación. Las columnas son variables.

- Variables independientes (entradas).
- Variables dependientes (salidas, categorías).
 - Dependen de los datos de entrada, son variables resultados que obtenemos luego de procesar los datos de entrada.
 - A veces puede haber variables dependientes dentro del mismo set de datos de entrada.

Las variables pueden ser:

- Cualitativas, estas variables describen características discretas.
 - Texto:
 - Nombres (categorías, ej.: países).
 - Ordinales (ej.: poco, mucho, muchísimo).
 - Numéricas:
 - Nominales.
 - Ordinales.

- Cuantitativas.
 - Discreta, conjunto de números continuos finito.
 - Continua, conjunto de números continuos infinito.

Variables y tipos de problemas

- Si la variable dependiente es **cualitativa**, el tipo de problema es de **clasificación**.
- Si la variable dependiente es **cuantitativa**, el problema es de **regresión**.
- Si **NO** hay variables dependientes, el problema es de **agrupamiento**.

Outliers (valor atípico)

Son valores que no están dentro del rango esperado, que pueden ser errores de medición, puede ser un valor real por más que sea poco probable (hay que considerar si se lo pondera o se lo saca del conjunto). A veces dependiendo del problema, tal vez, uno necesita encontrar esos valores atípicos.

Correlación de variables

“Dos variables están correlacionadas cuando varían de igual forma sistemáticamente”. Cuando dos variables están correlacionadas, quiere decir que van a variar de forma similar cuando se los compara una con otra.

- **Positiva**, cuando las variables varían de forma similar.
- **Negativa**, cuando las variables varían de forma opuesta.
- **Sin correlación**, cuando las variables varían de forma independiente.

Correlación **NO IMPLICA** causalidad:

- Que dos variables tengan alto índice de correlación no significa que una cause la otra (no implica que una modifique a la otra).
- Las relaciones de causalidad son más difíciles de encontrar y demostrar.
- Las correlaciones pueden suceder por otros motivos como: Una tercera variable que “empuja” a ambas o simplemente azar.

Varianza

La varianza es el promedio de la diferencia, entre todas las observaciones, respecto de su media.

Covarianza

En probabilidad y estadística, la covarianza es un valor que indica el **grado de variación conjunta de dos variables aleatorias respecto a sus medias**.

Es el dato básico para determinar si existe una dependencia entre ambas variables y además es el dato necesario para estimar otros parámetros básicos, como el coeficiente de correlación línea o la recta de regresión.

Una forma de visualizar las covarianzas, es graficándolas en un eje cartesiano todas las observaciones entre ambas variables y ver si existe alguna tendencia, que puede ser positiva, negativa o no exista ninguna tendencia.

Correlación de Pearson

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \text{Var}(Y)}}$$

donde

- σ_{XY} es la **covarianza** de (X, Y)
- σ_X es la **desviación estándar** de la variable X
- σ_Y es la **desviación estándar** de la variable Y

Esto no es necesario calcularla, ya que con pandas se calcula solo.

Para dos variables podemos medir su correlación línea con el coeficiente de correlación r (Pearson). Este coeficiente, es una función que mide cuán relacionada están dos variables de forma línea.

- Si da 0, NO existe correlación (no existe correlación lineal).
- Si da 1, están relacionadas linealmente de forma perfecta (todos los puntos estarán en una línea).
- Si da -1, existe una correlación negativa perfecta.

Correlación de Pearson: desvío estándar

Es una medida que se utiliza para **cuantificar la variación o la dispersión de un conjunto de datos numéricos**.

Una **desviación estándar baja** indica que la mayor parte de los datos de una muestra tienden a estar agrupados **cerca de su media** (también denominada el valor esperado), mientras que una desviación estándar alta indica que los datos se extienden sobre un rango de valores más amplio.

Métodos de regresión

La primera forma de regresión línea documentada fue el método de los **mínimos cuadrados** que fue publicada por Legendre en 1805, Gauss publicó un trabajo en donde desarrollaba de manera más profunda el método de los mínimos cuadrados.

El **concepto de regresión** proviene de la genética y fue popularizado por Sir Francis Galton a finales del siglo XIX con la publicación de *Regression towards mediocrity in hereditary stature*. Galton observó que las características extremas (por ejemplo, la altura) de los padres no se transmiten por completo a su descendencia. Más bien, las características de las descendencias retroceden hacia un punto mediocre (un punto que desde entonces ha sido identificado como la media).

Con el método de regresión buscamos predecir un valor en un rango continuo, para ciertos valores de entrada. Ejemplos:

- Temperatura
- Valor de una propiedad
- Etc.

El método más simple y antiguo es el método de regresión línea o ajuste línea que se puede calcular de varias formas, pero la forma tradicional es usando mínimos cuadrados. El método de mínimos cuadrados lo que va a hacer es aproximar todos los puntos del gráfico con una recta, tal que, la distancia euclidiana entre cada punto y la recta sea la mínima. Tal recta, me va a permitir predecir futuros valores. Es un método muy sensible a los outliers (valores atípicos).

Métodos de clasificación

Cuando resolvemos un problema de clasificación, buscamos, para ciertos datos de entrada, una categoría c de un conjunto C de categorías posibles. Estas categorías no solo son finitas, sino que además son conocidas de antemano.

Regresión Logística

En la regresión logística, lo que se busca es categorizar, clasificar. Es decir que, dado una serie de puntos, quiero encontrar una función (no es una recta en este caso) que separe los puntos en dos, en dos conjuntos.

Y una vez que se encuentra, puedo determinar para cualquier valor X futuro, el conjunto al cual pertenecerá. Está asociado a problemas de probabilidad.

Métodos de clusterización

Clustering

En este tipo de problemas se trata de agrupar los datos. Agruparlos de tal forma que queden definidos N conjuntos distinguibles, aunque no necesariamente se sepa qué signifiquen esos conjuntos.

El agrupamiento siempre será por características similares.

K-Means (algoritmo más usado y fundamental para la clusterización)

- El usuario decide la cantidad de grupo.
- K-Means elige al azar K centroides (K es la cantidad de grupos).
- Decide qué grupos están más cerca de cada centroide. Esos puntos forman un grupo. (Mide la distancia euclidiana de cada una de las observaciones a los centroides).
- K-Means recalcula los centroides al centro de cada grupo.
- K-Means vuelve a reasignar los puntos usando los nuevos centroides. Calcula nuevos grupos.
- K-Means repite punto 4 y 5 hasta que los puntos no cambian de grupo.

Entrenamiento

Cuando tenemos métodos supervisados, es necesario decirle al método con qué datos se lo va a entrenar, el conjunto de datos de entrenamiento tiene asignada la categoría o el dato de salida que nosotros queremos que regrese para determinada observación.

Un conjunto de datos etiquetado, es un conjunto que tiene los valores que nosotros queremos que devuelva para cada observación. Con un conjunto de datos de este estilo, se puede entrenar un método.

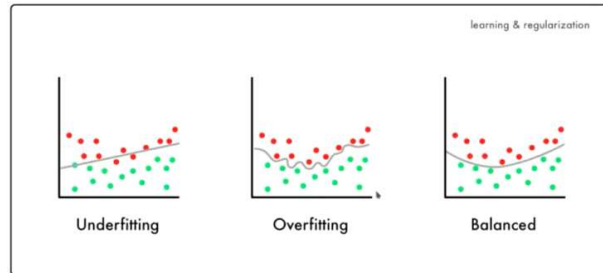
Hay métodos que se entrenan (son métodos supervisados) y métodos que no se entrenan (no supervisados). Por ejemplo, K-Means es un método no supervisado porque no es necesario darle un conjunto de datos previamente etiquetado, simplemente se le da los datos en crudo y los ordena. Los métodos supervisados suelen dar mejores resultados que los no supervisados.

Las técnicas comunes son el de partir el conjunto de datos en un conjunto de entrenamiento y uno de prueba más pequeño. Otra de las técnicas comunes es hacer varios conjuntos de prueba y entrenamiento.

Algunos errores que pueden surgir son que los conjuntos de datos no están balanceados, esto significa que hay un desbalance en la cantidad de los diferentes posibles casos y provoque que el entrenamiento no sea suficiente para ciertos casos que se pide predecir. Para solucionar este tipo de errores balanceando el conjunto de

datos, lo que se puede hacer es un **Undersampling**, retirar casos para que quede equitativo la cantidad de casos o un **Oversampling**, agrego datos falsos para aumenta la cantidad de casos que quiero.

Overfitting



Underfitting, es cuando está mal ajustado la clasificación de las observaciones, no se ajusta bien al modelo. Esto provoca que el modelo no pueda estimar bien para nuevos conjuntos de datos.

Overfitting, es cuando el modelo esta entrenado de más, sobre ajustado, esto quiere decir que el clasificador se aprendió de “memoria” por donde pasan cada punto y se pega a los bordes de estos, haciendo que la clasificación sea excelente y perfecta para el conjunto de entrenamiento. Pero este modelo no puede clasificar bien para observaciones nuevas, no es sensible a datos que no estén perfectamente alineados como el conjunto de entrenamiento.

Balanced, es cuando está clasificado de forma regular para todo el conjunto de datos, hay un estimador balanceado.

Métricas

Cuando creamos distintos modelos de clasificación (Regresión Logística, RandomForestClassifier, etc.) nos interesa conocer si el modelo está clasificando correctamente lo que queremos.

Hay distintas métricas que miden distintas que pueden ayudarnos en algunos casos, pero en otros pueden confundirnos, por ende, es muy importante tener bien claro que es lo que estamos midiendo.

Cuando uno hace una clasificación de tipo binaria, se puede cometer dos tipos de errores:

- **False Positive**, clasificar verdadero a algo que es falso.
- **False Negative**, clasificar falso a algo que es verdadero.

Precisión

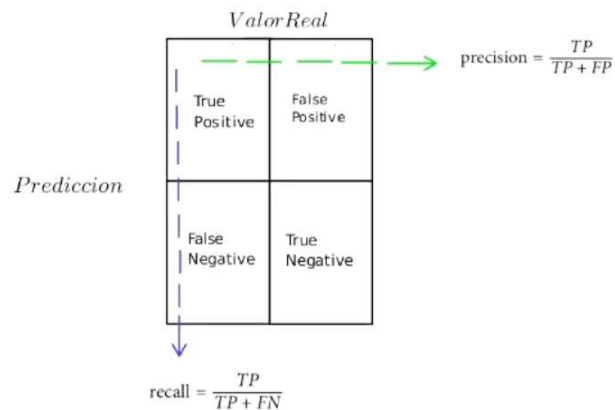
El cálculo de la **precisión** de una clasificación calcula la cantidad de casos verdaderos acertados en base al total de casos detectados y la ecuación es:

$$\text{Verdadero Positivo} / (\text{Verdadero Positivo} + \text{Falso Positivo})$$

Recall - Exhaustividad

Otra medida es la **recall/Exhaustividad** que calcula la cantidad de casos verdaderos acertados en base a la totalidad de casos verdaderos y la ecuación es:

$$\text{Verdadero Positivo} / (\text{Verdadero Positivo} + \text{Falso Negativo})$$



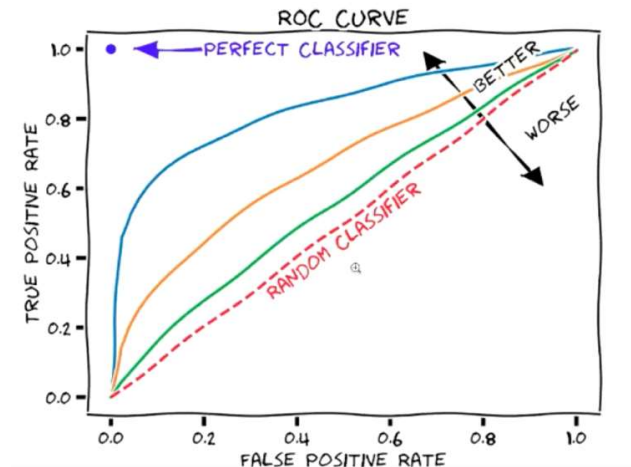
True Positive Rate – TPR (sinónimo de Recall)

Sobre el total de los casos, cuantos acertó el modelo que eran positivos verdaderos.

False Positive Rate – FPR

Sobre el total de los casos negativos, cuantos acertó modelo que eran negativos verdaderos.

$$\text{Falso Positivo} / (\text{Falso Positivo} + \text{Verdadero Negativo})$$



AUC

Una métrica bastante útil llamada **AUC** (AOC) es calcular el área debajo de la **curva ROC** recién determinada, esta medición va del cero al uno, y mientras más se acerque al uno, significa que el modelo es más preciso acertando los casos.

Hiperparametro

El hiperparametro es algo propio de cada modelo, es un parámetro que se le da al modelo para controlar la clasificación del modelo.

Aprendizaje Bayesiano

Uno de los usos más comunes y en donde más éxito ha tenido esta técnica es en la clasificación de documentos o clasificación de textos.

Clasificación de texto

La clasificación de texto sirve para asignar un tópico o categoría de forma automática a cualquier extracto de un texto. Se podría utilizar, por ejemplo, para:

- Clasificar un email como "spam" o "no spam".
- Identificar al autor del texto.
- Identificar el sexo o edad del autor de un texto.
- Identificar el lenguaje en el cual está escrito un texto.
- Realizar trabajo de análisis de sentimientos (sentiment analysis en inglés).

Definiciones:

- Entradas:
 - Un documento d (en lenguaje natural).
 - Un conjunto prefijado de clases $C=\{c_1,c_2,c_3,...,c_j\}$.
- Salidas:
 - Una clase c pertenece al conjunto C .

Métodos de clasificación de textos

Reglas escritas a manos:

Para la **detección del spam**, por ejemplo, podría tener una serie de reglas escritas por una persona que conozca sobre ese tópico:

REGLA: sí remitente (campo from) está en una lista-negra OR el asunto (subject) contiene la palabra: “viagra” => SPAM.

- **Pros:** la precisión puede ser muy alta.
- **Contras:** construir y mantener las reglas puede ser costoso. Tiene un recall muy bajo.

Aprendizaje automático supervisado

Puedo entrenar un clasificador diciéndole como clasificar.

- Entrada:
 - Un documento d .
 - Un conjunto prefijado de clases $C=\{c_1,c_2,c_3,...,c_j\}$.
 - Un conjunto m de documentos clasificados $m=\{(d_1,c_1),...,(d_n,c_j)\}$.
- Salidas:
 - Un clasificador entrenado $y: d \rightarrow c$. Dado un documento d , devuelve una clase c , con una determinada precisión y recall.

Tipos de clasificadores

- **Naïve Bayes** (Bayes ingenuo o bayes simple).
- **Logistic Regression** (Regresión logística).
- **Support-Vector Machines** (Máquinas de Soporte de vectores).

- **K-Nearest Neighbors** (K-Vecinos más cercanos).

Naïve Bayes – clasificación de texto

Un enfoque posible para la resolución del problema de la clasificación de texto es encararlo por el lado estadístico, entonces diría que, si tengo **n** documentos y **x** clases posibles, podría preguntarme:

¿Cuál es la probabilidad de que el documento **d** pertenezca a la clase **c**?

Parafraseando como probabilidad condicional:

Dado el documento **d**, ¿cuál es la probabilidad de que pertenezca a **c**? = **P(c | d)**

Y por el teorema de Bayes se puede plantear lo siguiente:

$$P(c | d) = \frac{P(d | c) P(c)}{P(d)}$$

Si tengo un **conjunto C** de clases, según Bayes, un documento **d**, pertenecerá a aquella clase que maximice su probabilidad condicional:

C_{map} = **argmax** P(c | d) para **c** ∈ **C**, (el conjunto de todas las clases).

- **map**: máximo a posteriorí, **C_{map}** es la clase candidata.
- **argmax**: función que devuelve el argumento máximo.

Por Bayes:

$$C_{\text{map}} = \underset{P(d)}{\text{argmax}} P(d | c) P(c), \quad c \in C$$

Como hay que calcular la probabilidad para cada **c** ∈ **C**, en cada caso, **P(d)** siempre va a ser la misma. El denominador **P(d)** queda como una constante. Se elimina entonces.

$$C_{\text{map}} = \underset{\cancel{P(d)}}{\text{argmax}} P(d | c) P(c), \quad c \in C$$

Si se elimina el denominador **P(d)**, la probabilidad no va a quedar normalizada, y no va a

ser realmente una probabilidad. Pero en este caso no nos interesa, ya que, lo que se busca es una clase candidata.

La clase candidata va a ser la clase con la mayor probabilidad de todas las clases. Calculándola con la siguiente función:

$$C_{\text{map}} = \text{argmax } P(d | c) P(c)$$

¿Cómo se calcula $P(c)$?

$P(c)$ es la probabilidad que tiene la clase de aparecer en una cantidad dada de documentos. Es más fácil de calcular, ya que es intuitiva.

$$p(c) = \frac{\text{cantidad de documentos de clase } c}{\text{cantidad de documentos totales}}$$

Cuento en mi conjunto de entrenamiento, cuantos documentos pertenecen a cada clase c del conjunto C y divido por el total de documentos. Con esa cuenta, obtengo la probabilidad de cada clase c . Esta probabilidad calculada, es un caso limitado, ya que la realidad puede ser muy diferente, otra posibilidad es asignar las mismas probabilidades a cada posible clase c .

Este valor $p'(c)$ no podemos conocerlo. Pero usando el conjunto de entrenamiento T , podemos estimar cuál sería esta probabilidad:

$$p'(c) = \frac{\text{cantidad de documentos de clase } c \text{ en } T}{\text{cantidad de documentos totales en } T}$$

¿Cómo se calcula $P(d | c)$?

$P(d | c)$ es la probabilidad de que dada una clase c , d sea un documento de ella. Esto es un poco más difícil de identificar. Y para ello tendremos que definir una forma de representar un documento.

Un documento, para Bayes Naive será una **bolsa de características**: $x_1, x_2, x_3, \dots, x_n$. Para nosotros, estas características serán las palabras que componen al documento.

Se le dice **bolsa de palabras** porque las palabras que contiene pueden estar repetidos y no estan ordenados.

Para ello asumiremos dos supuestos, muy importantes:

- No importa el orden de las palabras.
- Las probabilidades de cada característica, dada una clase c : $P(x_i | c_j)$ son independientes entre sí.

Problemas con los supuestos:

Bayes Naïve (Naïve = ingenuo).

Ahora bien, los problemas que pueden traer las características mencionadas anteriormente, son que, si una bolsa de palabras no tiene orden, entonces una oración puede no importar el orden de las palabras y siempre van a ser consideradas las mismas oraciones. Si no importa el orden de las palabras, entonces tampoco importaría si una palabra viene antes o después de otra, pero el problema con esto, es que, sabemos que hay ciertas palabras a las cuales le sigue otra palabra con muchísima más probabilidad que cualquier otra palabra.

Da a parecer que es un modelo malo y que da resultados malos, pero sorprendentemente devuelve resultados y es uno de los mejores clasificadores de texto.

Por ejemplo, dado un texto, primero separamos todas las palabras, este proceso se lo llama **tokenización**, cada palabra sería un **token** (opcionalmente podemos filtrar ciertas palabras de interés), contamos las ocurrencias de todas las palabras.

La forma tradicional de separar un documento en palabras, es hacer un **split** a todo el documento, utilizando los caracteres de espacio, salto de línea, tabulaciones y signos de puntuaciones como punto, punto y coma, signo de exclamación, signo de interrogación, etc. La separación en palabras, correctamente dicho, debería ser unigramas, ya que palabras puede ser un término muy ambiguo en diferentes contextos.

Stop Words, son las palabras que no aportan información, estas pueden ser una condición de filtro para así limpiar el documento y dejar las palabras importantes y de interés. Pero tal vez no siempre es necesario eliminarlas, ya que, a pesar de que es más costoso computacionalmente procesarlas, pero pueden seguir aportando información.

Entonces, retomando las fórmulas, ahora que sabemos cómo representar un documento:

$$P(d | c) = P(x_1, x_2, x_3, \dots, x_n | c)$$

Como la probabilidad de aparición de una palabra es independiente a que aparezca otra palabra. Por las leyes matemáticas, puedo convertir la probabilidad de varias variables independientes en el producto de diferentes probabilidades.

$$P(x_1, x_2, x_3, \dots, x_n \mid c) = P(x_1 \mid c) * P(x_2 \mid c) * P(x_3 \mid c) * \dots * P(x_n \mid c)$$

Entonces, reescribiendo la formula, la clase candidata C_{map} va a ser el máximo argumento de la probabilidad de c_j ($c_j \in C$) por el productor (\prod) de todas las probabilidades de cada una de las características condicionales (x_i con $i \in \text{Posiciones}$) para cada una de las c candidatas (c_j). Hago este cálculo j veces y cada vez i palabras.

$$C_{map} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in \text{Posiciones}} P(x_i \mid c_j)$$

¿Cómo calcular $P(x_i \mid c_j)$?

Para calcular la probabilidad condicional (solo los que aparecen en cierta clase) de una palabra dada una clase c , lo que se hace es contar la cantidad de veces que aparece la palabra en los documentos de clase c y lo divido por la cantidad total de palabras que tienen los documentos de clase c .

$$p(w_i|c) = \frac{\text{cantidad de veces que aparece } w_i \text{ en documentos en la clase } c}{\text{cantidad de palabras que aparecen en los documentos de la clase } c}$$

Asumiendo que hay una probabilidad real o perfecta de cada palabra, pero no conozco estos valores, pero los puedo estimar del conjunto de entrenamiento T .

$$p'(w_i|c) = \frac{\text{cantidad de veces que aparece } w_i \text{ en documentos en la clase } c \text{ en } T}{\text{cantidad de palabras que aparecen en los documentos de la clase } c \text{ en } T}$$

Finalmente, estas son las dos ecuaciones que tenemos que calcular para entrenar un clasificador Bayes Naïve:

$$p'(w_i | c_j) = \frac{\text{cantidad}(w_i | c_j)}{\sum_{w \in V} \text{cantidad}(w, c_j)} \quad V: \text{vocabulario (según T)}$$

$$p'(c_j) = \frac{\text{cantidad de documentos de clase } c_j \text{ en } T}{\text{cantidad de documentos totales en } T}$$

Laplace smoothing

Cuando aparecen palabras que nunca fueron contemplados y la probabilidad de aparición de estas son cero usando la clasificación de Bayes, Laplace smoothing o también conocido como Add-one resuelve este problema.

Laplace smoothing aplicado a Naïve Bayes:

Consiste en agregarle un **1** en el numerador a cada cantidad (w_i, c_j) calculada, para que nunca sea cero y en el peor de los casos sea uno, y también se le agrega **1** en el denominador para mantener la probabilidad normalizada para cada $w \in V$, o lo que es lo mismo sumar en el denominador la cantidad de palabras en el vocabulario.

$$p'(w_i | c_j) = \frac{\text{cantidad}(w_i | c_j) + 1}{\sum_{w \in V} (\text{cantidad}(w, c_j) + 1)}$$

$$p'(w_i | c_j) = \frac{\text{cantidad}(w_i | c_j) + 1}{\sum_{w \in V} \text{cantidad}(w, c_j) + |V|}$$

$w \in V$, w palabras que pertenecen a V vocabulario (conjunto de palabras con ocurrencias).

Redes bayesianas

Cuando se entrena un clasificador, lo que hace es modelizar el conocimiento. Las redes bayesianas son una forma de modelizado, y cuando se crea un clasificador con Bayes Naïve, lo que se genera una de estas redes Bayesianas.

- Grafo acíclico dirigido (forma de representación de las Redes Bayesianas).
- Los nodos representan variables.
- Las aristas representan dependencias condicionales.

Las redes bayesianas tienen asociada una tabla con probabilidades condicionales por cada nodo. Permiten realizar inferencias (preguntas), según la observación de un evento.

Si bien las redes bayesianas permiten inferencias mucho más precisas que la versión simplificada que construye Bayes Naïve, son más complejas de construir y de mantener.

Por otro lado, Bayes Naïve conjuga varias características positivas:

- Es muy rápido y requiere poco almacenamiento.
- Robusto ante características (palabras) irrelevantes.
- Muy bueno en dominios en donde hay muchas características y todas son importantes.

Además, si resulta que el supuesto sobre la independencia de las palabras es cierto, Naïve Bayes es óptimo.

Análisis de Sentimientos

El análisis de sentimientos es una tarea de clasificación de textos. Tiene múltiples usos y está aplicado a los valores objetivos, sentimientos que podemos encontrar en las diferentes frases, textos, etc.

Por ejemplo:

Críticas, Opiniones o Comentarios relacionados a libros, películas, productos, servicios etc.

- Decepcionante. **Negativo**
- Aburrida. **Negativo**
- Personajes memorables y bien desarrollados. **Positivo**
- Una gran respuesta en escena que no defraudará. **Positivo**
- Increíblemente predecible. **Negativo**

Además de poder clasificar críticas positivas o negativas, también se puede **estimar la confianza del consumidor**.

“La confianza del consumidor es un **indicador económico** que mide el grado de optimismo que los consumidores sienten sobre estado general de la economía y sobre su situación financiera personal. Qué tan seguras se sienten las personas sobre la estabilidad de sus ingresos determina sus **actividades de consumo** y por lo tanto sirve como uno de los indicadores claves en la forma general de la economía” (Wikipedia).

Predecir el mercado de valores, se usaron dos herramientas de Google que no estan liberadas:

- OpinionFinder: que mide opinión negativa versus opinión positiva.
- Google-Profile of Mood States (perfil google de estados de ánimo) que mide el humor en término de 6 dimensiones: calmado, alerta, seguro, vital, amable, feliz.

Sinónimos del nombre de “Análisis de sentimientos”:

- Extracción de opinión (opinion extraction).
- Minería de opiniones (opinion mining).
- Minería de sentimientos (sentiment mining).
- Análisis de subjetividad (subjectivity analysis).

Se puede usar este análisis en:

- Películas: ¿Es esta crítica positiva o negativa?
- Productos: ¿Qué piensa la gente del nuevo iPhone?
- Sentimientos públicos: ¿Cómo es la confianza del consumidor? ¿Crece de forma impar?
- Política: ¿Qué piensa la gente acerca de este candidato o de esta situación?
- Predicción: Predecir el resultado de una elección o una tendencia de mercado a partir de los sentimientos.

Tipología de Scherer de los estados afectivos

Scherer es un psicólogo que definió:

- **Emoción:** Respuesta relativamente corta del organismo a estímulos externos. Ejemplos de emoción son la **ira**, la **tristeza**, la **alegría**, el **miedo**, la **vergüenza**, el **orgullo** y la **desesperación**.
- **Estado de ánimo:** sentimientos subjetivos de baja intensidad y larga duración. Ejemplos de estados de ánimo: **alegre**, **triste**, **irritable**, **apático**.
- **Postura interpersonal:** posición afectiva respecto a otra persona en una interacción específica. Ejemplos de posturas interpersonales son: **distante**, **frío**, **de apoyo** y **de desprecio**.
- **Actitudes:** preferencia o predisposición de una persona respecto a otras personas u objetos. Ejemplo de actitudes son: **simpatía**, **amor**, **odio**, **deseo** y **valoración**.

- Rasgos de personalidad: tendencias en el comportamiento típico de una persona. Ejemplo de rasgos de personalidad: **nervioso, ansioso, imprudente, taciturno, hostil, envidioso y celoso**.

El análisis de sentimientos trabaja con las actitudes de las personas, es decir, se está detectando la “preferencia o predisposición de una persona respecto a otras personas u objetos”.

Luego se realizan tareas de **análisis más complejas** como:

- El portador de dicha actitud.
- El destinatario de dicha actitud.
- El tipo de actitud:
 - **Simpatía, amor, odio, deseo y valoración** (lista de candidatas).
 - O una polaridad ponderada: **positiva, negativa o neutral** (y a veces un valor asociado).
- Extraer porción de texto que contiene la actitud (documento u oración).

Resumen

- Una tarea sencilla de análisis de sentimientos consiste en:
 - Determinar si la actitud de un texto es positiva o negativa.
- Una tarea un poco más compleja de análisis de sentimientos consiste en:
 - Puntuar la actitud de un texto de 1 a 5.
- Una tarea realmente avanzada de análisis de sentimientos consiste en:
 - Detectar el portador, el destinatario y el tipo de actitud de un texto.

Algoritmo de Pang y Lee

Detección de la polaridad

- Tokenización del texto (dividir el texto en palabras).
- Extracción de características (palabras o frases claves).
- Clasificación utilizando distintos algoritmos de clasificación:
 - Naïve Bayes
 - MaxEnt
 - SVM

Problemas comunes a la hora de tokenizar:

- Lidar con los tags XML o HTML.
- Tener que reconocer las marcas de Twitter (si queremos sacar información de ahí) como los nombres de usuario y los hash tags.
- **El uso de mayúsculas.** Generalmente nos va a interesar conservar las mayúsculas de las palabras en las distintas fases del algoritmo.
- Número de teléfono y fechas.
- **Emoticones:** es muy útil detectar los emoticones cuando se está haciendo análisis de sentimientos.

En la etapa de **extracción de características** pueden surgir **dos problemas**;

- ¿Cómo lidar con la negación? (si elimino las Stop Words, podría cambiar completamente el significado de la oración).
 - **No** me gustó esta película.
 - Me gustó esta película.
- ¿Qué conviene usar?
 - Todas las palabras.
 - Solo los adjetivos.

Se demostró que al menos con la información de IMDB, es conveniente utilizar todas las palabras. Se obtienen así mejores resultados y en términos generales diría que siempre conviene utilizar todas las palabras ya que a veces los sustantivos y los verbos nos dan información valiosa sobre el juicio de valor de una crítica.

¿Cómo lidar con la negación?

Un método es reemplazar todas las palabras entre la negación y el siguiente signo de puntuación, por ejemplo:

No me gustó esta película, pero yo...

No **NO_me NO_gustó NO_esta NO_película**, pero yo...

De esta forma, creo nuevas palabras para esa crítica en específica y aumenta la cantidad de vocabulario que voy a tener.

Naïve Bayes multinominal binarizada (o booleana)

Variación del algoritmo de Bayes

Antes de comenzar a calcular las probabilidades de las clases y a contar las palabras, recorrer uno por uno, todos los documentos en el conjunto de entrenamiento y prueba **y eliminar las palabras duplicadas**.

En términos generales **esta variante del algoritmo da mejores resultados** que la versión tradicional que cuenta todas las ocurrencias de las palabras.

Problemas del algoritmo de Pang y Lee

No llega a detectar **sutilezas y expectativas frustradas**.

Ejemplo de sutileza:

“Si usted está leyendo esto es porque es **su fragancia favorita**, por favor **úsela exclusivamente en su casa y cierre bien las ventanas**.”

Ejemplo de expectativas frustradas:

“La película debería ser excelente ya que cuenta con grandes actores y una banda sonora fantástica, **sin embargo**, es terriblemente aburrida.”

Lexicón de sentimientos

Un lexicón significa diccionario. Por ejemplo, lexicón de sentimientos, que, en vez de tener definiciones de las palabras, tengo su valor emotivo, de sentimiento, clasificación de las palabras en “positivo” o “negativo”, también puede ser “Fuerte” vs. “Débil” o “Activa” vs. “Pasiva”.

El lexicón es un diccionario con palabras ya clasificadas, y existen muchos lexicones de todo tipo de clasificación, que son el resultado de un clasificador entrenado con un conjunto de datos mucho más amplio y con más variedad de datos. Esto puede facilitar la evaluación de cualquier texto, sin tener que entrenar un clasificador con un conjunto de datos reducido y puntual, haciendo que la clasificación sea poco genérica y sin tener que etiquetar a mano las cosas.

Lexicones:

- **The General Inquirer**
- **LIWC** (Linguistic Inquiry and Word Count)
- **MPQA subjectivity Cues Lexicón**
- **Bing Liu Opinion Lexicón**
- **SentiWordNet** (el más conocido y más usado)

Existe mucha diferencia entre SentiWordNet con los otros lexicones, se han considerado de forma diferente las palabras y tomado otras consideraciones y excepciones que afectan a la clasificación final.

Creación de un lexicón propio

Los diccionarios no siempre dan tan buenos resultados en comparación al clasificador que nosotros entrenamos con un conjunto de datos representativos del universo que a nosotros nos interesa medir.

El lexicón no es tan exacto, pero podemos crear uno propio o ampliarlo.

Se necesita:

- Un puñado de ejemplos previamente clasificados
- Algunas reglas escritas a mano que identifiquen ciertos patrones en una frase

Algoritmo de Hatzivassiloglou y McKeown para la ampliación de un lexicón

Adjetivos unidos por “y” tienen la misma polaridad (ambos tienen el mismo valor, ambos son negativos o positivos). Ej:

- Justo y legitimo
- Corrupto y brutal

Adjetivos unidos por “pero” tienen distinta polaridad (tienen valores inversos, uno positivo y el otro negativo). Ej:

- Justo pero brutal
- Corrupto, pero legitimo
- Hermosa pero malvada

Teniendo esta idea en mente, idearon un **algoritmo en 4 pasos**:

- Construyeron un Lexicón a mano con 1336 adjetivos:

- 657 positivos
- 679 negativos
- Buscaron en Google cada uno de los adjetivos con la formula: “**was <adjetivo> and**” y recolectaron la palabra que seguía a continuación. Luego lo repitieron con “**but**” en vez de “and”.
- Construyeron un mapa que vinculaban las palabras similares entre sí, mostrando también las que tenían sentido opuesto.
- Finalmente buscaron una forma de separar el mapa creado intentando que quede dos conjuntos bien diferenciados.

Si bien lograron ampliar considerablemente el Lexicón original, el nuevo Lexicón contenía algunos errores, es decir palabras mal catalogadas.

Es por ello que este algoritmo necesariamente necesita de un paso extra que consistía en la **revisión de los datos obtenidos**.

Algoritmo de Turney para obtener la polaridad de frases

Clasificación de **frases** positivas o negativas

Este algoritmo se puede desglosar en 3 pasos principales:

- Extraer frases de opiniones/criticas (reviews) y armar un Lexicón de frases.
- Aprender la polaridad de cada frase.
- Puntuar las críticas según el promedio de las polaridades de sus frases.

Las frases que extrajo Turney respetaban estas reglas:

Primer Palabra	Segunda Palabra	Tercer Palabra (no se extrajo)
Adjetivo	Sustantivo (plural o singular)	Cualquier palabra
Adverbio	Adjetivo	No Sustantivo
Adjetivo	Adjetivo	No Sustantivo
Sustantivo (plural o singular)	Adjetivo	No Sustantivo
Adverbio	Verbos	Cualquier palabra

Para verificar la polaridad de las frases, se verificó cuan cerca aparecían estas de palabras con polaridad ya conocida como, por ejemplo: “**excelente**” y “**pobre**”.

La idea detrás de esto es que la **co-ocurrencia** de una frase junto con la palabra

“excelente” o bien con la palabra “pobre” **no es casualidad**, sino que la frase misma tiene una carga de valor, una polaridad.

Para saber si realmente una frase ocurre cerca de una palabra positiva o negativa de forma independiente, aleatoria o tiene cierta dependencia entre la frase y la palabra, utilizo una forma matemática llamada PMI (**Pointwise mutual information**).

El **PMI** de un par de valores **x** e **y** (frase y palabra) pertenecientes a dos variables aleatorias discretas: **X** e **Y** respectivamente, cuantifica la discrepancia entre la probabilidad de su coincidencia dada su distribución conjunta y su distribución individual y asumiendo su independencia.

Matemáticamente:

$$PMI(X,Y) = \text{Log}_2 \frac{P(x,y)}{P(x)P(y)}$$

P(x,y): probabilidad de ocurrencia conjunta de la frase y la palabra (x e y).

P(x): probabilidad de ocurrencia de frase.

P(y): probabilidad de ocurrencia de palabra.

Cuando son independientes la frase de la palabra, **P(x,y)** es exactamente la misma que **P(x)** multiplicado por **P(y)**.

En otras palabras: cuanto más posible es que el evento **X** aparezca vinculado al evento **Y** a que aparezcan ambos de forma independiente entre sí.

La **probabilidad de la palabra P(x)** es:

$$P(\text{palabra}) = \frac{\text{cantidad de resultados para "palabra"}}{\text{cantidad total}}$$

$$P(\text{palabra}) = \frac{\# \text{palabra}}{N}$$

La **probabilidad conjunta P(x,y)** es:

$$P(\text{palabra1}, \text{palabra2}) = \frac{\text{cantidad de resultados para "palabra1 NEAR palabra2"}}{(\text{cantidad total})^2}$$

$$P(\text{palabra1}, \text{palabra2}) = \frac{\#(\text{palabra1 NEAR palabra2})}{N^2}$$

NEAR significa “cerca de”, indica que **palabra1** (o frase) apareció a no más de **N** palabras de distancia de **palabra2** en un texto dado. **NEAR** es un número arbitrario y se asume que es chico el número (1 o 2), es la cantidad de palabras en dos palabras seleccionadas.

La **Pointwise mutual information** es:

$$PMI(\text{palabra1}, \text{palabra2}) = \log_2 \frac{\frac{\#(\text{palabra1 NEAR palabra2})}{N^2}}{\frac{\#(\text{palabra1}) \#(\text{palabra2})}{N * N}}$$

Los denominadores se cancelan y queda:

$$PMI(\text{palabra1}, \text{palabra2}) = \log_2 \left(\frac{\#(\text{palabra1 NEAR palabra2})}{\#(\text{palabra1}) \#(\text{palabra2})} \right)$$

Calculando la polaridad de una frase (ejemplo):

$$\text{Polaridad}(\text{frase}) = PMI(\text{frase}, \text{"excelente"}) - PMI(\text{frase}, \text{"pobre"})$$

En resumen:

$$\text{Polaridad}(\text{frase}) = \log_2 \frac{\#(\text{frase NEAR excelente}) \#(\text{pobre})}{\#(\text{frase NEAR pobre}) \#(\text{excelente})}$$

(Los signos de puntuación cuentan como separación y no cuenta como NEAR)

Armar una lista de frases y asociarles el valor obtenido con los cálculos anteriores para luego descomponer las críticas en sus frases y realizar el promedio.

Conclusión

Trabajo Turney sobre 410 opiniones de Epinions y la exactitud promedio fue del 74%:

- Críticas cinematográficas: 66%.
- Críticas sobre bancos y automóviles: 80% y el 84%.
- Críticas sobre viajes; intermedio.

Aspectos

En una crítica puede haber más de un sentimiento en la misma frase, que son aspectos, independientemente uno del otro, por ejemplo:

¡La comida era excelente pero el servicio pésimo!

En una crítica puede haber varios aspectos y eso puede dificultar la clasificación de la crítica en general, ya que, cada aspecto tiene una polaridad diferente. Entonces para diferenciar la polaridad de cada una es necesario descomponer el texto y analizarlos para entender de lo que se habla en la crítica.

Método de Minqing Hu y Bing Liu

- Frecuencia
- Regla

Frecuencia: Buscaron todas las **frases frecuentes** en las críticas de un lugar dado. Luego buscaron si a continuación de la frase frecuente aparecía algún elemento o palabra que tuviese un juicio de valor.

Por ejemplo, para un mismo restaurante encontraron que se repetía muchas veces la frase: **“tacos de pescado”**. A este tipo de frases las llamaron **aspectos, atributos** o bien **objetos de sentimiento** ya que representan el objeto al cual se está criticando.

Reglas: Filtraron todas esas frases frecuentes con algunas reglas como: “ocurre después de una palabra que indica sentimientos”.

Ejemplo: **“geniales tacos de pescado!”** => indica que “tacos de pescado” es muy probablemente un aspecto.

Consideraciones finales

- El aspecto puede no ser mencionado en una sentencia.
- Para hoteles/restaurantes los aspectos son generalmente conocidos y fáciles de identificar.
- Es posible utilizar una clasificación supervisada:

- Para pequeños corpus, se puede utilizar una clasificación manual de aspectos: comida, decoración, servicio, precio, **nada**.
- Y luego entrenar un clasificador:
 - Dada una sentencia, tiene alguno de estos aspectos: “comida, decoración, servicio, precio, nada”.
- Si la cantidad de críticas no está balanceada (entre positivas y negativas o entre los rangos elegidos)
 - No se puede utilizar estimador: precisión.
 - Hay que utilizar F-score visto anteriormente.
- Si el desbalanceo es muy pronunciado se puede degradar severamente el rendimiento del clasificador
 - Dos soluciones comunes:
 - Tomar un muestreo parejo.
 - Penalizar más severamente al clasificador por un error al categorizar la clase más rara.

Aspecto de **grano grueso**. Ejemplo de un restaurante: ubicación, precio , servicio, comida, etc.

Aspecto de **grano fino**. Ejemplo de un restaurante: taco de pescado, milanesa, etc.

Extracción de Información

El objetivo de la extracción de información es capturar ciertas **partes relevantes** de un **texto**. Muchas veces en el contexto de varios documentos distintos, y generar luego, con dicha información, una representación estructurada, limpia y legible, como podría ser una tupla en una base de datos relacional.

Información fáctica

¿Quién hizo qué a quién y cuándo?

Ejemplo:

*“Las oficinas de **Google** en la **Argentina** ya tienen su historia. La empresa abrió su filial local en **2008** en **Puerto Madero**. Allí trabajan 215 empleados en 6000 m2 que ocupan las instalaciones.”*

Una manera sensilla de parsear:

- SEDE(“Google_Argentina”, “Puerto Madero”).

- APERTURA_SEDE("Google_Argentina", "2008").

SEDE y APERTURA_SEDE son un tipo de relación semántica entre dos entidades. Esta es una forma de extracción de información de forma estructurada.

Métodos supervisados y auto-supervisados

Supervisados: Requieren un conjunto de datos previamente etiquetados. Requieren tiempo, esfuerzo y una intervención humana importante.

Auto-Supervisados: Aprenden a etiquetar y generar su propio conjunto de entrenamiento. Son escalables.

Reconocimiento de nombres de entidades (NER)

Para realizar extracción de información, lo primero que hay que hacer es el NER.

Ejemplo:

"Ready Player One es un libro de ciencia ficción. Es la primera novela de Ernest Cline. El libro fue publicado en España en noviembre de 2011 por Ediciones B. Es el año 2044 y el mundo es un desastre. Las fuentes de energía fósiles están prácticamente agotadas y el precio del combustible está por las nubes. En medio de una enorme depresión a nivel mundial la mayoría de la gentes subsiste como puede. Sin embargo, un videojuego de realidad virtual llamado OASIS proporciona..."

Luego de detectar y extraer los nombres de las entidades, identificarlas:

- Ready Player One => Libro
- Ernest Cline. => Persona
- España => Lugar, país
- 2011 => Fecha, año
- Ediciones B. => Empresa, editorial
- 2044 => Fecha, año
- OASIS => OTRO (Video Juego de ficción)

La identificación de las entidades, sirve para:

- Índices o enlaces a contenidos relacionados.
- Destinatario de los sentimientos en Sentiment Analysis.
- Extracción de información.

- Preguntas y respuestas (question answering).

1er paso: Tokenización. Luego se etiquetan las palabras dependiendo de la clase que pertenecen (clasificados por los colores).

- Ready => Libro
- Player => Libro
- One => Libro
- Ernest => Persona
- Cline. => Persona
- España => Lugar, país
- 2011 => Fecha, año

Modelos de etiquetamiento secuencial para el reconocimiento de nombres de entidades (NER)

Pasos del entrenamiento

- Conseguir un conjunto de documentos representativos de nuestro dominio.
- Etiquetar cada palabra (token) con la clase que le corresponde (persona, organización, etc.) o bien marcarla con la etiqueta: “otra”.
- Especificar características de extracción que se adecuen a las clases y el texto que tenemos.
- Entrenar un clasificador secuencial para predecir las etiquetas del conjunto de prueba. Un clasificador que tenga en cuenta la secuencia de las palabras y no solo tomar como una bolsa de palabras.

El IO – encoding detecta y etiqueta la clase del nombre.

El IOB – encoding además de detectar y etiquetar la clase, también detecta el comienzo y fin del nombre (ej.: si es una persona, etiqueta como B-PER como comienzo e I-PER como final del nombre).

Identificación de características

Basadas en las palabras

- Palabra **actual**. Ej.:
 - Pertenecer al diccionario (lexicón).
- Palabra **previa o siguiente**. Ej.:
 - “a las ...” => hora.

- “en ...” => lugar.
- “at ...” => lugar (inglés).
- **Substring** de una palabra. Ej.:
 - oxa (**substring**) => Drogas (**categoría**) => Cotrimoxazole (**ejemplo**).
 - field (**substring**) => Lugares (**categoría**) => Banfield (**ejemplo**).
 - : (**substring**) => Películas (**categoría**) => 2001: Odisea del espacio (**ejemplo**).
- Forma de una palabra. Ej.:
 - Xx-xxx (**Forma**) => Varicela -zóster (**Nombre de entidad**).
 - xXXX (**Forma**) => mRNA (**Nombre de entidad**).
 - XXXd (**Forma**) => CPA1 (**Nombre de entidad**).

Basada en otro tipo de inferencia lingüística

- Etiquetado gramatical. Ej.:
 - **Adjetivo + Sustantivo + Sustantivo => Nombre de entidad**

Contexto de etiquetado

- Etiqueta anterior y siguiente. Ej.:
 - **PERSONA + PERSONA + ???? => Posiblemente sea PERSONA** (Por ejemplo: Juan(P) Carlos(P) Perez(?)). Frente a una palabra que no está en mi diccionario, siguiendo el contexto, hay una alta probabilidad de que la palabra sea un nombre de persona en este caso del ejemplo.

No puedo usar Bayes porque usa bolsa de palabras y no tiene en cuenta el orden y ninguna otra característica de las palabras.

Algoritmos de inferencia

Clasificadores que sí tienen en cuenta el orden de las palabras y otras características que permiten entrenar un método supervisado para detectar nombres de entidades. Son algoritmos dedicados a la extracción de nombres de entidades:

- **Greedy Inference**
- **Beam Inference**
- **Viterbi Inference**
- **CRFs**

Extracción de relaciones semánticas

Este es el segundo paso luego de la extracción de nombres de entidades.

Ontología

Una ontología representa el conocimiento a través del grafo, donde hay conceptos en los nodos y están unidos por las aristas que los relacionan

- **Is-a (hipónimo):** Ej.: Jirafa es un rumiante es un mamífero es un vertebrado es un animal. Construye una **escala jerárquica** a partir de la relación “**es un**” o “**is-a**” en inglés.
- **Instance-of:** Ej.: Buenos Aires es una instancia de ciudad. Aquí no importa el orden jerárquico, sino que es una instancia concreta de una categoría abstracta.

Cuando la ontología está creada, uno puede hacer preguntas, y se obtiene respuesta de esta.

¿Cómo contruir una ontología?

- **Reglas escritas a mano**, de tipo **pattern-matching**. (X e Y son los términos que quiero extraer).
 - Y como X (,X*) (,and | or) X).
 - Tanto Y como X.
 - X o otra Y.
 - X y otra Y.
 - Y incluyendo X.
 - Y, especialmente X.

Ventajas: Tienden a tener alta precisión y pueden ser adaptados a dominios específicos.

Desventajas: Pero suelen tener muy bajo recall (exhaustividad), implica una gran cantidad de trabajo pensar en todos los patrones posibles y más aún para todas las relaciones. Se puede mejorar la precisión con otros métodos.

- **Aprendizaje automático supervisado.** Buscar dos nombres de entidades, generalmente en la misma oración, decidir si están o no relacionadas, si lo están, clasificar la relación. Pasos:
 - Decidir qué relaciones nos interesa extraer (ej.: extraer relaciones “is-a”).
 - Decidir qué **nombres de entidades** son **pertinentes** a dichas relaciones.
 - Encontrar un conjunto de datos propicios (**corpus**):

- Etiquetar las entidades detectadas en el corpus.
- **Etiquetar a mano las relaciones** entre esas dos entidades.
- Partir este conjunto de datos en entrenamiento de prueba.
- Entrenar un clasificador sobre el conjunto de entrenamiento.

En el **Aprendizaje automático supervisado**, luego de haber etiquetado las palabras seleccionadas, uso Bayes Naïve, con el modelo de Bag of Word y quitando los Stop Word. Además de usar las palabras como características, también uso entidades y análisis sintáctico.



Por ejemplo, entre las características adicionales que le proveyo a Bayes para mejora su clasificación:

Mención1: American Airlines y **Mención2:** Tim Wagner

- **Bigramas solo de las menciones:** {American; Airlines; Tim; Wagner; "American Airlines"; "Tim Wagner"}.
- **Agregar:**
 - Mención2 -1: "Spokeman" (agregar la palabra anterior).
 - Mención2 +1: "Said" (agregar la palabra posterior).
- Bolsa de palabras entre las entidades: {unit, AMR, immediately, move, spokeman, said}.

Características adicionales relacionadas con las entidades detectadas:

- Tipos de entidades:
 - Mención1: Organización.
 - Mención2: Persona.
- Concatenación entre ambas: **Organización-Persona**. Esta concatenación se lo paso a Bayes dentro de la bolsa de palabras. Antes uso a NER para detectar las entidades y le paso a Bayes las entidades y la concatenación de ellas según el orden.

Características relacionadas con análisis sintáctico:

- Utilización de la **categoría gramatical** de cada palabra o las secuencias de palabras, por ejemplo: NP, NP, PP, VP, NP, NP.

- Utilización de **path** (camino) de cada constituyente del **árbol sintáctico**.
Ej.: NP↑NP↑S↓S↓NP.
- Utilización del **árbol de dependencias**. Indica como una parte de la oración depende de otra, por ejemplo: Airlines matched Wagner said.

NP (noun phrase): **frase nominal**.

PP (prepositional phrase): **frase preposicional**.

VP (verbal phrase): **frase verbal**.

Clasificadores que se pueden utilizar: **MaxEnt**, **Bayes Naïve**, **SVM**.

- **Auto-supervisado**. La característica principal de este método es que el algoritmo va a etiquetar solo las relaciones semánticas entre entidades.

Algoritmo Bootstrap

- Para un par de “entidades” semilla cuya relación es conocida buscar coincidencias.
- Extraer el contexto de la oración, en partículas las “entidades” y reemplazarlas por *comodines*.
- Realizar búsquedas con esos patrones para encontrar nuevas entidades y repetir.

Por ejemplo, quiero encontrar la **relación** de **persona** y **lugar** en donde esta **enterrado**: **PERSONA – LUGAR**. Encuentro un patrón y reemplazo las entidades por *comodines* (**X** e **Y**).

- **Jorge Luis Borges** está enterrado en **Ginebra**, Suiza.
 - **Patrón**: **X** está enterrado en **Y**.
- La tumba de **Borges** está en **Ginebra**.
 - **Patrón**: la tumba de **X** está en **Y**.
- **Ginebra** es el lugar de descanso final de **Borges**.
 - **Patrón**: **X** es el lugar de descanso final de **Y**.

Este método, cuando se obtienen los resultados de búsqueda de posibles patrones, puede ser que a veces no se pueda saber cuando cortar la frase y que palabras incluir o excluir del patrón.

Una vez que obtengo los patrones reemplazados, busco en base a esos patrones y voy a obtener nuevos nombres de autores y nuevos lugares de entierro.

Algoritmo de Dripe

En este caso se quiere encontrar dos entidades que son autores y libros. Partió de 5 semillas:

- Issac Asimov => The Robots of Dawn.
- David Brin => Startide Rising.
- James Gleick => Chaos: making a new science.
- Charles Dickens => Great Expectations.
- William Shakespeare => The Comedi of Errors.

Busco en internet y encuentro **patrones**. Ej.:

- The Comedy of Errors, **by** William Shakespeare, was.
- The Comedy of Errors, **by** William Shakespeare, is.
- The Comedy of Errors, **one of** William Shakespeare's earliest attempts.
- The Comedy of Errors, **one of** William Shakespeare's most.

Si utilizamos solo la parte común, los patrones quedarían:

- **X**, by **Y**,
- **X**, one of **Y's**

Se buscan varias semillas para poder ver cuando se repiten los patrones y **saber las partes si y las que no pertenecen al patrón**.

Algoritmo Distan Supervision

Combina **Boostrapping** con **aprendizaje supervisado**.

En vez de usar solo 5 semillas, utiliza una gran base de datos para obtener una enorme cantidad de entidades-semillas.

Con los patrones que obtiene, extrae las características como se vio en el punto anterior. Y luego entrenar un clasificador.

- En común con los **supervisados**.
 - Usa un clasificador con varias características.
 - No requiere iterar N veces para esextraer los patrones.
- En común con los **no-supervisados**.
 - Usa grandes cantidades de datos sin etiquetar.
 - No es sensible a cómo se generó el corpus.

Detalle del algoritmo

- **Para cada realción**, por ejemplo: “nació-en”.
 - **Para cada tupla en una gran base de datos**: <Borges, Buenos Aires>, <Albert Einstein, Ulm>, <Gabriel García Márquez, Aracataca>...
 - **Ecuentra sentencias en un gran corpus**, en donde aparecen estas entidades semillas: Borges nació Buenos Aires; Einstein, quien nació en 1879 en Ulm; Gabriel García Márquez: lugar de nacimiento Aracataca, Colombia...
 - **Extraer las características frecuentes**. PERSONA nació en Lugar, ...
 - **Entrenar un clasificador**, utilizando cientos de patrones. $P(\text{“nació-en”} | p_1, p_2, \dots, p_{4000})$.
- **No supervisado** para la web o **Open Information Extraction (OIE)**.

Algoritmo KnowItAll (2005), no es exactamente OIE.

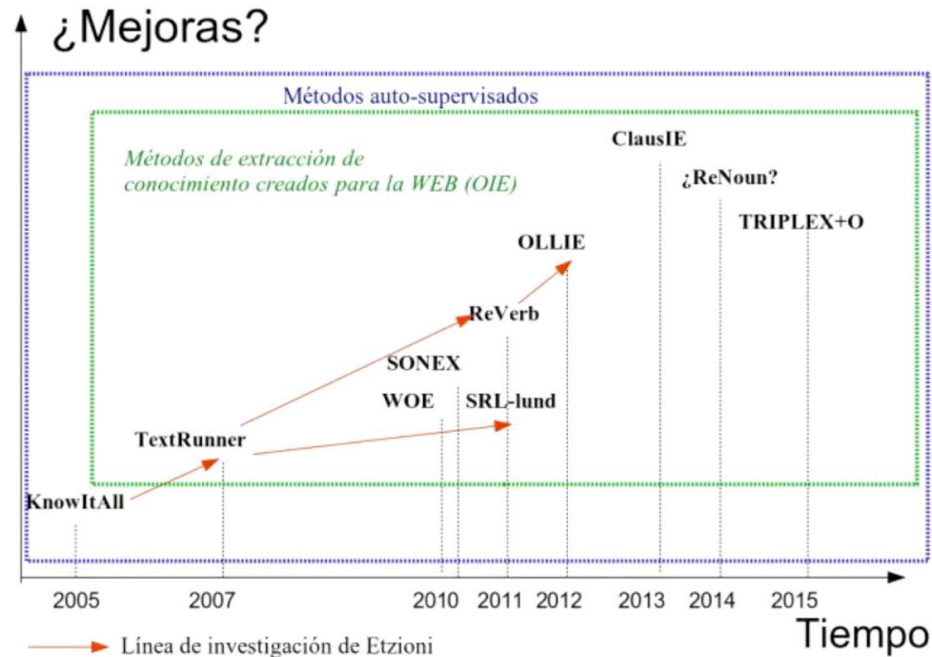
Principales características de Open Information Extraction:

- **No supervisado.**
- **Independiente del dominio.**
- **Escalable.**

Algoritmo TextRunner (2007), primer OIE.

- Realiza una sola pasada sobre el conjunto de datos como ya se mencionó.
- Extrae relaciones semánticas no definidas a priori. Puede extraer “cualquier cosa” que considere que es un relación entre dos entidades.

Evolución de los métodos de OIE:



Preprocesamiento Y Transformación de datos

Algunas tareas de estas etapas son:

- **Integración de datos:** Integración de múltiples bases de datos, archivos, etc. Crear tu propio dataset.
- **Limpieza de datos:** Completar valores faltantes, eliminación de ruido, identificar o eliminar valores atípicos y corregir incoherencias.
- **Reducción de datos:** Reducción de dimensionalidad, Reducción de Numerosidad. Tanto reducción de columnas, como de filas.
- **Transformación de datos:** Normalizaciones, generación de jerarquías conceptuales, etc. (Feature Engineering).

Limpieza de datos

Datos faltantes y sus tipos:

¡No sólo Nans!

Existen diferentes mecanismos de faltantes, los estándares son:

- **Missing completely at random MCAR**
 - En este caso la razón de la falta de datos es ajena a los datos mismo. No existen relaciones con la variable misma donde se encuentra los datos faltantes, o con las restantes variables en el dataset que expliquen por qué faltan.
- **Missing Not At Random MNAR**
 - La razón por la cual faltan los datos depende precisamente de los mismos datos que hemos recolectado (está relacionado con la razón por la cual falta). Ej.: Cada vez que una variable debería tener un valor entre 10 y 20, el mismo no se encuentra registrado (independientemente de los valores que tomen las variables restantes).
- **Missing At Random MAR**
 - Punto intermedio entre los dos anteriores. La causa de los datos faltantes no depende de estos mismos datos faltantes, pero puede estar relacionada con otras variables del dataset. Por ejemplo: encuestas mal diseñadas.

Estrategías para trabajar con datos faltantes

Eliminar registros o variables: Si la eliminación de un subconjunto disminuye significativamente la utilidad de los datos, la eliminación del caso puede no ser efectiva.

Imputar datos: En vez de estar eliminando datos, se podría intentar utilizar métodos de relleno de faltantes. Imputaciones puntuales, de una variable.

- **Sustitución de casos:** Se reemplaza con valores no observados. Debería ser realizado por un experto en esos datos.
- **Sustitución por Media o Mediana:** Se reemplaza utilizando la media calculada de los valores presentes. **Algunas desventajas:**
 - La varianza estimada de la nueva variable no es válida porque está atenuada por los valores repetidos.
 - Se distorsiona la distribución.
 - Las correlaciones que se observen estarán deprimidas debido a la repetición de un solo valor constante.
- **Imputación Cold Deck:** Selecciona o usa relaciones obtenidas de fuente distintas de la base de datos actual. Buscar y completar los datos faltantes en función a los datos que tengo.
- **Imputación Hot Deck:** Se reemplazan los datos faltantes con valores obtenidos de registro que son los más similares.
- **Imputación por regresión:** El dato faltante es reemplazado con el valor predicho

por un modelo de regresión. Se hace la predicción en función a los datos que tengo.

MICE – Multivariate Imputational by Chained Equations.

Imputar variables de forma **multivariada**, muchos variables, pero de a una a la vez. Trabaja bajo el supuesto de que el origen de los faltantes es Missing At Random (MAR).

Es un proceso de imputación de datos faltantes iterativo, en el cual, en cada iteración cada valor faltante de cada variable se predice en función de las variables restantes. Esta iteración se repite hasta que se encuentre convergencias en los valores. Por lo general 10 iteraciones es suficiente. (**En cada iteración genera un dataset**).

A	B	C
0.93	1.40	1.53
0.24	0.46	0.76
	0.80	
0.95	1.24	1.46
0.23	0.57	
0.90		1.28
0.15	0.42	
0.47	0.54	0.63
	1.14	
0.89	1.23	1.45

A	B	C
0.93	1.40	1.53
0.24	0.46	0.76
	0.80	
0.95	1.24	1.46
0.23	0.57	
0.90		1.28
0.15	0.42	
0.47	0.54	0.63
	1.14	
0.89	1.23	1.45

A	B	C
0.93	1.40	1.53
0.24	0.46	0.76
	0.80	
0.95	1.24	1.46
0.23	0.57	
0.90		1.28
0.15	0.42	
0.47	0.54	0.63
	1.14	
0.89	1.23	1.45

A	B	C
0.93	1.40	1.53
0.24	0.46	0.76
	0.80	
0.95	1.24	1.46
0.23	0.57	
0.90		1.28
0.15	0.42	
0.47	0.54	0.63
	1.14	
0.89	1.23	1.45

El método inicia el dataset original, con los datos faltantes

En cada dato faltante se imputa un valor utilizando por ej alguno de los métodos vistos antes

Con las variables B y C, se genera un modelo para redecir los faltantes originales en la variable A

Con las variables A y C, se genera un modelo para redecir los faltantes originales en la variable B

Continúa...

De la forma ilustrada en la imagen, se iteran varias veces hasta converger a la forma final a la que se quiere llegar. Todo esto asumiendo que las variables estan relacionadas y vinculadas de alguna forma para poder realizar las predicciones.

Análisis de valores atípicos

Análisis de Outliers:

“Un outlier es una observación que se desvía tanto de las otras observciones como para despertar sospechas que fue generado por un mecanismo diferente”. D.Hawkins. Identification of Outlier (1980).

- Es un concepto subjetivo al problem.

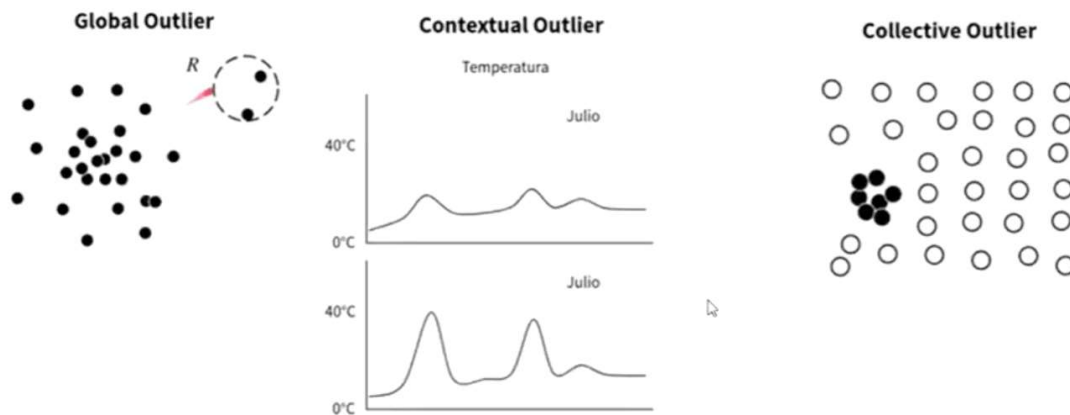
- Son observaciones distantes del resto de los datos.
- Pueden deberse a un error de medición, aleatoriedad, que esa instancia pertenezca a una familia distinta del resto, etc.

Ejemplo 1: Un set de datos con información de personas y una persona muestra 120 años de edad. O una persona de 4 años que mide 1,80 mts.

La detección de los outliers es importante su presencia puede influenciar los resultados de un análisis estadísticos clásico.

¿Es necesario eliminarlos?

- Deben ser **cuidadosamente inspeccionados**.
- Pueden estar alertando anomalías, en algunas situaciones nuestra tarea de interés será encontrarlos. Ej.:
 - Detección de Fraudes.
 - Detección de Fallas.
 - Patologías Médicas.



Global Outlier: Son los datos que se alejan de todo el conjunto de puntos.

Contextual Outlier: Son casos atípicos que dependen del contexto, por ejemplo: como muestra en el gráfico, en el mismo mes hay una gran diferencia de temperatura, puede ser atípico.

Collective Outlier: Son conjuntos de observaciones que se comportan de forma anómala colectivamente como se ve en el gráfico anterior.

Otra forma de distinguir los outliers:

Univariado:

- Son valores **atípicos** que podemos encontrar en una simple variable.
- El problema de los enfoques univariados es que son buenos para detección de extremos, pero no en otros casos.

Multivariado:

- Los valores **atípicos** multivariados se pueden encontrar en un espacio n-dimensional o mas de una variable.
- Para detectar valores atípicos en espacios n-dimensionales es necesario ajustar un modelo.

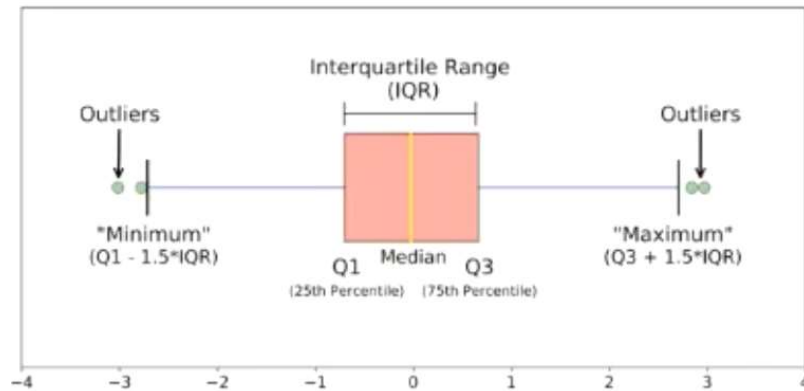
En grandes volúmenes de datos la detección de outliers resulta más eficiente estudiando todas las variables.

Los outliers, en **caso multivariados**, pueden provocar dos tipos de efectos:

- El **efecto de enmascaramiento** se produce cuando un grupo de outliers esconden a otro/s. Es decir, los outliers enmascarados se harán visibles cuando se elimine/n el o los outliers que los esconden.
- El **efecto de inundación** ocurre cuando una observación sólo es outlier en presencia de otra/s observación/es. Si se quitara/n la/s última/s, la primera dejaría de ser outlier.

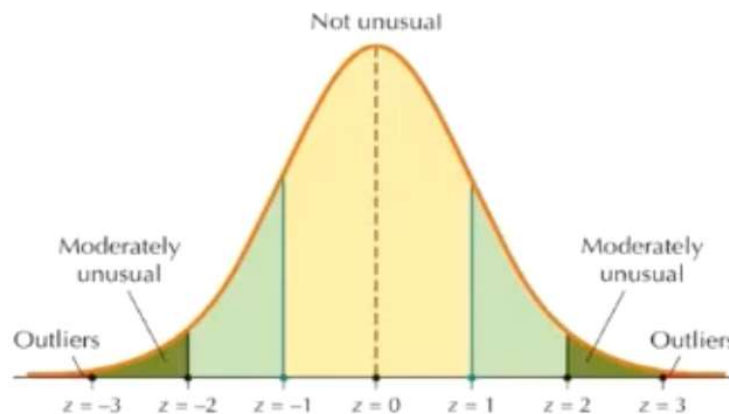
Métodos Univariados (detección de outliers)

- **IQR** (rango intercuartil): Analizar los valores que nos dan los Boxplot, que están por fuera del IQR.



- Z-score y Z-score Modificado.

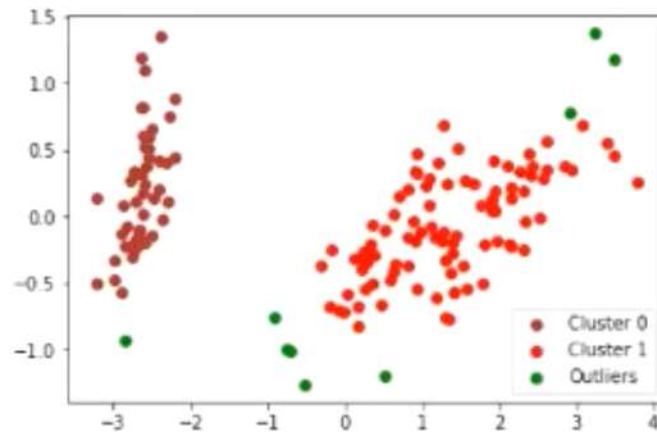
Detecting Outliers with z-Scores



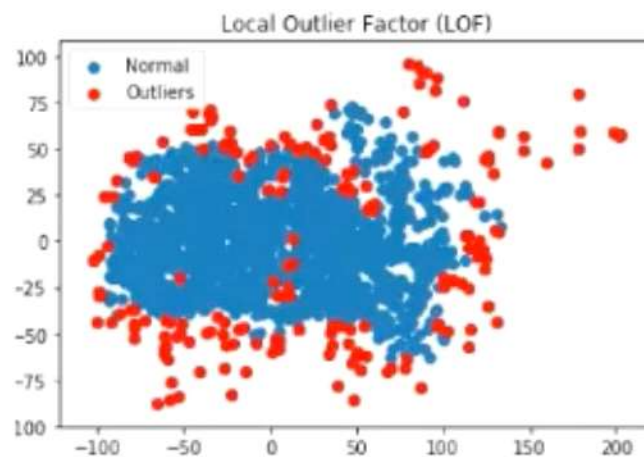
- Identificar valores extremos a partir de 1, 2 o 3 desvíos de la media.

Métodos Multivariados (detección de outliers)

- Análisis globales: **Clustering**.
 - Utilizando medidas de distancia como **Mhalanobis**. Los valores similares son agrupados y los que quedan aislados pueden ser considerados outliers.



- Local Outlier Factor (**LOF**).
 - Es un método de detección de outliers basado en distancias.
 - Calcula un score de outlier a partir de una distancia que se normaliza por densidad.



- Métodos basados en árboles de búsqueda: **IsolationForest**.

Analizando cada método univariado

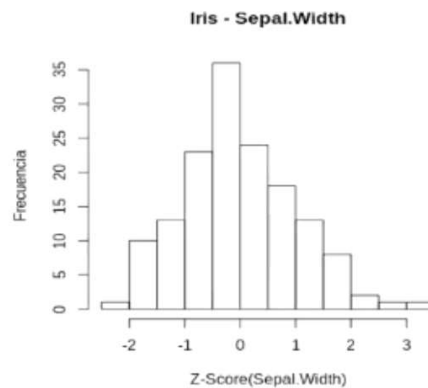
Z-Score

Es una **métrica** que indica **cuántas desviaciones estándares** tiene una **observación** de la **media muestral**, asumiendo una distribución gaussiana.

$$z_i = \frac{x_i - \mu}{\sigma}$$

Cuando calculamos Z-Score para cada muestra debemos fijar un umbral, un número de score limite que se considere anomalía una vez que se lo supera:

- Un valor como “regla de oro” es $Z > 3$. Es muy poco probable que Z-Score de mas o menos de 3.



Z-Score Modificado

La media de la muestra y la desviación estándar de la muestra, pueden verse afectados por los valores extremos presentes en los datos.

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD} \quad \text{Median Absolute Deviation} \quad MAD = \text{median}\{|x_i - \tilde{x}|\}$$

A cada una de las observaciones (x_i) le resto la mediana, lo divido por el **MAD** y multiplico todo por 0.6745.

Es la mediana de los desvíos absolutos respecto de la mediana.

Para hacer **MAD** comparable con la desviación estándar, se normaliza por **0.6745**.

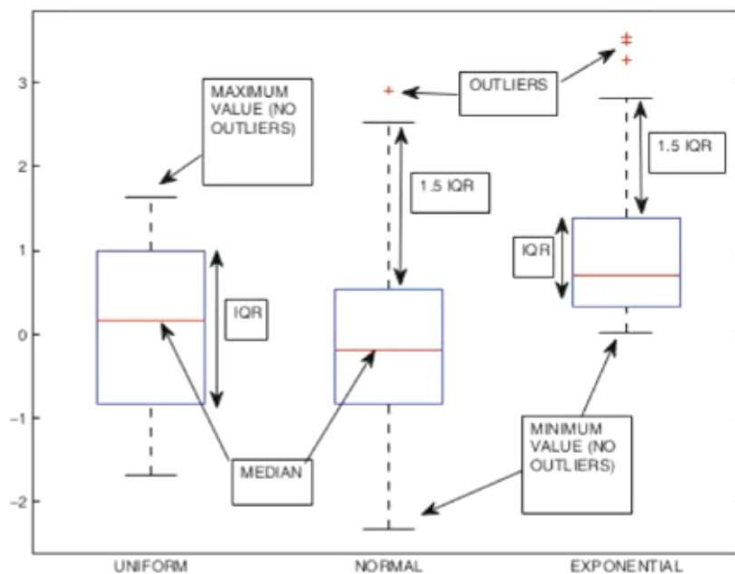
Regla de oro: Valores **mayores a 3.5** son considerados **outliers**.

Análisis de Box-Plot

Los Box-Plots permiten visualizar valores extremos univariados.

Las estadísticas de una distribución univariada se resumen en términos de cinco cantidades:

- **Mínimo/máximo** (bigotes).
- **Primer y tercer cuantil** (bordes de la caja).
- **Mediana** (línea media de la caja).
- **IQR** (rango intercuartil, altura de la caja) = $Q3 - Q1$.



Generalmente la regla de decisión:

- $\pm 1.5 \cdot \text{IQR}$ \Rightarrow Outliers **moderados**.
- $\pm 3 \cdot \text{IQR}$ \Rightarrow Outliers **severos**.

Problema

El análisis univariado solo detecta los valores extremos.

Una forma de tratar valores atípicos es eliminar los valores más altos y más bajos de una variable.

Esto puede funcionar bastante bien, pero no tiene en cuenta las combinaciones de variables.

Analizando cada método univariado

Distancia de Mahalanobis

La idea de mahalanobi es calcular la distancia de las observaciones a la nube de puntos a través de los siguientes pasos:

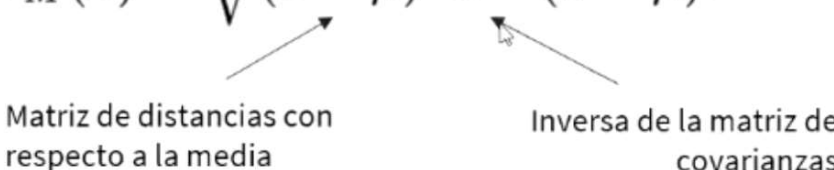
Es una medida de distancia entre el punto:

$$\vec{x} = (x_1, x_2, x_3, \dots, x_N)^T$$

Y un conjunto de observaciones con media:

$$\vec{\mu} = (\mu_1, \mu_2, \mu_3, \dots, \mu_N)^T$$

Y una matriz de covarianza S.

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$


Matriz de distancias con respecto a la media

Inversa de la matriz de covarianzas

LOF – Local Outlier Factor

El método LOF valora puntos en un conjunto de datos multivariados.

Es un método basado en densidad que utiliza la búsqueda de vecinos más cercanos.

- Se compara la densidad de cualquier punto de datos con la densidad de sus vecinos.
- Parámetro k (cantidad de vecinos) y métrica de distancia.

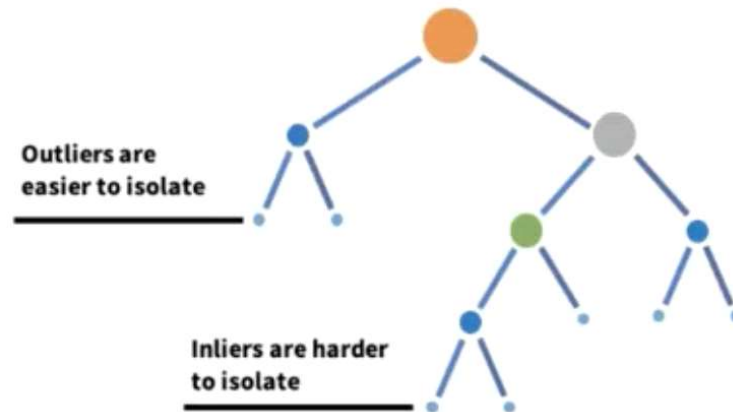
El método calcula los scores para cada uno, se debe definir un umbral de corte (depende del dominio).

- Si el score del X es 5, significa que la densidad promedio de los vecinos de X es 5 veces mayor que su densidad local.

Isolation Forest

Es un algoritmo no supervisado y no paramétrico basado en árboles de decisión.

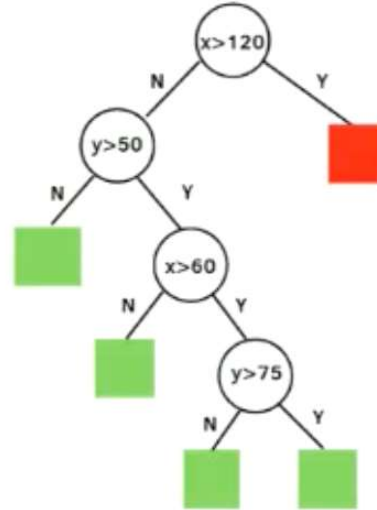
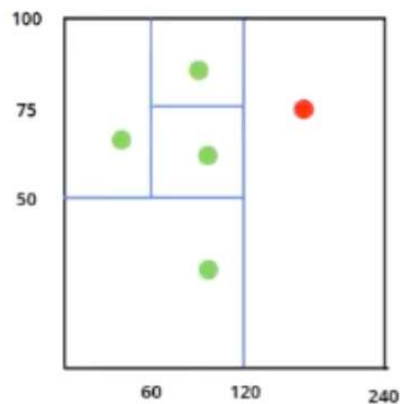
Idea principal: los datos anómalos se pueden aislar los datos normales mediante particiones recursivas del conjunto de datos.



Tomar una muestra de los datos y construir un árbol de aislamiento:

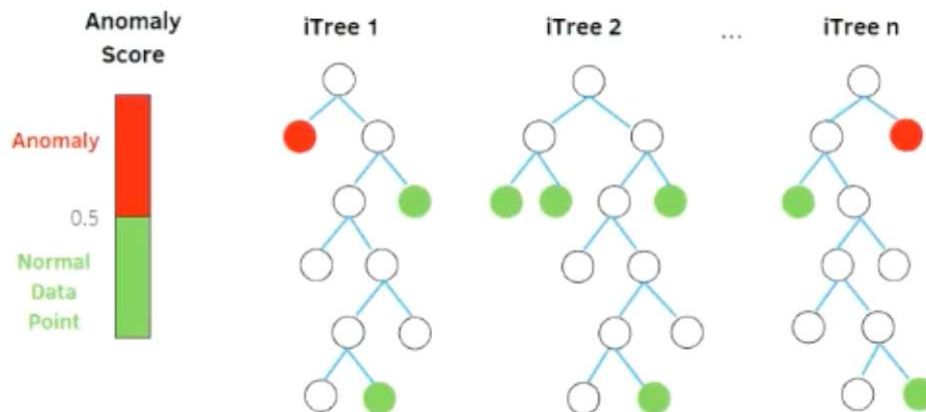
- Seleccionar aleatoriamente n características.
- Dividir los puntos de datos seleccionados aleatoriamente un valor entre el mínimo y el máximo de las características seleccionadas.

La partición de observación se repite recursivamente hasta que todas las observaciones estén aisladas.



Isolation Forest identifica anomalías como las observaciones con longitudes de ruta promedio cortas en los árboles de aislamiento.

- Utiliza la altura del árbol (cantidad de aristas).



De un conjunto de datos, suponiendo que se toma una muestra y se crea un árbol a partir de **dos o más** de las columnas aleatoriamente que tiene el dataset. Con el árbol creado aísla todos los puntos (que queden todos en las hojas del árbol). Una vez que termine con un árbol, creo otro, con otras columnas aleatoriamente y repito el procedimiento. Entonces, suponiendo que tenga un outlier global, casi cualquier atributo que use para hacer las particiones, la va a dejar afuera enseguida (va a estar muy cerca de la raíz).

Árboles

Modelo ID3

- **ID3**: Iterative Dichotomiser 3 (dicotomizador iterativo 3 -> tree -> árbol).
- Creado por: **Ross Quinlan**.
- Genera un árbol de decisión a partir de un conjunto de ejemplos.

(**Dicotomizador**: Método de clasificación que consiste en dividir en dos un concepto sucesivamente).

Representación de un **Árbol de decisión**. La salida del algoritmo ID3 se representa como un **grafo** en forma de árbol, cuyos componentes son:

- Un nodo principal llamado raíz en la parte superior
- Nodos terminales, como su nombre lo indica, son nodos donde termina el flujo y que ya no son raíz de ningún otro nodo. Estos nodos terminales deben contener una respuesta, o sea, la clasificación a que pertenece el objeto que ha conducido hasta él.
- Los demás nodos representan preguntas con respecto al valor de uno de los atributos.
- Las líneas representan las posibles respuestas que los atributos pueden tomar.

Entropía (de la información)

La medida del desorden o la medida de la pureza. Básicamente, es la medida de la impureza o aleatoriedad en los datos. Los árboles en particular ID3 trabaja con la entropía de la información.

Para calcular la entropía de **n clases** se utiliza la formula:

$$\text{Entropia}(S) = \sum_{i=1}^n -p_i \log_2 p_i$$

Dónde:

- **S**: es una lista de valores posibles.

- P_i : es la probabilidad de los valores.
- i : Cada uno de los valores.

Importante

- Para una muestra **homogénea** la entropía es igual a 0. Cuando la probabilidad es igual a 1, igual a una sola clase.
- La máxima entropía viene dada por $\log_2(n)$, n son los posibles valores de salida. Si $n=2$ (TRUE o FALSE) entonces, la máxima entropía es 1. O sea la máxima incertidumbre.

Ejemplo:

Imaginemos que queremos saber cuál es la entropía de Shannon de tirar un dado.

Estados posibles

$H(X) = 6 * \frac{1}{6} \log_2(6) = 2.58$

Ganancia de información:

La ganancia de información se aplica para cuantificar **qué característica**, de un conjunto de datos dados, **proporciona la máxima información** sobre la clasificación. Si tuviésemos que elegir una sola característica, para clasificar. ¿Cuál sería?

$$\text{Gan Inf}(S, A) = \text{Entropia}(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} \text{Entropia}(S_v)$$

A la entropía de S (conjunto) se le resta la entropía de la característica o valor. Se aplica la $\text{Gan Inf}(S, A)$ para todos los atributos (variables/columnas).

Dónde:

- S : Es una lista de valores posibles para un atributo dado: A .
- A : Uno de los atributos, en la lista de ejemplo.

- **V(A)**: Conjunto total de valores que A puede tomar.
- **Sv/S**: Probabilidad de un valor, para el atributo A.
- **Entropía (Sv)**, entropía calculada para el valor “v” de A.

Algoritmo básico:

- Calcular la **entropía** para todas las clases.
- Calcular la entropía para cada valor posible de cada atributo.
- Seleccionar el mejor atributo basado en la reducción de la entropía. Usando el cálculo de **Ganancia de Información**. Cuando encontremos la mayor Ganancia de información, ese va a ser, el primero de los nodos.
- **Iterar**, para cada sub-nodo. Excluyendo el nodo raíz, que ya fue usado.

La ganancia de información lo que me quiere decir, es que, cuanta información obtengo sabiendo solo un atributo. Si solo pudiera saber un valor de un atributo, con que tanta seguridad podría predecir una clase.

Recapitulando

- Un nodo de decisión está asociado a uno de los atributos y tiene 2 o más ramas que salen de él, cada una de ellas representando los posibles valores que puede tomar el atributo asociado.
- Un nodo-respuesta está asociado a la clasificación que se quiere proporcionar, y nos devuelve la decisión del árbol con respecto al ejemplo de entrada.

Impureza de Gini

La impureza de Gini es una medida de cuán a menudo un elemento elegido aleatoriamente del conjunto sería etiquetado incorrectamente si fue etiquetado de manera aleatoria de acuerdo a la distribución de las etiquetas en el subconjunto.

Algunas implementaciones de árboles de decisión utilizan la impureza de Gini en lugar de la ganancia de información, ya que es más fácil de calcular (computacionalmente menos costosa). **Ejemplo: Scikit-learn**

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

La clasificación 100% correcta de un atributo en sus nodos hoja, significa que es puro, sino se le dice que es impuro, que sus nodos son impuros. Cuando todos los árboles son impuros, para saber cual árbol clasifica mejor, se necesita medir la impureza de cada nodo y de cada árbol. Y la forma más fácil de medir la impureza es la de Gini.

Usando la formula de Gini, se calcula la impureza de cada hoja de cada árbol. Luego se calcula la impureza de Gini total para el nodo raíz, de cada árbol. La impureza total se trata del promedio ponderado (porque puede no estar balanceado el conjunto) entre los resultados de impureza de los nodos hoja. Para calcular el promedio de la impureza total, se hace la suma y la división por dos entre los resultados de los nodos hoja, pero al ser ponderado, cada impureza nodo hoja, se lo multiplica por la cantidad de casos de esa hoja dividido la suma total de todos los casos de todas las hojas.

Una vez que obtenga los resultados de la impureza total (promedio ponderado) de cada árbol, comparo entre ellas, y el número mas chico es el que menos impuro es y el que mejor clasifica. Entonces tomo ese atributo menos impuro y lo asigno como nodo raíz.

Luego se procede de igual forma que en el caso de la ganancia de información, pero calculando la impureza de Gini.

C4.5

Mejora del algoritmo ID3:

- Campos numéricos, rangos continuos.
 - Si un atributo A, tiene un rango continuo de valores. El algoritmo puede, dinámicamente crear un campo Booleano (partir el conjunto de datos en dos) tal que si $A < C$ $A_c = \text{TRUE}$, sino $A_c = \text{FALSE}$. Para encontrar ese umbral C, vamos a cortar el rango de forma que C nos quede con la mayor ganancia de información.
 - Ordenar A de menor a mayor, por ejemplo.
 - Identificar los valores adyacentes (de la clase de salida).
 - Detectar cuando hay un cambio de valor de salida, entonces en esos límites seguramente están los C_i candidatos.
 - Crear varios C_i , que dividen en dos el rango. Para cada uno de estos rangos calcular la ganancia de información.
 - Tomar el o los que dan mejor resultado. Puede haber N mejor.
- Datos faltantes. En ID3 no podía haber nulos.
 - Manejo de los datos de información con valores de atributos faltantes – C4.5 permite valores de los atributos para ser marcado como “?” para faltantes. Los valores faltantes de los atributos simplemente no se usan

en los cálculos de la ganancia de información y la entropía.

- Poda. Es una subrutina que corta ramas del árbol que no aportaban información. Este método consiste en:
 - Generar el árbol tal y como se vio anteriormente en ID3.
 - A continuación, analizar recursivamente, y desde las hojas, qué preguntas (nodos interiores) se pueden eliminar sin que se incremente el error de clasificación con el conjunto de test.
 - Se elimina un nodo interior cuyos sucesores son todos nodo hoja.
 - Se vuelve a calcular el error que se comete con esete nuevo árbol sobre el conjunto test.
 - Si este error es menor que el error anterior, entonces se elimina el nodo y todos sus sucesores (hojas).
 - Se repite.

(Si hay un **ruido** el árbol tendrá un error, es decir, cantidad de casos mal calificados, es la imprecisión en los datos de entrada, en el conjunto de entrenamiento. La poda lo que haces es subsanar ese error que contiene el árbol, eliminando los nodos con error para que este no lo aprenda).

Error = Casos bien clasificaddos / Casos Totales.

Random Forest

Bootstrap aggregating:

Es una técnica, o **meta-algoritmo** (esta un paso más arriba que el algoritmo) que dice lo siguiente:

Dado un conjunto de entrenamiento D , de tamaño n , la técnica de **bagging** generará m nuevos conjuntos de entrenamiento $D_1, \dots, D_i, \dots, D_m$ cada uno de tamaño n' tomando muestras aleatorias de D .

Y en general $n' < n$. Siendo n' aproximadamente un $2/3$ de n .

Haciendo m sub-tablas tomando solo algunas filas, de forma aleatoria. ($2/3$ del total de filas).

Attribute bagging (o random subsapace):

Luego para cada una de las m tablas, se escogen sólo algunos atributos (**COLUMNAS**) de forma aleatoria también. La cantidad de atributos que se escogen, es aplicando la **RAÍZ CUADRADA** del número total de atributos. (Es recomendable sobre todo cuando hay muchas columnas).

Teniendo **m** tablas reducidas en atributos y para cada una de ellas se entrena un árbol. Para cada árbol se calcula su **matriz de confusión** (true negative, false negative, true positive, false positive).

Una vez calculada la matriz para cada árbol, se calcula la tasa de error de cada árbol, esto es: **FALSO POSITIVOS + FALSOS NEGATIVOS SOBRE EL TOTAL DE EJEMPLOS CALSIFICADOS**.

(Sobre un conjunto de entrenamiento o sobre un porcentaje del conjunto utilizado).

- Se toma el mejor árbol de los m, y **se repite el proceso K veces**.
- Al final se obtiene K árboles.

$$(FP + FN) / TOTAL = \text{Tasa de error}$$

¿Cómo clasificar una vez finalizado el proceso?

Luego para clasificar un nuevo ejemplo se realiza lo siguiente:

Ejecutando el caso que se quiere probar por cada uno de los **K** arboles. Si la mayoría de los casos devuelve que la categoría final es **CATEGORIA-A**, entonces se toma ese valor (A) de salida.

Recapitulando

Cada árbol representante de las m tablas es malo, ninguno por si solo es tan buen estimador como un árbol **C4.5** entrenado con la totalidad de ejemplos en el conjunto original sin ninguna reducción de atributos ni subsampleo, pero todos estos árboles que son estimadores mediocres, juntos y combinados superan a un árbol entrenado en el conjunto original.

Ensamblajes

- En vez de entrenar un solo modelo muy preciso o exhaustivo, entrenar varios modelos no tan buenos, c/u sobre datos distintos. Conjuntamente estos modelos mediocres van a tener mejor resultado que un estimador muy eficiente.
- Cada modelo sobre-ajusta de manera diferente.
 - Cada modelo: bajo sesgo, alta varianza.
 - Por ejemplo: árboles profundos.

Bias vs. Variance

Bias

El error debido al Bias (error de sesgo) de un modelo es simplemente la diferencia entre el valor esperado del estimador (es decir, la predicción media del modelo) y el valor real.

Cuando se dice que un modelo tiene bias muy alto quiere decir que **el modelo es muy simple y no se ha ajustado a los datos de entrenamiento** (suele ser **underfitting**), por lo que produce un error alto en todas las muestras: entrenamiento, validación y test.

Variance (varianza)

La varianza de un estimador es cuánto varía la predicción según los datos que utilizemos para el entrenamiento.

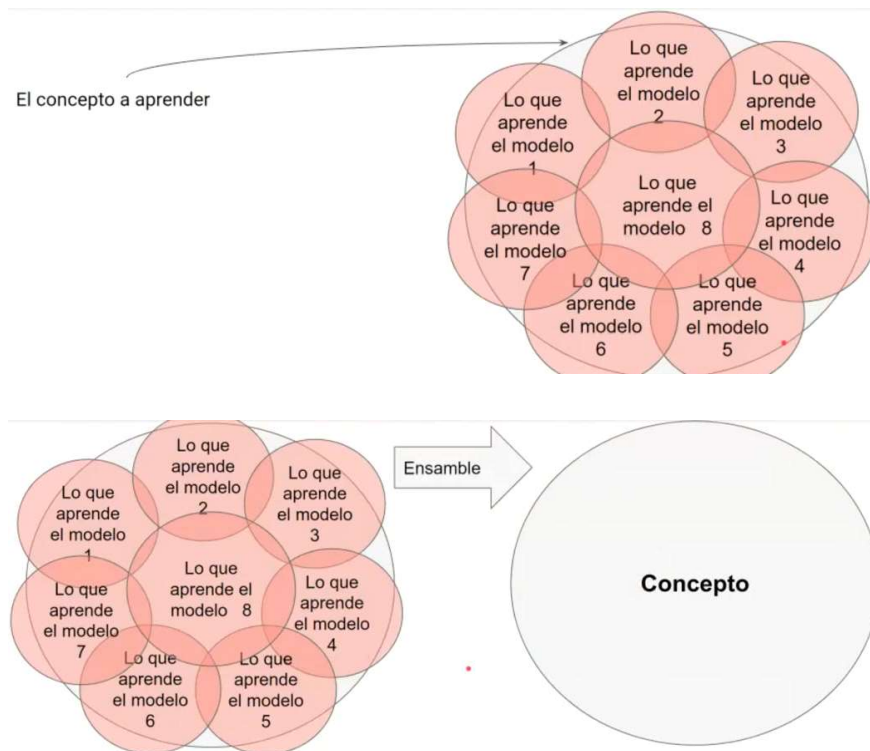
Un modelo con **varianza baja indica que cambiar los datos de entrenamiento produce cambios pequeños en la estimación**.

Al contrario, un modelo con **varianza alta quiere decir que pequeños cambios en el dataset conlleva a grandes cambios en el output** (suele ser **overfitting**).

Un **modelo balanceado** es cuando tiene un low Bias y low Variance.

Ensamblados de Modelos

Consiste básicamente en que, queremos que el algoritmo se aprenda un concepto, y el modelo que entrenamos abarca solamente una parte, pero si entrenamos varios modelos con diferentes porciones del concepto, datasets y formas de ajuste, juntos pueden cubrir diferentes aspectos del total del concepto a aprender y muchas veces con solapamiento como se ve en la siguiente imagen.



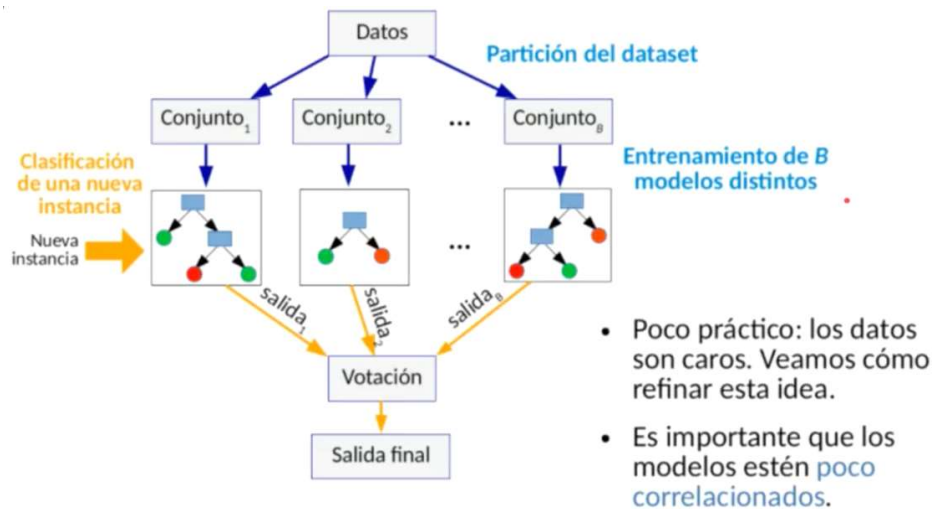
Todos los modelos que entrenamos se ensamblan y se convierten en el concepto que queremos que el algoritmo aprenda. Una forma para utilizar estos modelos para la predicción de un valor es la:

Votación: Para una nueva instancia, clasificarla con todos los modelos, y devolver la clase más elegida. (ej.: **Random Forest** usa el modelo de votación, cada modelo da una predicción y se utiliza el valor que más aparece en las predicciones). La votación en general reduce la varianza de la clasificación. Si los modelos individuales devuelven probabilidades, se puede hacer una votación ponderada.

Bagging

Es una de las dos técnicas principales para construir modelos de ensemble, y la más sencilla.

La técnica/meta-algoritmo (nos enseña como construir un algoritmo) nos dice que, a un conjunto inicial de datos, realizar **n particiones**, formando subconjuntos del original y entrenar un clasificador para cada subset. Por lo tanto, va a haber **n salidas** por cada clasificador, y a partir de ahí puede ir a la **instancia de votación** para elegir una salida final. (Esta es la técnica que usa Random Forest).



Es una técnica que consiste en construir nuevos conjuntos de entrenamiento usando **bootstrap** (bootstrap es una técnica usada por Bagging para tomar datos aleatorios del dataset original de forma aleatoria para formar los subconjuntos) para entrenar distintos modelos, y luego combinarlos.

Paso a paso

- Dividimos el conjunto de entrenamiento en distintos subconjuntos, obteniendo como resultado diferentes muestras aleatorias.
 - Las muestras son uniformes (misma cantidad de individuos).
 - Son muestras con reemplazo (los individuos pueden repetirse en el mismo conjunto de datos).
- Entrenamos un modelo con cada subconjunto.
- Construimos un único modelo predictivo (usar votación o ponderación entre todos los modelos) a partir de los anteriores.

Características

- Disminuye la varianza en nuestro modelo final.
- Muy efectivo en conjuntos de datos con varianza alta.
- Puede reducir el overfitting.
- Puede reducir el ruido de los outliers (porque no aparecen en todos los datasets).
- Puede mejorar levemente con el voto ponderado.

Problemas al usarlo con árboles

- Si pocos atributos son predictores fuertes, todos los árboles se van a parecer entre sí. El árbol siempre filtra y convierte el atributo con mayor ganancia de información en su raíz, si esos atributos predictores fuertes son pocos, muchos árboles van a tener la misma raíz de donde partir, en caso contrario, si hay muchos atributos predictores fuertes, van a ser más variados y más útil la técnica de Bagging.
- Esos atributos terminarán cerca de la raíz, para todos los conjuntos generados con **bootstrap**.

La única diferencia que hay entre Random Forest y la técnica de Bagging es que, en cada nodo, se considera sólo un subconjunto de **m** atributos (se limitaba la cantidad de columnas que se tomaban) elegidos al azar, esto es algo particular de Random Forest y que no pertenece a la técnica de **bagging** tradicional.

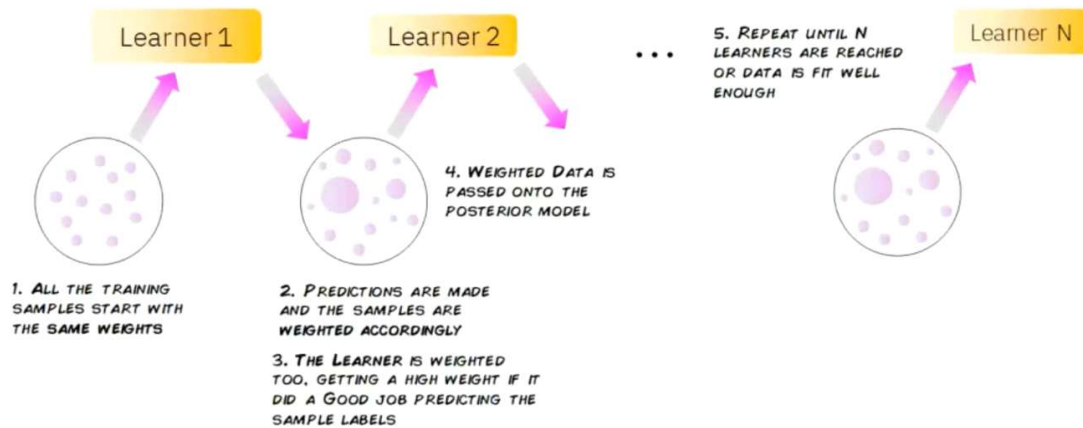
Boosting

Segunda técnica para la construcción de modelos de ensamble. Se basa en un entrenamiento secuencial, partiendo de un modelo inicial entrenado con el conjunto inicial, cada modelo que se entrena se enfoca en los errores cometidos el modelo anterior e intenta corregirlos.

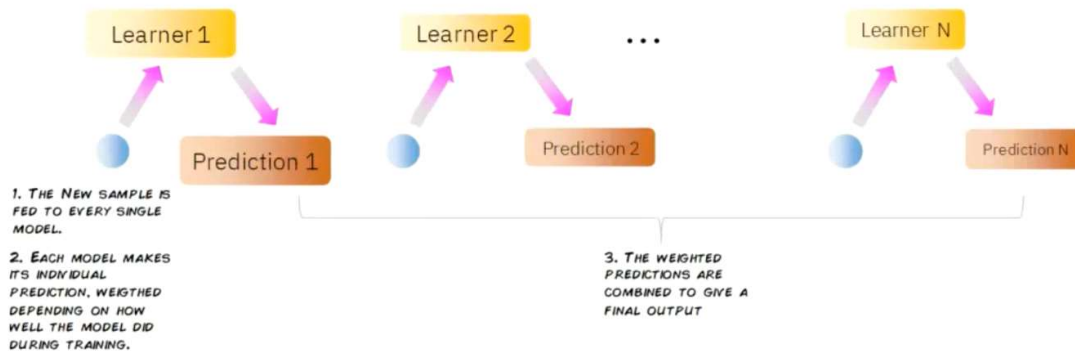
Paso a paso

- Comenzar con un modelo (simple) entrenado sobre todos los datos: **h_0** .
- En cada iteración **i**, entrenamos **h_i** dando mayor importancia a los datos mal clasificados por las iteraciones anteriores.
- Terminar al conseguir cierto cubrimiento, o luego de un número de iteraciones.
- Clasificar nuevas instancias usando una votación ponderada de todos los modelos contruidos.

TRAINING BOOSTING MODELS



PREDICTING WITH BOOSTING MODELS



Algunos **modelos exitosos** que utiliza boosting son:

- **AdaBoost**
- **Gradient Boosting**
- **XGBoost**: eXtreme Gradient Boosting

Gradient Boosting vs. XGBoost

- La velocidad de entrenamiento de **XGBoost** es mucho menor gracias a su implementación y a estar mejor orientado al uso eficiente del hardware (GPU).
- El **accuracy** (o la métrica adecuada) también es mejor debido a que **XGBoost** maneja mejor el **overfitting** mediante regularizaciones (esto lo veremos más adelante).

Resumen de boosting

- Necesita Pesos, esto implica que:
 - Debemos adaptar algoritmo de aprendizaje.
 - Y tomar muestras con reemplazo según pesos.
- Puede sobreajustar.

AdaBoost

La técnica detrás de AdaBoost consiste en entrenar un predictor, un clasificador base (por ejemplo, un árbol de decisión), verificar los errores que comete y entrenar luego otro predictor que corrija estos errores, (estas instancias mal clasificadas). AdaBoost repite este proceso hasta disminuir el error o encontrar un clasificador perfecto.

Utiliza para verificar el error el mismo conjunto de entrenamiento. Solo que, antes de entrenar el siguiente predictor, pondera las instancias mal clasificadas, aumentando su peso relativo. De esta manera el siguiente predictor entrenado estará focalizado en corregir los errores del primero.

Contras

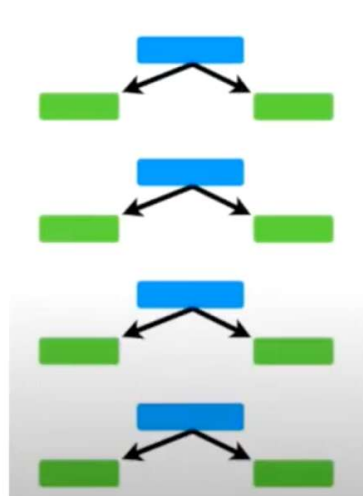
El entrenamiento AdaBoost **no puede hacerse en paralelo** (como los árboles de Random Forest) y es por lo tanto **poco escalable**.

La implementación más común es con árboles. A diferencia de **Random Forest**, donde tenemos N árboles completos (de distinta profundidad, pero completos), en **Adaboost** tenemos un bosque de tocones (stumps, solamente hay una raíz con dos hojas).

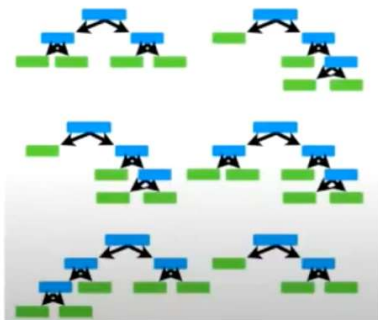
No son buenos clasificadores

Los tocones o **stump**, son árboles con un nodo raíz y dos hojas.

Stump forest

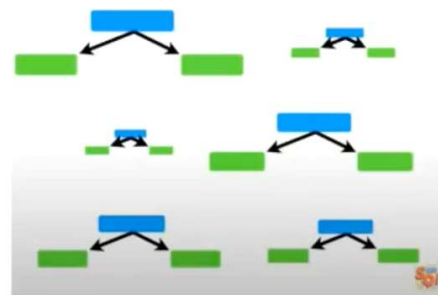


Random Forest



- Todos los árboles votan y su voto vale 1
- Cada árbol está construido de forma independiente de los otros
 - No importa el orden de creación

AdaBoost



- Los árboles votan de forma ponderada
- La creación de uno depende de los anteriores.
 - Importa el orden de creación

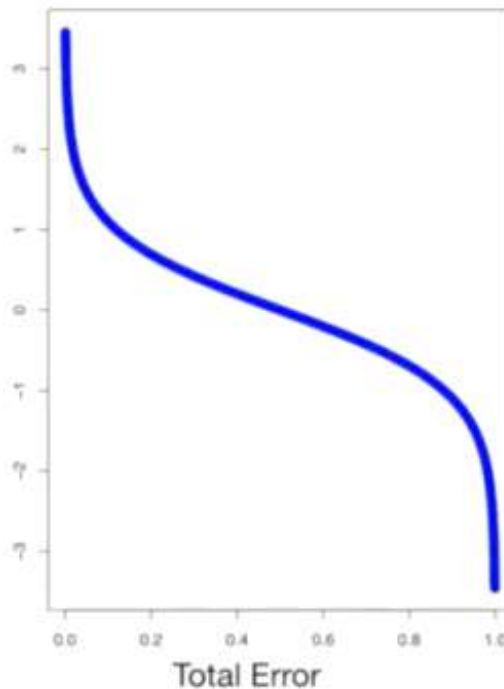
Dado un set de datos, encontrar el atributo que mejor clasifica, parecido al calculo de la impureza de Gini. Primero se crean los **n** tocones en base a los **n** atributos, cada atributo es el nodo raíz y en sus dos hojas muestran la cantidad de aciertos correctos e incorrectos. Luego se calcula el indice de Gini para cada **stumps**, el que tenga la impureza más baja, sera el primer tocón del bosque.

Es necesario calcular el peso relativo del tocón en el clasificador final. Esta ponderación es llamada: **Amount of say** y para calcularla es necesario el **Error Total** del tocón.

En un principio, todos los ejemplos de datos tienen el **mismo peso** (sample weight). El **Error total** del tocón se calcula en base a la cantidad de errores de clasificación con respecto al total de clasificación del atributo. Ahora, para calcular el **Amount of say** se usa la siguiente formula:

$$\text{Amount of Say} = \frac{1}{2} \ln \left(\frac{1 - \text{Total Error}}{\text{Total Error}} \right)$$

En el siguiente gráfico, los valores del eje x, representa la impureza de **Gini** y los del eje y representa el **Amount of Say**. La curva graficada, es inversamente proporcional al error.



Una finalizado con los calculos del primer tocón del bosque, para entrenar el segundo clasificador o encontrar el segundo tocón, es necesario focalizar en la instancia mal clasificada de la anterior clasificación. Para eso, se necesita ver cómo modificar los pesos, para que el próximo tocón a entrenar, tenga en cuenta los errores cometidos por el primero. Aumentando el peso relativo de la instancia mal clasificada, y disminuyendo (no puede ser de otro modo, ya que todo los pesos suman 1) el de los demás. Esto se logra usando la siguiente formula:

$$\text{New sample weight} = (\text{sample weight}) * e^{\text{amount of say}}$$

El $e^{\text{amount of say}}$ crece exponencialmente, pero si “amount of say” es relativamente bajo, crecerá menos que si es relativamente alto. Además, al estar multiplicado por el peso anterior (que es un número entre 0 y 1), el nuevo peso será un número intermedio.

Ahora hay que reducir el peso relativo de los ejemplos bien clasificados usando la siguiente formula.

$$\text{New sample weight} = (\text{sample weight}) * e^{-\text{amount of say}}$$

El $e^{-\text{amount of say}}$ decrece exponencialmente, si “amount of say” es relativamente bajo estará cerca de 1 (sin jamas llegar al 1). Y si es relativamente alto, estará cerca del 0. En otras palabras, si el amount of say es muy bajo (el nodo anterior no tiene mucho que decir), los pesos serán multiplicados por 1, o un número cercano a 1 y quedarán casi igual. Pero si el nodo anterior tiene mucho que decir los pesos serán disminuidos proporcionalmente.

Una vez obtenido los nuevos pesos de los ejemplos, se agrega una nueva columna con los pesos normalizados (que se calcula dividiendo cada peso por la suma total de ellos). Se reemplaza la vieja columna de pesos con la nueva y así crear el siguiente tocón (stump) del bosque.

Existen **dos posibilidades** para calcular el segundo tocón:

- Utilizar la **impureza de Gini ponderada** para saber qué tocón clasifica mejor el ejemplo con más peso. Se repiten los mismos pasos y en el paso del calculo de impureza de Gini se considera los nuevos pesos de los ejemplos multiplicandolos y a partir de ahí tomar el menor impuro.
- Generar un nuevo dataset teniendo en cuenta los pesos de los ejemplos.
 - Primero crear un conjunto de datos vacío del mismo tamaño que el anterior.
 - Armar una nueva columna de distribución de valores, sumando los pesos. (ej.: fila1: 0.07, fila2: 0.07 + 0.07, fila4: 0.14 + 0.07, fila4: 0.21 + 0.49, fila5: 0.70 + 0.07, ..., etc.).
 - Elegir al azar un número entre 0 y 1. Verificar en que fila de la columna de distribución cae. (ej.: 0.05 está entre 0 y 0.07 o 0.35 esta entre 0.21 y 0.70). Se agregan las filas elegidas al nuevo dataset.
 - Se reemplaza el viejo dataset con el nuevo y se agrega nuevamente el

peso de forma equitativo para todos los ejemplos. La fila con más peso en el dataset viejo, puede aparecer de forma repetida en el nuevo y algunas fila ni siquiera aparecen. Para ejemplo repetido, su peso depende de la cantidad de veces que esta repetido.

- Calcular nuevamente los tocones para los atributos.

De esta forma los errores cometidos por un árbol o tocón son utilizados para construir un segundo árbol y los errores de este servirán a su vez para construir el siguiente árbol y etcetera.

Finalmente, para predecir un nuevo valor, se agarran todos los tocones y se calcula el total del Amount of Say, por ejemplo, para los casos de tocones True y los False. El que tiene más puntaje gana y se lo toma como resultado final de la predicción.

Resumen

- AdaBoost combina un montón de **weak learners** (estimadores pobres) para hacer clasificaciones. Estos **weak learners**, son generalmente **stumps** (tocones).
- Algunos **tocones** tienen más peso que otros en la votación final, tienen más que decir (**amount of say**).
- Cada uno de estos tocones está construido teniendo en cuenta los errores del tocón anterior.
 - Lo podemos hacer usando función de **impureza de Gini ponderada**, para cada ejemplo.
 - O simplemente regenerando los datos.

Gradient Boost

- Gradient Boost crea una cadena de árboles de profundidad fija.
- Comienza con un solo valor, un nodo hoja. Y luego calcula árboles para calcular el error cometido por el anterior.
- Cada árbol está ponderado por un factor constante llamado **learning rate** o tasa de aprendizaje.
 - Los árboles están restringidos en su crecimiento.

El modelo de **gradient boost** puede usarse para problemas de regresión (ej.: cuando se quiere determinar un valor en un rango continuo) o de clasificación.

Dado un dataset y se necesita determinar un valor en un rango continuo. Problema de regresión.

- **Paso 1:** Calcular una sola hoja de un árbol, que clasifica todos los casos. Esta hoja es el promedio. Calcular el **valor promedio** entre todos los valores del atributo clasificatorio y asignar ese resultado como la predicción estandar para todos los ejemplos.
- **Paso 2:** Calcular el error cometido por el estimador hasta ahora. Se lo llama **Residuo** (la diferencia entre el valor de la clase observado y el valor predicho). Se reemplaza la columna con los valores de la clase por la columna con los valores Residuales. **Nota:** El término correcto es **pseudo-residuo**. Los **residuos** son los errores en **regresión lineal**, acá se toma la palabra por similitud, pero se le agrega el **pseudo** para entender que es otro modelo.
- **Paso 3:** Construir un árbol para los Residuos (errores) que estará limitado a solo **4 hojas** (en el dataset ejemplo del video hay solo 3 atributos y la columna de Residuos). En casos reales se usa un valor entre 8 y 32 hojas. En los nodos padres siempre van los valores de los atributos, y en los nodos hoja los valores Residuales, que pueden haber más de uno en una misma hoja, en ese caso se unifica calculando el promedio.

Entonces, para predecir un nuevo valor, se toma el **promedio calculado inicialmente** (en el **Paso 1**) y se lo **suma** con el **valor residual dependiendo de la salida del árbol** (recorre el árbol dependiendo de las condiciones de los atributos del nuevo valor con las del árbol para identificar el valor residual del nodo hoja que se toma para la suma). El resultado de la suma es la predicción final de este modelo.

Pero solo con el resultado de la suma entre esos dos valores no alcanza para obtener una predicción segura, en un sentido de que, el modelo puede haberse aprendido todos los casos y haber un alto variance que implica un **overfitting**, por eso, para manejar este problema, **Gradient Boost** usa el parametro **Learning Rate** (valor entre 0 y 1), para escalar la contribución de cada árbol. Por lo tanto, para predecir un valor, se multiplica el learning rate al valor residual de la salida del árbol y finalmente sumarle el promedio. El resultado obtenido por esta nueva cuenta, tal vez no sea una muy buena predicción pero es mucho mas flexible y no esta sobreajustado provocando un **overfitting**, igual sigue siendo mejor que la estimación calculado en un comienzo (Paso 1).

Según **Jerome Friedman**, el autor de este método, la evidencia empírica sugiere que tomar muchos pequeños pasos en la dirección correcta resulta en una mejor predicción en un conjunto de datos de prueba. Es decir, tendrá una **varianza baja**.

No se áca termina todavía. Luego, se aplica para todos los ejemplos la misma suma realizado anteriormente con el valor de learning rate incluido y con el resultado obtenido se vuelve al **Paso 2** calculando nuevamente el valor Residual en base al valor original de la clase del dataset. Se puede notar que los errores (residuos) se redujeron. Y con esos nuevos valores de error (Residuos) se crea un nuevo árbol (**Paso 3**) y

entonces para predecir un valor, se realiza el calculo con el mismo valor promedio de la estimación inicial más el valor residual del primer árbol multiplicado por el learning rate y ahora se le suma tambien el valor residual del nuevo árbol multiplicado por el mismo learning rate.

De esta forma, seguimos añadiendo árboles, hasta que los residuos no cambien significativamente o bien alcanzamos un número preestablecido de árboles seteado como parámetro.

Finalmente, para estimar un nuevo valor, cuando el método termina de entrenar, procedemos como antes, usando el valor inicial y sumando los residuos ponderados de cada árbol.

XGBoost: eXtreme Gradient Boost

XGBoost fue diseñado para **Big Data**, es decir para **conjuntos de datos grandes y complejos**. Sin embargo, a fines de entender el algoritmo principal lo usaremos con un conjunto de datos simples (y para el caso de regresión).

Dado un conjunto de datos super sencillo, se busca predecir el valor de una clase binaria dado cierto valor de un atributo.

- **Primer paso:** Hacer una predicción inicial, que puede ser cualquier valor, ya sea para regresión o clasificación. En base a esta estimación se calculan los errores (Residuos).
- **Segundo paso:** Construir un árbol para los residuos al igual que el caso de **Gradient Boost**, pero con la diferencia de que se crea un nodo hoja con todos los residuos. (Hay otras formas para construir el árbol de XGBoost, pero este es el más común).
- **Tercer paso:** Calcular el **Similarity Score**, para los residuos del nodo hoja.

$$\text{Similarity Score} = \frac{(\text{Suma de residuos})^2}{\text{Cantidad de residuos} + \lambda}$$

λ (lambda): es un parámetro de regularización (evitar el overfitting).

Regularización

Cuando no se tiene un conjunto de entrenamiento ideal y se quiere flexibilizar y evitar el overfitting y los problemas de alta varianza. Una forma de solucionar este problema, es usando una variación de la regresión lineal llamada: **Ridge Regression**. Es la misma

regresión lineal sumado un parámetro de regularización (la idea es encontrar una línea que no ajuste tan bien). Para ello se va a introducir un pequeño sesgo o **bias**, en los datos de entrenamiento (**Bias-variance tradeoff**). El error de sesgo introducido en el estimador, hará que caiga el error por la varianza de forma mucho más abrupta en el conjunto de pruebas.

Al usar **Regresión Lineal**, es decir cuadrados mínimos, lo que se está haciendo es minimizando la **suma del cuadrado de los residuos**. En cambio, en **Ridge Regression** se minimiza lo mismo más $\lambda * (\text{pendiente})^2$. La pendiente al cuadrado representa una penalización al método tradicional. Y λ determina qué tan severa es esta penalización.

Volviendo al cálculo de **Similarity Score** del tercer paso, una vez obtenido el resultado del cálculo usando todos los residuos.

- **Cuarto paso:** Para ver cuál es el siguiente nodo al primer nodo hoja, se calcula la ganancia total, según se escoja una opción u otra a partir del árbol.
 - **Opción 1**, tomar como umbral la distancia intermedia entre las primeras dos observaciones.
 - **Opción 2**, tomar como umbral la distancia intermedia entre las segundas 2 observaciones.
 - **Opción n**, tomar como umbral la distancia intermedia entre las observaciones **n-1** y **n**.

Entonces, dependiendo de la cantidad de observaciones que se tiene, hay que verificar cada opción hasta encontrar el mejor. Para eso se divide el conjunto de observaciones en dos (se sacan dos nodos hoja del nodo raíz que contenía todos los **residuos**) y se calcula el **Similarity Score para cada nodo hoja** y para obtener la ganancia total (**Gain**) del árbol de dos hojas se calcula de la siguiente forma:

$$\text{Gain} = \text{Nodo_Izquierda}_{\text{Similarity Score}} + \text{Nodo_Derecha}_{\text{Similarity Score}} - \text{Raiz}_{\text{Similarity Score}}$$

De esta forma se calcula la ganancia total (**Gain**) para cada umbral que se toma (para cada opción del cuarto paso). Entonces, el árbol que tenga un **mayor Similarity Score**, se lo considera como el umbral que mejor divide los residuos del árbol (la mejor opción). De esta forma se consigue un árbol con dos hojas.

- **Quinto paso:** Se repite el anterior paso hasta alcanzar la profundidad del árbol estipulada. (Se repite el cuarto paso para los nodos hojas que tengan más de dos valor Residual y así armar el árbol (XGBoost por defecto puede llegar hasta 6 niveles profundidad)).

- **Sexto paso:** Una vez finalizada la creación del árbol, se aplica un algoritmo de poda que consiste en los siguientes pasos:
 - Elegir un número al azar llamado gamma (γ), completamente al azar.
 - Luego, calcular la diferencia entre el Gain (ganancia total), del nodo más bajo y gamma. **Gain - γ** .
 - Si la diferencia es < 0 , entonces removemos el nodo.
 - Si no, el nodo se queda y se termino la poda.

Si se elige un valor muy alto de gamma, se termina podando todos los nodos del árbol, quedando solo la estimación inicial. Por defecto, gamma es igual a 0 (cero). Cuanto más alto es gamma, más conservador es el algoritmo.

- **Septimo paso:** Volviendo a calcular el árbol (repite el **Tercer paso**), solo queda esta vez se usa lambda igual λ a 1 al calcular el Similarity Score.
 - Con lambda > 0 , los Similarity Scores son mucho más chicos.
 - La disminución es proporcional a la cantidad de residuos en el nodo.
- **Octavo paso:** Calcular la ganancia de cada nodo con los nuevos valores de Similarity Score. Es decir, el **Gain**. Tener lambda > 0 hace que el **Gain** también sea menor. Luego se vuelve al sexto paso nuevamente para aplicar la poda.
 - Cuando $\lambda > 0$ es más probable tener que podar un árbol, ya que los **Gain** calculados son menores.
 - Aún eligiendo $\gamma = 0$ podemos tener que podar nodos, ya que la ganancia (**Gain**) puede ser negativa.
 - Con $\lambda = 1$, se previene el sobreentrenamiento o sobreajuste del modelo en el conjunto de entrenamiento.

¿Cómo calcular la respuesta en XGBoost?

Con el árbol calculado hasta ahora, calcular el **Output** para cada nodo hoja del árbol (formula parecida al Similarity Score).

$$\text{Output} = \frac{\text{Suma de residuos}}{\text{Número de residuos} + \lambda}$$

Tomando $\lambda = 0$ ya que es el valor por defecto.

Al final para hacer una nueva estimación (predicción), se busca el **Output** indicado en el árbol (la salida del árbol), se le multiplica por ϵ (es un **Learning rate** que **por defecto** es **0.3**) y a ese resultado se le suma la estimación inicial.

El resultado de la predicción tal vez no sea muy bueno, pero se acerca de a poco al valor real y reduce el error (Residuo).

- **Nuevo paso:** Calcular todos los residuos utilizando el árbol hasta acá. Se puede notar una mejora respecto a la estimación inicial. Sin embargo, los errores (residuos) siguen siendo altos.
Con los nuevos residuos, se construye un nuevo árbol repitiendo todo desde el **Segundo paso**. Con el nuevo árbol, se calcula la salida de cada elemento y luego los residuos. Y de esta forma, se iteran los pasos hasta que los residuos son prácticamente cero o bien se alcanza el número máximo de árboles predefinido. Al final se van acumulando y sumando los Outputs de cada nuevo árbol multiplicado por el Learning Rate (**0.3**) y la estimación inicial.

Ensamblajes Híbridos

- Ensamblajes homogéneos: combinan el mismo tipo de modelo.
 - Bagging.
 - Boosting.
- Ensamblajes híbridos: combinan clasificadores de distinto tipo.
 - Voting.
 - Stacking.
 - Cascading.

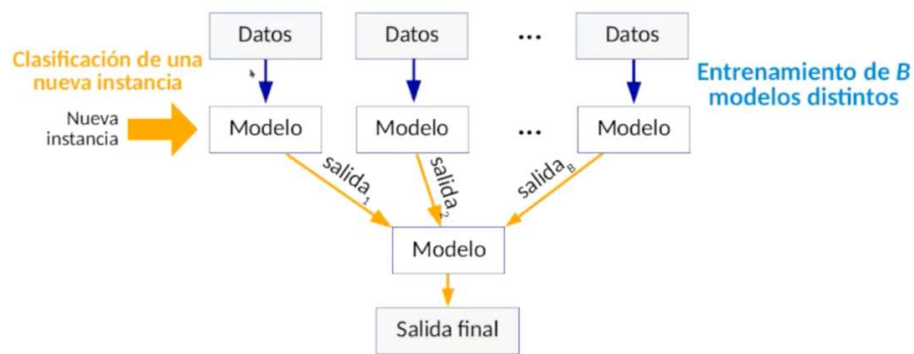
Voting

Construir N modelos utilizando los mismo datos y luego tomar la predicción mayoritaria.



Stacking

- Entrenar diferentes modelos (modelos base) **y un modelo más**, que decide, dada una instancia nueva, qué modelo usar.
- Concepto de **meta-aprendizaje** para **reemplazar el mecanismo de voto**.
- Pueden apilarse tantas capas de modelos como se desee.



Características del stacking:

- Los modelos para el **meta-aprendizaje** suelen ser: Árboles, Naive Bayes, SVM o Perceptrón (tema de Redes Neuronales).
- Para los otros puede usarse cualquiera.
- Menos popular que boosting, baggin.
 - Dificultad de análisis teórico: caja negra.
 - Múltiples variantes.
- Se puede interpretar como una mejora (generalización) del método de votación (Voting).
- Si los clasificadores base pueden generar medidas de certeza, suele funcionar mejor.

Cascading

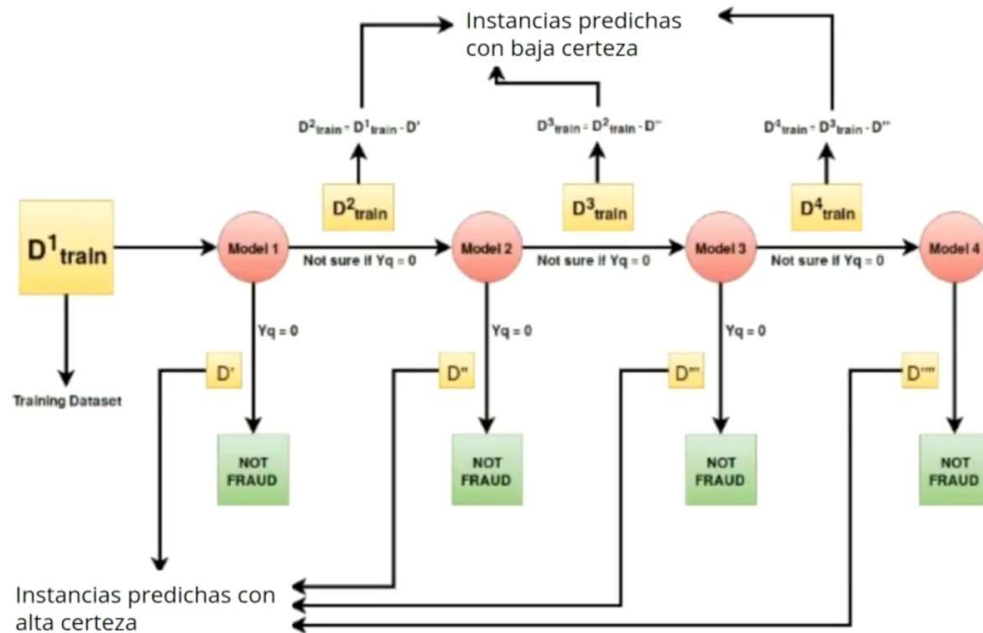
- Enfoque en el que se **pasa sucesivamente los datos de un modelo a otro**.
- A diferencia de Stacking, **cada "capa" tiene un solo modelo**.
- El input de cada modelo son las instancias predichas con poca certeza por el modelo anterior.
- Suele utilizarse cuando se necesita una **alta certeza en la predicción**.

Ejemplo

- Se quiere un modelo que prediga si una transacción con tarjeta de crédito es fraudulenta.
- Se necesita una alta certeza para definir si no lo es, en caso de error las pérdidas pueden ser millonarias.
- Construir entonces una secuencia (o cascada) de modelos de ML.

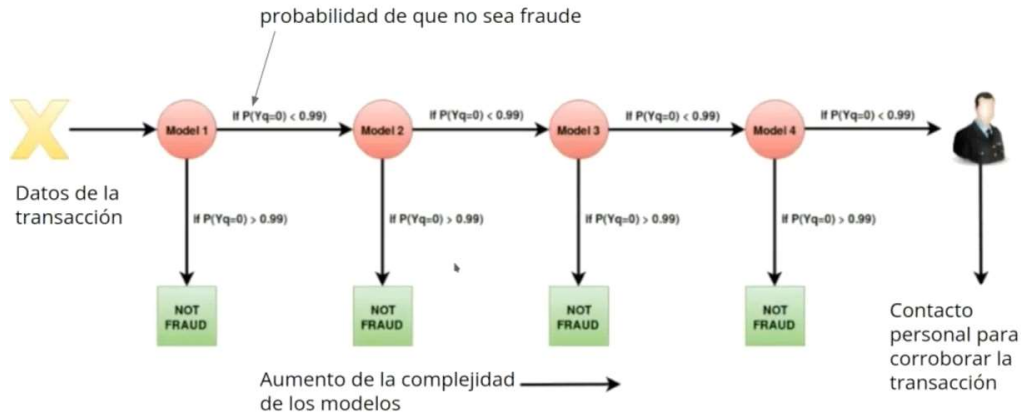
Entrenamiento

- Entrena N modelos distintos utilizando el set de entrenamiento.
- Cada modelo es entrenado sobre las instancias predichas con baja certeza del modelo anterior.
- Suele hacerse de forma tal de empezar por modelos simples y a medida que se entrenan nuevos, los mismos sean de mayor complejidad.



Predicción

- El primer modelo recibe los datos de la transacción.
- Si la probabilidad de que no sea fraude es menor a 0.99, se pasa al siguiente modelo.
- Caso contrario se descarta la posibilidad de fraude.
- Sin ningún modelo descarta el fraude con $p < 0.99$ se procede a confirmar la operación personal.



Características

- A diferencia de Voting y Stacking que tienen un enfoque de modelos “Multi-expertos”, Cascading tiene un enfoque “Multi-estado”.
- Inicialmente creados para computer vision.
- Cascadas muy profundas pueden producir overfitting.

Resumen y propiedades de los ensambles

- La **combinación de clasificadores** permite generar **clasificadores más precisos** a cambio de **menor comprensión** de los mismos.
- **Métodos homogéneos básicos**: bagging, boosting.
 - Combinan mismo tipo de clasificador.
 - Manipulación del conjunto de entrenamiento.
- **Método heterogéneo básico**: Voting.
 - Combinan distintos tipos de clasificador.

Ventajas

- Generalmente logran **mejores predicciones** que los modelos vistos hasta ahora.
- Buen **trade-off** entre sesgo y varianza.

Desventajas

- Se pierde interpretabilidad.
- Tienen una complejidad computacional mayor.

Redes Neuronales

Las redes neuronales son varias neuronas juntas, junto a una función de activación aprende encontrando los mejores valores para los pesos y biases.

Un perceptrón simple es la forma más simple de una función de activación de una red neuronal. Esta limitado a problemas que se puedan dividir con una línea recta.

Con un perceptrón multicapa (otra capas de neuronas) se puede resolver estos tipos de problemas. Estas están fully connected.

Las redes SOM tienen dos capas (entrada y salida). Mediante una distancia se obtiene una neurona la cual esta mas cerca de la neurona de entrada de manera iterativa se va ajustando el plano de salida. (No supervisada)

El error cuadrático medio se calcula para saber como resultó la predicción, el error total es la suma de todos los errores.

Backpropagation es la técnica de usar el descenso por gradiente para volver a calcular los **pesos** de la red neuronal en base al resultado obtenido, de manera recursiva.

Para decirle a la red si está funcionando bien o no utilizamos el descenso por gradiente por backpropagation, una neurona depende de los valores de las activaciones de las neuronas de la capa anterior. Así ajustamos los pesos hasta que estemos conformes. A medida que vayamos calculando las derivadas de las capas iniciales los valores se vuelven mas pequeños y el gradiente se va desvaneciendo. Converge cuando hay muchas capas.

Cuando tengo una regresión utilizo la función de activación sigmoidea, lineal, relu, etc, para las clasificaciones **binarias la sigmoidea** y para las clasificaciones **multiclase la softmax** (Permite transformar salida de neuronas en probabilidades).

Para **evitar el overfitting** se pueden aplicar varios métodos, como **regularización L1 y L2** (penalizan el valor de los pesos de la red, evita que se le dé más relevancia a un elemento que a otro), **Dropout** (Apaga activaciones aleatoriamente durante el entrenamiento), o **Early Stopping** (Para el set de entrenamiento antes de que el error del set de validación empiece a aumentar).

Optimizadores

Los optimizadores son una implementación del algoritmo de backpropagation, básicamente casi todos los optimizadores se complementan, parcheando cosas de los anteriores.

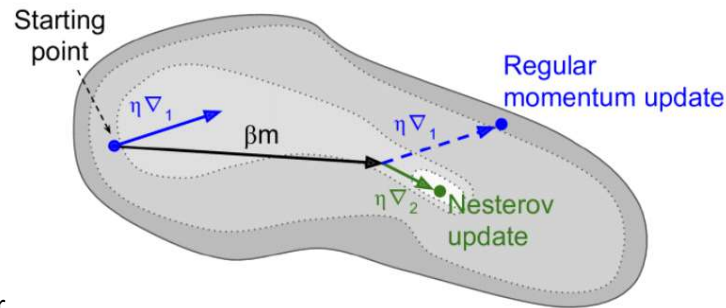
SGD > Backpropagation normal

Momentum > El gradiente se utiliza para la aceleración y no para la velocidad (es un hiperparam.)

Nesterov > Variante de Momentum, calcula el **gradiente** movido en **dirección del momento**. (Hiperparam.)

AdaGrad > Bueno para problemas de N dimensiones, reduce el gradiente a lo largo de las

dimensiones más empinadas. Bueno para tareas sencillas pero **NO SE USA** para **redes profundas**. Evita caer en mínimos locales para ir en dirección de mínimos globales pero se suele



detener antes de llegar.

RMSPProp > Soluciona el problema de AdaGrad “olvidando” las pendientes anteriores (Hiperparam. β o rho,

Adam > Combina las ideas de Momentum y RMSPProp

AdaMax > Modificación de Adam que obtiene el máximo.

Nadam > Adam + Nesterov (converge más rápido que Adam)

AdaDelta > Variación de **AdaGrad**. Tiene una ventana en la que guarda algunos pasos anteriores y no todos. Similar a RMSPProp ya que olvida pendientes anteriores.

AdaDelta es bueno rápido, RMSPProp y AdaGrad van parecidos.

Con una capa oculta para la mayoría de los casos ya es suficiente.

Redes Profundas

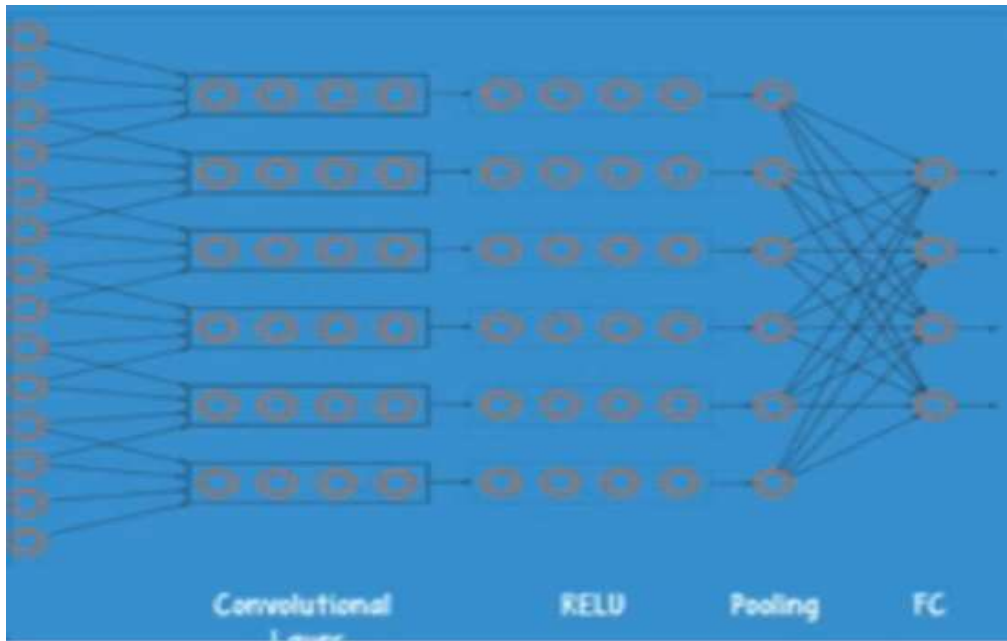
Las redes profundas sirven para el procesamiento de textos o el reconocimiento de patrones, imágenes y objetos.

Las RBM ejecutan de los dos lados hacia el medio hasta que se asemejen lo más posible. Por otro lado, la DBN es igual a un perceptrón multicapa, pero entrena diferente.

Estas son autoencoders, que aprenden a producir la salida igual a la entrada. Estas se entrenan con backpropagation y la métrica LOSS.

Las **redes convolucionales** sirven para la clasificación de imágenes. Son similares a las redes vistas anteriormente, sólo que algunas capas aplican operaciones de convolución.

En estas **capas de convolución** se aplican filtros y Kernel, los cuales son los encargados de buscar features relevantes. Luego **capas de RELU**, entrenadas con backpropagation, y luego las **capas de pooling** son utilizadas para reducir la dimensionalidad, la más usada es la maxpool pero existe average pool. Termina con una **capa Fully Connected** para clasificar los ejemplos.



La convolución es una manera de combinar dos funciones en una nueva. Se usan **kernels** para conseguir features convolucionados.

RELU es una función de activación. $y = \max(0, x)$

Las redes recurrentes se usan cuando los patrones en los datos cambian con el tiempo. Pueden recibir una secuencia y devolver una secuencia. Pueden capturar imágenes en movimiento o clasificar textos. Es muy recurrente el desvanecimiento del gradiente. Se puede solucionar con Gating (Olvidar el input de manera momentánea).

Redes de tensores recursivas, tienen estructura de árbol y utilizan backpropagation, especialmente útiles para análisis de sentimientos y detección de componentes en imágenes.

Las Deep fake nets utilizan RBM y DBN para generar las caras con el autoencoder.

Las redes GAN no clasifican datos ni resuelven problemas de regresión, son no supervisadas. Generan datos trasponiendo una red convolucional.