**RESUMEN DE INTRODUCCIÓN A LA IA**

1. **Unidad 1: MACHINE LEARNIG**

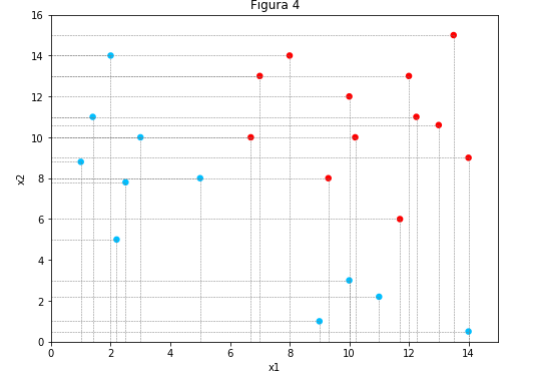
La variante más efectiva de la IA actual, el **Aprendizaje Automático** o **Machine Learning**, necesita datos de los cuales "aprender".

En IA se trabaja con grandes cantidades de datos, si nosotros quisiéramos notar el algoritmo en una lista de datos lo primero que se recomienda es ordenar los datos con cierto criterio, haciendo esto se pueden ver a simple vista los patrones.

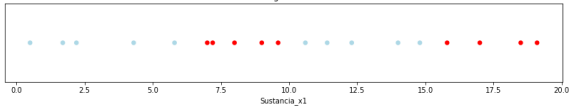
Otro método grafico podría ser ordenar los datos en un sistema de ejes cartesianos, así podríamos notar más fácil el patrón, pero esto solo se puede realizar con máximo de 3 variables (features), ya que si supera a esto no podríamos graficarlo. También se lo podría acomodar a los datos si son numéricos en una lista numérica, así podría determinar también el patrón. Para poder plasmarlas de estas formas, los datos deberán ser si o si de tipo numérico.

Conociendo esto determinamos que en ciencia de datos se le da mucha importancia a la visualización de los datos

**Ejemplo**:



**O**



**ÁRBOLES DE DECISIÓN:**

El árbol de decisión es una manera clara y sencilla de resumir las reglas, además, este diagrama adquiere más trascendencia cuando el patrón tiene muchas reglas y variables. Este modelo de representación es el más utilizado en la IA.

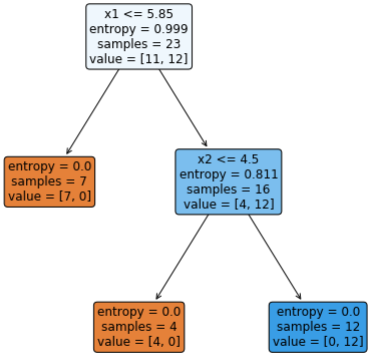
Podemos usar el modelo de árbol para resolver problemas que tengan muchas variables, por otro lado, como ya habíamos dicho no podemos extender nuestra intuición geométrica de subdividir el plano o el espacio a más de 3 dimensiones o variables, así que cuando tengamos más de 3 variables es recomendable utilizar este diagrama.

Generalmente los resultados de los árboles de decisión tienen muchos datos, pero son definidos muy sencillamente. **¿Cómo se leen los diagramas de árboles?**:

Un árbol desarrollado con Python tiene varias partes, las cuales son:

* El rectángulo se denomina **nodo**, y el primer nodo se denomina **raíz**. El tipo de nodo "terminal" se denomina **hoja**. Una vez que todos los nodos se transforman en hojas se acaba el modelo
* **Samples:** cantidad de casos, observaciones o filas que tienen nuestros datos
* **Value:** nos indica que los valores de la columna que queremos pronosticar está conformado por x cantidad de cada caso
* **Entropy (entropía):** indica el grado de incertidumbre que tenemos en cada nodo. Cuando los resultados de x e y se encuentran en la misma cantidad, la entropía es máxima, de lo contrario es mínima. Cuando la entropía es 0 significa que es una hoja.
* **x <= 5,85:** nos indica el valor donde dividió, se puede considerar como una pregunta, por ejemplo: **¿x es menor o igual a 5,85?**
* **Flechas:** la flecha a la **izquierda** es para una respuesta **afirmativa**, mientras que a la **derecha** corresponde una **negativa**

**Ejemplo:**

****

**¿Por dónde dividir los datos de un diagrama de ejes cartesianos?**

La respuesta está en la cantidad de observaciones o puntos que separamos en cada caso:

Si en un diagrama al separar verticalmente dejamos un grado de incertidumbre menor (mayores datos aislados) a que si hubiésemos empezado horizontalmente, deberíamos empezar separando verticalmente. Ejemplo: si al subdividir en vertical separo 7 de 13 datos y si separo en horizontal separaría 5 de 13 datos, me convendría mucho más sacarme de encima 7 datos y luego ver como sigo.

1. **UNIDAD 2: PROBLEMAS CON FEATURES DE TIPO CATEGORICO**

Para estos casos, como dijimos anteriormente se recomienda usar los ARBOLES DE DECISIONES, donde dividiremos el espacio tantas zonas como features tengamos en el problema, y estas zonas darán el resultado de si o no, dependiendo a que resultado se quiera llegar.

**¿Cómo armamos nuestro árbol?**

Analizamos los datos que tenemos y ¿qué estamos buscando? la respuesta es encontrar una variable que separará o distinguirá con total "precisión" los casos de No y Si, esa variable ideal no existe, pero tenemos una idea de lo que estamos buscando. Para empezar, deberemos analizar cada variable y elegir la que disminuye la incertidumbre inicial. Y así procederemos sucesivamente con cada nodo hasta terminar.

Procedimiento:

1. Calcular la entropía de la variable que deseamos pronosticar
2. Calcularemos la entropía de todas las variables como primero nodo
3. Calcularemos la entropía luego de la partición
4. La variable que nos dé una mayor disminución de la entropía será elegida como primer nodo.
5. Crearemos las ramas de este primer nodo.
6. Repetiremos los pasos anteriores para cada uno de los nodos que se creen, excepto que sea una hoja

**Entropía e Incertidumbre:**

Para empezar a armar un árbol de decisión tendremos que generar nodos e ir separando variable por variable, como habíamos nombrado antes. El nodo que elegiremos primero será el que genere la menor entropía posible, dicho de otro modo, lo que haremos será tratar de que la incertidumbre disminuya cuando elegimos separar con alguna de las variables.

La incertidumbre se puede medir con un valor numérico, mediante la fórmula de la **Entropía de Shannon**

**Fórmula de la Entropía de Shannon:**

Una de las cosas que deberemos calcular es la proporción en que aparece cada uno de los valores, que es la cantidad de si o no / cantidad total de casos. Ej:



**ARBOL DE DECISIÓN POR GANANCIA DE INFORMACIÓN:**

Los conceptos de entropía son opuestos al concepto de información, por ende, la disminución de la incertidumbre implica también una ganancia en la información. La mayor caída en la incertidumbre se produce cuando particionamos el árbol con la con menor incertidumbre; y por lo tanto con ella comenzaremos a armar nuestro árbol.

Para determinar que variable usaremos como nodo principal calcularemos la Entropía de cada rama y luego las sumamos multiplicadas por la proporción de casos con respecto al nodo superior, luego compararemos cada una y elegiremos la que disminuya la entropía, como ya habíamos dicho. ***Ejemplo: S1inicial= 8/13 \* S\_izquierda + 5/13 \* S\_derecha***

1. **UNIDAD 2: LA INTELIGENCIA ARTIFICIAL Y SUS PARTES**

**¿Qué es la Inteligencia Artificial (IA)?:**

La IA es un área que ha evolucionado desde mediados del Siglo XX desde antes que existieran las computadoras, esto es porque los fundamentos de la IA son lógicos, por lo que no se necesitan de una computadora para realizarlo. **Evolución de la IA**: Primero se crea la inteligencia artificial, luego el machine learning o aprendizaje automatico y lo más actual es el Deep learning.

**Sistemas Expertos:**

Los primeros intentos exitosos de IA se basaron en los Sistemas Expertos (SE). La forma de resolver problemas con los **SE** es recopilar la opinión de expertos y con esto crear las reglas para luego terminar programándolas en sistemas (if elif else).

**Machine Learning (Aprendizaje Automático):**

El machine learning es el campo de estudio que le da a las computadoras la **habilidad de** **aprender sin haber sido programadas**. En este caso, aprender significa que el sistema deberá crear o descubrir las reglas necesarias para resolver el problema y deberá recordarlas para luego usarlas en futuros problemas.

En **ML** se usa mucho la palabra **pronosticar**, y significa aplicar lo aprendido a nuevos casos. A través de estos sistemas es habitual que se descubran nuevas reglas y es posible descubrir características que se pensaba que tenían poca importancia resultaran ser de gran importancia para la evaluación futura

Los sistemas de **ML** aprenden de los datos, **¿cómo deben ser estos?** de gran cantidad, de buena calidad, de fuentes confiables, bien documentados (una misma unidad de medida), ser bien manipulado, bien recolectado y estar ordenados en tablas digitales. Además, deben mostrar la misma cantidad de datos que son lo que queremos diagnosticar y lo opuesto, por ejemplo, si quisiéramos entrenar un programa que diga si es o no una silla, deberíamos cargarle miles de imágenes de sillas de diferentes ángulos o formas y la misma cantidad de imágenes que no lo sean.

**¿Cómo sabremos cuántas fotos son suficientes?** Cuando el sistema alcance la medida de evaluación fijada, por ejemplo: si entrenamos al sistema de ML con 100 fotos y obtenemos un 50% de aciertos podemos asumir que el sistema es neutro, al aumentar la cantidad de fotos de sillas el % de aciertos aumentará y será más eficiente.

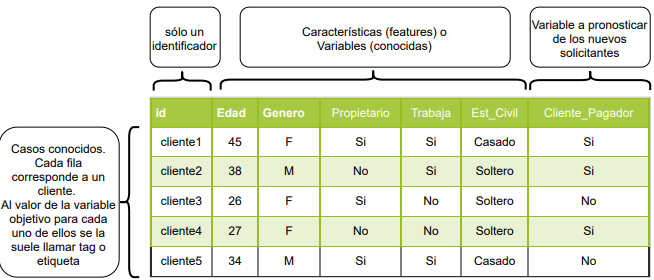
**¿A cuál correspondería el Árbol de Decisiones que creamos con anterioridad?:**

El árbol que creamos antes corresponde a un sistema de **ML**, ya que si creamos el árbol basándonos en datos almacenados del historial sería un sistema de ML, en cambio si creáramos un árbol generando varios cuestionarios y preguntándole a expertos como serían las reglas para diagnosticar una enfermedad, por ejemplo, estaríamos utilizando SE

**Organización de los Datos:**

La mayoría de los Métodos de ML requiere que los datos estén organizados de determinada manera, se los suele denominar tiddy data, que quiere decir “datos ordenados".

La **variable que queremos pronosticar** se llamará variable o función objetivo (**target**) o variable o función de salida **o variable a pronosticar**. En cambio, **los datos ya conocidos** que usaremos para crear las reglas se llamaran Características (**features**) o variables conocidas o variables. Se recopilan todas estas variables y se ordenan en forma de tabla como la siguiente:

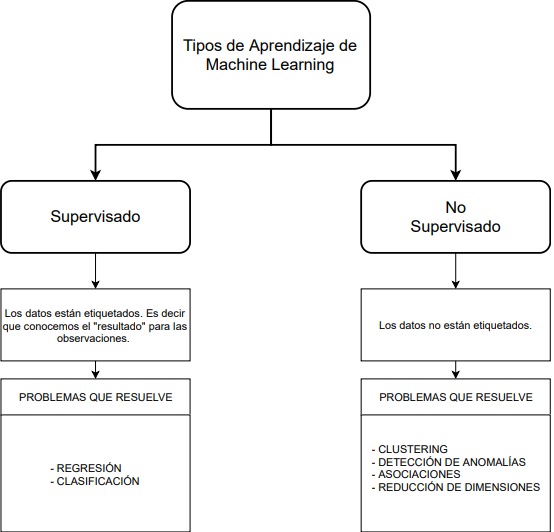


En esta tabla tenemos:

* **Filas**: Cada fila correponde a los datos de una categoría, a las filas en ML se suele denominar Observación o también registro.
* La **columna id** es sólo un identificador, no nos brinda ninguna información. En ML no es necesario agregarla, y por lo que no es una variable.
* Las **columnas: Edad, Genero, Propietario, Trabaja, Est\_Civil** son las Características, features o variables conocidas y estas nos dan la información importante.
* la **columna Cliente\_Pagador** es la variable objetivo o target, es decir lo que queremos pronosticar. Como conocemos el valor de esta variable se dice que están etiquetadas o tienen tag o label o que están clasificadas.

**Tipos de Aprendizaje que se utilizan en Machine Learning:**

Existen diversas formas de aprender en ML. Estas se utilizan para resolver distintos tipos de problema y dependen de qué tipo de datos dispongamos.



**APRENDIZAJE SUPERVISADO:**

En el Aprendizaje Supervisado el conjunto de datos con el que vamos a entrenar al sistema está etiquetado, es decir **conocemos el valor de la variable objetivo** que usaremos en el entrenamiento. Lo que queremos es que nuestro sistema aprenda de esas observaciones etiquetados para aplicar ese conocimiento a nuevos casos de los que no conocemos el resultado.

**Ejemplo de Aprendizaje Supervisado**: una entidad bancaria que deseaba pronosticar si dar o no dar créditos a solicitantes, ya que conocíamos los datos históricos de otros clientes referidos a si eran Clientes Pagadores o no.

El Aprendizaje Supervisado se aplica a problemas de:

* **REGRESIÓN:** Los problemas de Regresión son aquellos en los cuales **la variable objetivo es un valor numérico** continuo
* **CLASIFICACIÓN:** Los problemas de Clasificación son aquellos en los cuales **la variable objetivo es una categoría** o clase.

**APRENDIZAJE NO SUPERVISADO:**

En este caso tenemos observaciones, pero **no tenemos los resultados de** ninguna **variable a la que quisiéramos pronosticar**, por ende, el programa recibirá los datos y buscará similitudes entre los datos e irá formando grupos de similitudes.

El aprendizaje no supervisado **se suele utilizar para crear nuevas variables que podemos agregar a un problema de aprendizaje supervisado**. Uno de los métodos más usados es el **clustering**, con estas técnicas podemos descubrir otras formas de agrupar.

**Aprendizaje Reforzado (Reinforcement Learning):**

En este caso se presentan observaciones sin etiquetar, como en el caso del aprendizaje no supervisado, pero contamos con feedback o realimentación del entorno que nos indica si otra variable fue correctamente pronosticada o no.

1. **UNIDAD 2: ORANGE**

El programa que utilizaremos en la materia será el Orange3 de Anaconda, y ahí realizaremos nuestros futuros árboles y pronósticos.

**¿Cómo desarrollamos un programa en Orange3?**

Primero cargamos los datos, para eso debemos cargar los datos con la herramienta **file** que se encuentra a un costado. Una vez seleccionado los datos debemos determinar qué tipo de datos son cada variable, por ejemplo, seleccionar si es variable tipo texto, que nomás serviría para dar nombres, tipo categórico, numérico y el target que es lo que queremos diagnosticar a futuro. Una vez seleccionado crearemos una tabla con estos datos. De esta tabla generaremos un **data sampler**, que sería dividir los datos del **train set** y del **test set**, que luego desarrollaremos mas que es cada cosa. Del **train set** generaremos otro **data sampler** y de este sacaremos el **validation train set** y el **validation test set.** Con el **validation train set** probaremos diferentes profundidades en los árboles y elegiremos la que nos dé un mejor **AC**, que lo veremos generando una herramienta llamada **test and score** que le conectaremos a esta todos los arboles, el **validation test** y el **validation train**. Una vez encontrado el árbol mas optimo usaremos esa profundidad para generar un árbol que entrenaremos con el **train set** y el **test set**, otra vez llamaremos al **test and score** y el valor de nuestro **AC** será la precisión de nuestro programa. Por ultimo entrenaremos al programa con el 100% de los datos y a ese lo usaremos para evaluar futuros pronósticos.

1. **UNIDAD 3: EL PROBLEMA CON LOS DATOS**

En Machine Learning el objetivo es aprender a partir de los datos y si entra basura, saldrá basura, tomándose como basura los problemas que pueden tener los datos al momento de ser cargados.

problemas que se nos pueden presentase en los datos:

* **Errores en los dispositivos de medición**: Ejemplo: una variable que deseamos medir es la altura, puede ocurrir que comencemos midiendo con una cinta y después usemos otra, dando resultados distintos. Otro caso, usando la misma cinta, pero las mediciones son llevadas a cabo por distintas personas, quizás los resultados nos darán distintos.
* **Ruido**: debe interpretarse como interferencia producida generalmente por algún elemento distinto al instrumento de medición.
* **Observaciones con valores faltantes**: por diversos motivos puede que algunas observaciones estén incompletas.
* **Datos mal cargados**: Pueden darse problemas similares al utilizar unidades de medidas distintas para valores de la misma columna. Observe:



* **Datos inconsistentes**: En la tabla a continuación los primeros dos registros, tienen los mismos valores para las variables, pero en un caso la variable a pronosticar tiene valor Si y en el otro No. Esos podrían ser inconsistentes porque por ejemplo alguien se equivocó al cargar los datos o nos falta utilizar alguna variable más que "separe" estas dos observaciones.

****

* **Datos escasos**: Recordemos el caso del médico que nos pedía que con los datos del análisis de sangre pronosticáramos si el paciente tendría o no determinada enfermedad, el médico consiguió datos de mil pacientes. Supongamos que el médico era de la Ciudad de Córdoba, ahí viven más de un millón de personas, por ende, contaríamos con información del 0.1% de los casos posibles en ese momento.

A no ser casos excepcionales, **nuestros datos siempre corresponderán a una muestra del Universo posible de datos**, y **siempre habrá casos fuera de nuestros datos que presenten algún patrón de comportamiento distinto** a los que forman parte de los que poseemos.

Nuestros pronósticos nunca darán resultados perfectos al aplicarse a la población, nuestras observaciones no abarcan todos los casos posibles y por lo tanto las predicciones que pueda hacer nuestro modelo estarán sujetas a errores al aplicarlas a casos no observados del resto de la población. Por lo tanto, nunca se debe entrenar a nuestro modelo y no testearlo con respecto al resto de la población.

**¿cómo podemos saber cómo se comporta nuestro modelo con los casos que no tenemos a disposición?**

Con las siguientes acciones:

**TESTEO DE HIPÓTESIS (TESTING) en Aprendizaje Supervisado**

Existen diversos métodos aplicables al Aprendizaje Supervisado; el que vamos a investigar se usa comúnmente, existen variantes para diversos casos, pero mantienen la misma idea:

* Entrenaremos a nuestro modelo con el **Train Set**, y con el elaboraremos nuestro modelo o hipótesis
* Testearemos al modelo con el **Test Set**, que son datos que no formaron parte del entrenamiento
* Nunca debemos **testear** nuestro modelo con los datos de **Train Set**
* Utilizaremos algún criterio previamente establecido para medir qué tan bien pronosticó

**¿De dónde saldrán los datos del Test y Train Set?**

Los dos saldrán de la división del Conjunto de Datos Original en dos partes:

* La mayor parte (70%, 80%, etc) las utilizaremos como **Train Set**
* El restante (30%, 20%, etc) lo reservaremos y formará el **Test Set**
* Esta división suele hacerse en forma **aleatoria**
* Las proporciones entre 66%/33% a 80%/20% son valores **razonables** para la mayoría de los casos

1. **Unidad 3: Medidas de Evaluación de problemas de Clasificación**

Entrenado nuestro modelo con el Train Set se tiene que evaluar qué tan bien se comportará frente a los casos en los que no ha sido entrenado, para ello usamos el Test Set.

Como conocemos los resultados de cada observación podemos correr nuestro modelo y ver si los resultados pronosticados aciertan o no, de esta manera podremos medir qué tan bien o mal funciona nuestro modelo.

Existen diversas maneras de medir esta performance:

**Confusion Matrix:**

supongamos que tenemos un problema y nuestro target son dos, SI y NO, y nuestro total de observaciones son 1235. Cuando pronosticamos con nuestro modelo y lo testeamos contra el Test Set pueden ocurrir 4 situaciones:

* Que el valor real sea Si y nuestro modelo pronostique Si (un acierto para el modelo)
* Que el valor real sea Si y nuestro modelo pronostique No (error para el modelo)
* Que el valor real sea No y nuestro modelo pronostique No (acierto para el modelo)
* Que el valor real sea No y nuestro modelo pronostique Si (error para el modelo)

Esa información se suele resumir en forma de matriz y se denomina "Confusion Matrix".



De esta manera:

* la primera fila corresponde a las observaciones en las cuales el valor real es No
* la segunda fila corresponde a las observaciones en las cuales el valor real es Si
* la primera columna corresponde a las observaciones en las cuales el valor pronosticado es No
* la segunda columna corresponde a las observaciones en las cuales el valor pronosticado es Si

La matriz se lee de la siguiente forma:

* De las 1100 observaciones que tenían valor real No, nuestro modelo pronosticó como No a 1024 (bien pronosticadas)
* De las 1100 observaciones que tenían valor real No, nuestro modelo pronosticó como Si a sólo 76 (mal pronosticadas)
* De las 135 observaciones que tenían valor real Si, nuestro modelo pronosticó como No a 68 (mal pronosticadas)
* De las 135 observaciones que tenían valor real Si, nuestro modelo pronosticó como Si a 67 (bien pronosticadas)

**AUC (Area Under Curve):**

lo veremos más adelante

**CA (Classification Accuracy o exactitud):**

Una de las métricas más intuitivas y más usadas para evaluar modelos de clasificación. Su forma de cálculo solo es contar la cantidad de observaciones bien hechas y dividirlas sobre el total de las observaciones. Este valor siempre estará comprendido entre 0 y 1, siendo mejor cuando se acerca a 1. A través de la Matriz de Confusión podemos calcular la exactitud.

¿Cuáles son los aciertos?

* Los que eran No y nuestro modelo clasificó como No
* Los que eran Si y nuestro modelo clasificó como Si

**F1:**

Métrica más completa que la exactitud para evaluar la performance de nuestro modelo, está comprendido entre 0 y 1 siendo mejor cuando se acerca a 1

**Precisión:** lo veremos más adelante

**Recall:** lo veremos más adelante

**Continuación con los arboles de decisión:**

Hasta el momento solo vimos 4 pasos para armar el árbol, que son dividir los datos en dos (test y train), encontrar la profundidad en el train y luego testearlo con el test. Ahora nos falta el quinto paso, que será repetir el proceso de crear modelos de árbol, pero con diversas profundidades y luego evaluar cual modelo será el que mejor funcione. Para evaluarlos prestaremos atención en al valor de F1 y CA.

**El problema del Overfitting:**

A medida que vamos aumentando el nivel de profundidad del árbol, el modelo se va pareciendo más al **Train Set** y al principio va pronosticando mejor en el "resto del mundo", pero en algún momento comienzan a aparecer las diferencias y nuestro modelo comienza a adaptarse muy bien al **Train Set,** pero se comienza a apartar del "resto de mundo". Este problema se produce cuando nuestro modelo se parece más al **Train Set** que lo que debería, se denomina **OVERFITTING** lo que podemos traducir como **SOBREAJUSTE**.

**Conclusión:**

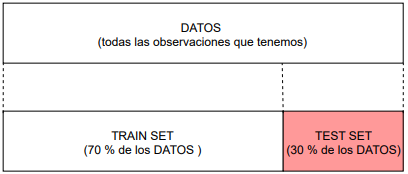
Nuestro interés es que el modelo pronostique lo mejor posible en las observaciones desconocidas, ahora que conocemos el **overfitting** no será conveniente generar un árbol de profundidad infinita, sino que deberemos entrenar con el Test Set árboles de profundidad creciente hasta encontrar aquel que nos provea el mejor valor de F1 testeando en el Test Set.

Cuanto más profundo es el árbol, más se parecerá al limitado Train Set. Ejemplo: encontramos que el F1 aumenta hasta un nivel de profundidad de 4, luego baja y luego sube nuevamente hasta alcanzar un valor máximo similar al 4 en el nivel 15; preferiremos adoptar el árbol de nivel 4, porque generalizará mejor en la realidad.

1. **Unidad 3: ENTRENAMIENTO, TUNNING, Y EVALUACIÓN DEL MODELO**

Para resolver un modelo hay que dividir el Dataset en dos partes: Train set y test ser, con el objetivo de que nuestro modelo aprenda del Train Set y evalúe en el Test Set.

La idea es que el Test Set represente todas las observaciones que existen en el Universo y no forman parte las que conocemos en el momento de entrenamiento, por lo que no debemos permitir que nuestro modelo aprenda del Test Set. Este solo debe servir para evaluar cómo se comportará nuestro modelo frente a las observaciones que no conocemos.



**HIPERPARÁMETROS**

Todos los modelos en machine learning tienen hiperparámetros. En el caso de los árboles de decisión los hiperparámetros son:

* profundidad del árbol
* cantidad mínima de observaciones
* el método que utilizamos para efectuar las divisiones en el árbol

Antes de entrenar a nuestro modelo tenemos que seleccionar qué valores de los hiperparámetros vamos a utilizar.

¿habrá algunos valores de estos hiperparámetros con los cuales nuestro modelo funcione mejor? los procesos para determinar los mejores valores para los hiperparámetros se denominan **TUNNING**.

Para determinarlos podríamos pensar en:

* Seleccionar distintos valores para los hiperparámetros
* Entrenar con el Train Set
* Testear frente al Test Set
* Repetir la operación con varias combinaciones de hiperparámetros
* Elegir aquellos con los cuales obtuvimos mejores resultados en el Test Set.

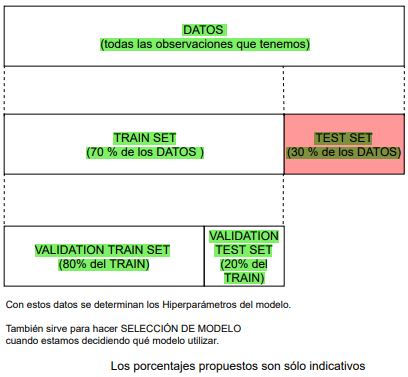
El procedimiento que veníamos haciendo tiene un problema: Si estamos seleccionando los valores para los hiperparámetros al ver qué resultados obtenemos en el Test Set, nuestro modelo está aprendiendo del Test Set y dijimos que el Test Set se utilizará sólo para testear.

**¿Cómo resolvemos este problema?**

**VALIDATION SET**

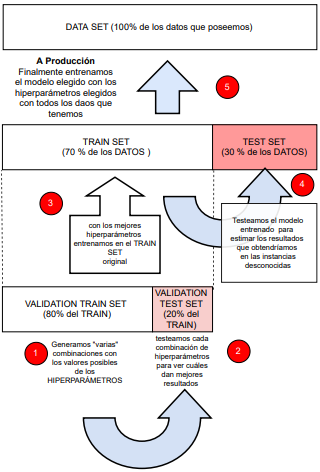
Para usar los validation dejamos de lado el Test Set. Por lo que cualquier cosa que queramos hacer para determinar el valor de los hiperparámetros deberá ser hecha con observaciones que no provengan de él. Por lo que nos quedaría solo para trabajar con el Train Set.

A este lo dividiremos en dos conjuntos

* VALIDATION TRAIN SET
* VALIDATION TEST SET

**¿Cómo procederemos?**

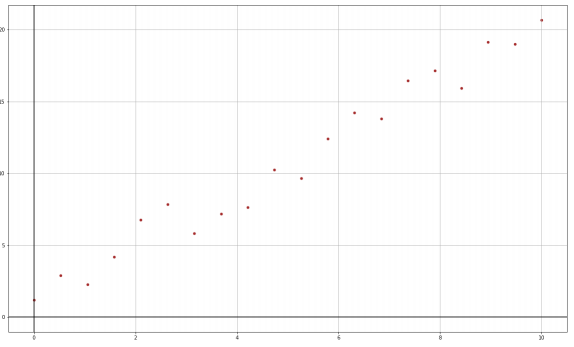
* Utilizaremos el **VALIDATION TRAIN SET** para entrenar a nuestro modelo con distintas combinaciones de valores para los hiperparámetros
* A cada una la testearemos frente al **VALIDATION TEST SET**, calculando Accuracy y F1
* Elegiremos la combinación que obtuvo mejores resultados y con ella armaremos nuestro modelo.
* Encontrado los mejores valores con el hiperparámetro lo entrenaremos con **TODO** el **TRAIN SET** y lo evaluaremos con el **TEST SET**.
* Por ultimo poner nuestro modelo el **Producción** lo entrenaremos de nuevo con todos los datos que tenemos en el **Data Set original**.



1. **Unidad 4: REGRESIÓN LINEAL con UNA VARIABLE**

Las situaciones correspondientes a **Regresión** es cuando la variable que deseamos pronosticar es un **valor numérico**.

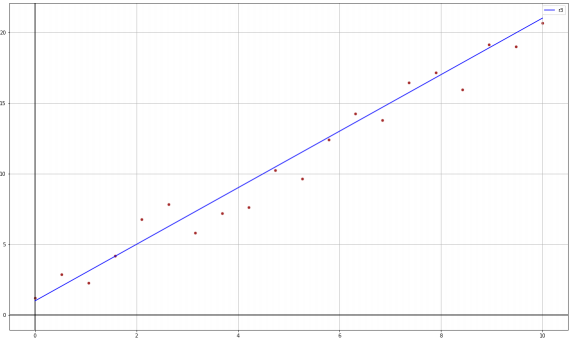
Supongamos que tenemos una variable numérica (**x**); y una variable que deseamos pronosticar numérica (**y**). Podemos mostrar estos datos como:



Al mirar la distribución de los puntos da la impresión de que están alineados, peor por diferentes rectas como candidatas a ser la que mejor represente los puntos, por ejemplo:



Observemos una cualquiera de las otras tres:



Es una recta bastante representativa de los puntos, lo cual quizá la verdadera relación que habría entre **x** e **y** sería una línea recta perfecta y los puntos no se observan perfectamente alineados.

**¿A qué podría deberse?**

* La relación entre los valores de x e y no fueran una línea recta
* Solemos atribuir estas diferencias a algún tipo de error

El error es debido al azar o a diferencias de percepción del operador y algunas de las veces que medimos el valor que obtuvimos era mayor que el real y otras veces era menor, por lo que el valor real debería ser cercano al **promedio**, denominado valor esperado.

Se suele expresar esto diciendo que los valores que observamos de **y** no son los valores reales, sino que son los valores reales afectados por el error y la mejor aproximación que tenemos de los reales son los valores esperados: 

En el grafico anterior mostrado si graficáramos  los puntos se encontrarían alineados.

Lo que nos interesa es encontrar una expresión matemática para  denominada función **hipótesis**: **h(x)**, correspondería a la recta que mejor aproximaría a los puntos que poseemos.

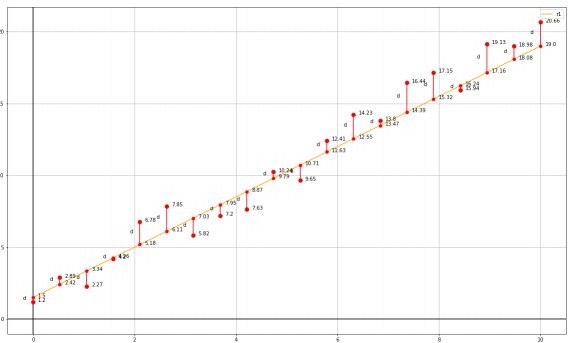
Hemos visto que la ecuación de una recta era: **y = a\*x + b**, donde **a** es la pendiente y **b** es la ordenada al origen.

Nuestra función hipótesis la escribiremos como: **h(x) = w0 + w1x**, donde **w0** cumple el rol de ordenada al origen y **w1** es la pendiente. Nuestro objetivo es encontrar la recta, **h(x) = w0 + w1x,** que mejor aproxime a los puntos que tenemos como datos, es decir que queremos **encontrar** los valores de w0 y w1 para formar la ecuación de dicha recta. Por lo cual los parámetros w0 y w1 son las **incógnitas** en nuestro problema.

**¿qué entenderíamos cuando decimos "que mejor aproxime"?**

Una alternativa sería calcular las distancias que separan a nuestros puntos de una de las rectas. Veamos:

Las distancias entre una recta y los puntos que poseemos se podría medir **verticalmente**:



Para cada punto conocido podríamos calcular la distancia entre ese punto y la recta, que sería lo que mide cada uno de los segmentos rojos, llamados **di**, luego sumar todas estas distancias para cada uno de los puntos conocidos

Donde **m** es la cantidad de puntos conocidos u observaciones, obtendríamos un valor que representaría el "error total" que estaríamos cometiendo con esa recta. Si lo dividimos en la cantidad puntos que tenemos, obtendríamos el **Error Medio** cometido por esa recta.

Si repetimos esta operación para todas las rectas que queremos evaluar, podríamos elegir la que tuviera el menor Error Medio.

Aclararemos que **la distancia no puede ser un valor negativo**. A partir de las figuras anteriores podemos calcular la distancia vertical entre cada punto de la tabla y la recta h(x) para cada valor de xi de la siguiente manera: **yi − h(xi)**

Habría un problema: para algunos valores de xi daría positivo, cuando el punto (x i, y i) queda por sobre la recta, pero daría negativo, cuando el punto (x i, y i) queda debajo de la recta, y la idea de distancia no puede ser negativa.

Podemos resolverlo tomando el valor absoluto de esta diferencia: **|yi − h (xi)|**, el valor absoluto de un número se define como "la parte positiva del número", es decir el valor absoluto de cualquier número positivo o el 0, es el mismo número y en el caso de que sea un número negativo, su valor absoluto sería positivo.

Entonces el Error en todos los puntos correspondería a sumar el error cometido en cada uno de ellos:

El **Error Promedio Absoluto (MAE: Mean Absolute Error)** lo podríamos calcular como:  y se ajustaría a nuestro criterio.

Para cada recta posible calcular errores y nos quedaríamos con la recta que tuviera el menor valor, lo cual es imposible hacer de a una sí es posible hacerlo, pero con otros métodos: el método de maximización o de minimización que provienen de la potencia del Cálculo.

Para que estos métodos se puedan poner en práctica es necesario que la función que deseamos minimizar sea derivable, cosa que la nuestra no cumple por el valor absoluto introducido en nuestra fórmula.

Deberemos buscar otra operación que no sea el valor absoluto que nos "transforme en positivo" el resultado de la resta **yi − h (xi)**, la operación de **elevar al cuadrado** siempre da por resultado un número no negativo y nos dará una fórmula derivable a la que se le podrán aplicar los métodos del Cálculo.

Fórmula para calcular el error:

se la suele denominar **Error Cuadrático** y es la forma más habitual de calcular el error en problemas de Regresión Lineal.

Recordemos que nuestro objetivo es elegir los **parámetros** w0 y w1 de la recta con el **mínimo valor** para el Error Cuadrático de la fórmula anterior.

A este método se denomina de los **MÍNIMOS CUADRADOS (Least Squares)**

****El Método de los **Mínimos Cuadrados** se minimiza: es decir que se buscan los valores de w0 y w1 que hacen que la sumatoria de por resultado el menor valor posible, y es la más utilizada para los casos de Regresión Lineal.

**Métricas para Evaluación: Error Medio Cuadrático, Raíz del Error Medio Cuadrático y R 2**

obtenidos los valores de los parámetros que optimizan nuestro modelo tenemos que evaluar qué tan bien ajusta nuestro modelo a los datos que poseemos, en este aspecto el Error Cuadrático presenta algunos problemas.

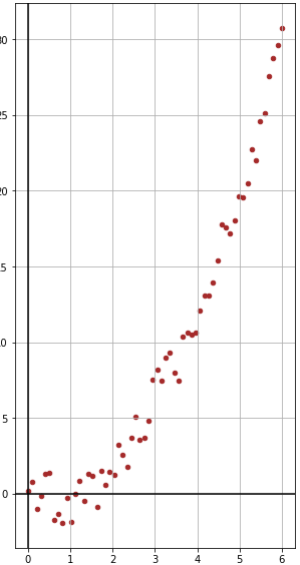
1. Cuantos más datos tengamos, mayor será el resultado del Error Cuadrático. Podemos solucionarlo calculando el **Error Cuadrático Medio (MSE: Mean Square Error)** dividiendo el error cuadrático por la cantidad de observaciones, esto nos dará un valor del error promedio.
2. **Incómoda interpretación del valor del error:** supongamos que la variable que estamos pronosticando fuera la altura de las personas medida en centímetros en función de x. Al calcular el Error Cuadrático Medio la unidad del resultado correspondería a las unidades de y al cuadrado, podría ser de 36 centímetros cuadrados, ¿qué significa un error de 36 cm2 al pronosticar la altura de una persona? Puede solucionarse tomado la **raíz cuadrada al Error Cuadrático Medio (RMSE: Root Mean Square Error)** 

Para el ejemplo anterior el **RMSE**, en vez de 36 cm2, nos hubiera dado un error de 6cm, más fácil la lectura. El **RMSE** es el valor que se suele utilizar para medir qué tan bien ajusta nuestro modelo a los datos. También es muy utilizado para evaluación del modelo el **MAE**

Otra métrica utilizada es **R2** o **Coeficiente de Determinación**. Tiene origen en la Estadística y es un número comprendido entre 0 y 1 y cuanto más cercano a 1 mejor será.

**Otros Modelos Lineales con respecto a los parámetros w**

Observe el siguiente gráfico, que la relación entre los valores de **y** con los valores de **x no es una recta**, por lo cual no sea una buena idea aproximarla con una hipótesis donde h(x) sea una recta.



Podríamos pensar en otro tipo de función para h(x), como las **cuadráticas**: **h(x) = w0 + w1x + w2x²**, por lo cual la función no solo tiene que ser un polinomio de grado 1, si no que puede ser hasta **h(x) = w0 + w1x + w2x² + w3x 3 + . . . + wnx n**

**Comentario con respecto al grado del polinomio**

En el mundo real generalmente se utilizan polinomios de grado bajo, esto ocurre por 2 motivos:

1. **Complejidad**: Es raro pasar de un polinomio de tercero o cuarto grado la realidad no suele ser tan compleja como para requerir un polinomio de grado 20
2. **Overfiting**: cuanto mayor sea el grado del polinomio que utilicemos, aumenta el riesgo de cometer overfiting

**¿Qué pasa con la linealidad?**

Suelen confundirse dos expresiones referidas a la linealidad. Es muy habitual que a las rectas también se las denomine funciones lineales o polinomios de 1er grado, lo cual es correcto. Ejemplo la expresión funcional suele ser: **y = ax + b**

Lo que no se suele aclarar es que *y* es una función lineal con respecto a la *x*. Para que la relación entre la *y* y la *x* se diga que es lineal a lo sumo la x puede aparecer a la primera potencia y multiplicada por un número. Cuando **y = ax** la relación entre la *y* y la *x* es lineal. Ahora bien **y = ax 2**la *x* aparece al cuadrado, por lo cual ya no existe una relación lineal entre la *y* y la *x*, ya que dijimos que la x no puede estar elevada a otra potencia más que 1.

Tampoco la habría en los siguientes casos:

* y = 4x 3 + x\*b
* y = sen(x) + b
* y = a\*ex
* y = √x
* y = a/x

Cuando planteamos nuestra hipótesis h como: **h(x) = w0 + w1x** planteamos una relación lineal entre la *h* y la *x*. ¿Qué pasa cuando plantemos que la forma h es polinómica? Ahora la relación entre h y x ya no es lineal.

**¿Por qué hemos dicho que podemos usar una expresión polinómica para h si estamos estudiando Regresión Lineal?**

Respuesta: cuando hablamos de Regresión Lineal no nos estamos refiriendo a una relación lineal entre los valores de h con x, sino a la relación entre los valores de h con los **parámetros** w del modelo

Observemos: **h(x) = w0 + w1x + w2x² + w3x 3 + . . . + wnx n**, h es lineal con respecto a w1 ya que este parámetro está elevado a la primera potencia y multiplicado por un número, lo mismo ocurre con el resto de los parámetros, por lo que **h** es lineal con respecto a los parámetros, aunque no es lineal con respecto a x; cuando se dice que el modelo es de **Regresión Lineal** **se refiere a que sea lineal con respecto a los** **parámetros**, no con respecto a la x

Para poder aplicar el método de Mínimos Cuadrados es necesario que **el modelo sea lineal con respecto a los parámetros w**