**Notes intro to machine learning**

**Terminología**

**Inteligencia artificial:** se refiere a la capacidad de una máquina de realizar tareas que requieren de la inteligencia de un ser humano, haciéndolas al mismo nivel o mejor.

**Machine learning:** es el subcampo de ai donde las maquinas tienen la habilidad de realizar una tarea específica, sin necesidad de programarlas explícitamente.

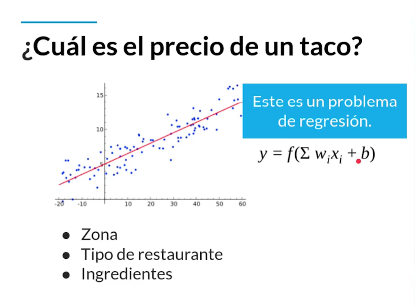
Tipos de ML

* Supervised learning (aprendizaje supervisado): se caracteriza por tener etiquetas, para poder realizar por ej clasificaciones
* Unsupervised learning (aprendizaje sin supervisión): en este caso no hay etiquetas sino solo inputs. En este caso es posible agrupar y buscar patrones
* Reinforcement learning (aprendizaje por refuerzo autónomo): en este caso hay una recompensa o varias las cuales me servirán para buscar y maximizar mis resultados (en función del premio). Es aprendizaje basado en experiencia.

**Deep learning:** es un tipo de ML donde se usan redes neuronas

**Artificial neuronal network:** es un sistema de software construido por nodos interconectados. Puede ser una transformación lineal ( peso + bias) o no lineal (función de activación)

**Terminología y regresión lineal**



y: indica el precio del taco 🡪 output 🡪 lo que estamos prediciendo

se tiene una sumatoria dado que son varias entradas

w: es el peso/importancia

x: entradas (zona, tipo de restaurante, ingredientes) 🡪 feature

b: bias, es un sesgo, un corrimiento, para ubicar los datos en el contexto donde esta

ahora miremos otro caso





para llevar acabo este sistema debemos tener

* Modelo: define una relación entre features y labels

Y = mx +b

Y’ = b + wx

* Training: darle un dataset al modelo para que aprenda
* Inference: usar el modelo para realizar predicciones.

**Construyendo un modelo**

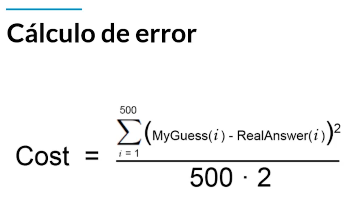
Precio = habitaciones \* w0 + tamaño \* w1 + ubicación \* w2



Luego de tener el modelo se pasa al **proceso de aprendizaje**, donde se realizan varias iteraciones para reducir el loss y poder acercarnos a un valor cercano.

¿Y cómo sé que tan cerca estoy del valor que estoy buscando (sale Price)?

Calculando el costo

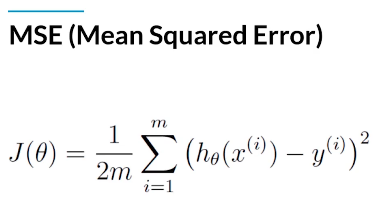
****

Se supone hay 500 filas

Calculamos el error al cuadrado, en lugar de error simple, para que el error siempre sea positivo y también se hace para castigar los errores más grandes.

Si no se eleva al cuadrado unas veces el error daría positivo o negativo. Otra opción es usar el valor absoluto, sin embargo, tendríamos una función no derivable y es importante tener una para hacer posible el uso de algoritmos de optimización como por ej: el descenso del gradiente.

Esto es lo mismo que

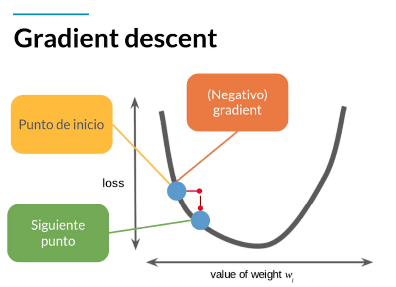


**Luego necesitamos minimizar el error**

Para eso usamos el gradiente, el cual es un vector (magnitud y dirección) y lo vamos a calcular con una derivada parcial.

La derivada parcial nos va a servir para encontrar el gradiente y saber en qué dirección movernos

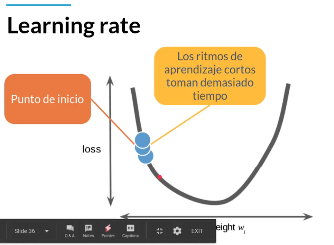
Esta derivada parcial la vamos a calcular sobre las variables (entradas)

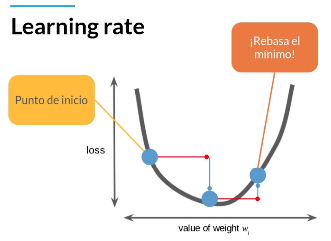


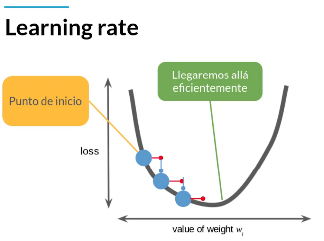
Queremos llegar al mínimo, en cada iteración obtenemos la dirección en la cual debemos movernos para llegar al objetivo.

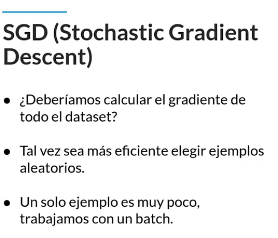
Cada paso tiene un tamaño y depende de:

**Learnign rate ( tasa a la que el modelo va aprender):** indica que tan grande o pequeños son los pasos que estoy dando



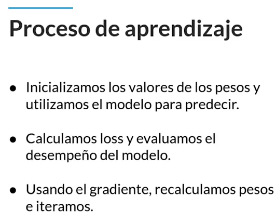
 este no sirve

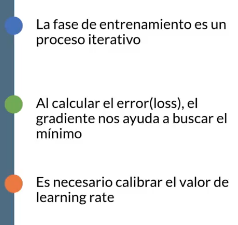
 esta es la idea



Un batch es un punto intermedio, un grupo del dataset donde vamos a calcular el gradiente y poder movernos al mínimo.

En resumen





**Pytorch**

Pytorch es un framework para ia donde la ejecución es inmediata (cuando trabajamos con ml se construye el árbol y este se ejecuta al mismo tiempo)

Torchcry es una herramienta donde se construye el árbol y luego se ejecuta

Actualmente hay 2 frameworks principales para ML 🡪 tensorflow y pytorch

Proceso de aprendizaje en pytorch

* Forward pass 🡪 predicción
* Backpropagation 🡪 proceso donde probamos y nos devolvemos hasta obtener los pesos adecuados
* Optimización

**Tensores**

Un tensor es una generalización, una estructura de datos que me permite representar por ejm imágenes

**Stride()**

El método ‘.stride()’ nos permite conocer de qué forma se encuentra almacenada nuestra estructura de datos, es decir, si nuestros datos serán almacenados de forma vertical, la función nos retornará algo de la forma ‘(n,1)’, donde n es el número de parámetros que tiene cada fila (Cada fila es un dato); si por el contrario nuestros datos tienen una estructura de forma horizontal, la función nos retornará algo de la forma ‘(1,n)’, donde n es el número de parámetros que tiene cada columna (Cada columna es un dato).

**Dim**

**Eje en una suma**

x = torch.tensor([  
[1, 2, 3],  
[4, 5, 6]  
])

**dim = 0**

**suma por cols**

torch.sum(x, dim=0)

tensor([5, 7, 9])

**dim = 1**

**suma por filas**

torch.sum(x, dim=1)

tensor([6, 15])

**en numpy sucede lo mismo solo que se llama axis**

**Regresión Logística**

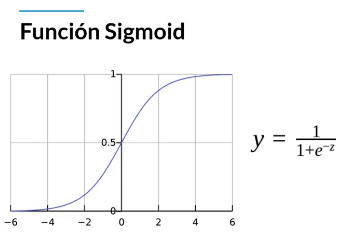
Es un mecanismo eficiente para calcular probabilidades. El resultado puede utilizarse tal cual o convertirlo a una categoría binaria (clasificar)

Se utiliza en problemas de

**Clasificación**

* Para resolver algunos problemas, es necesario predecir probabilidad.
* Spam vs no spam
* Aprobado vs reprobado.
* Rojo o azul

Para garantizar que el resultado este entre 0 y 1 se utiliza la función sigmoide, en casos de generalización se utiliza la fun softmax



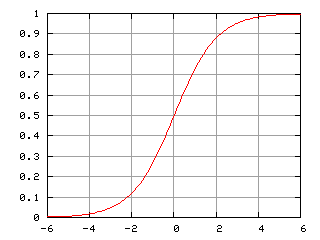
La función sigmoid transforma los valores introducidos a una escala (0,1), donde los valores altos tienden de manera asintótica a 1 y los valores bajos tienden de manera asintótica a 0.

**Características**

* Satura y mata al gradiente (¿?)
* Lenta convergencia
* No esta centrada en el cero
* Buen rendimiento en la última capa (¿?)

**Contextualización**

La función sigmoid incluye la función logística





Donde p representa la población, t el tiempo

Esta función matemática aparece en diversos modelos de crecimiento de poblaciones, propagación de enfermedades epidémicas y difusión en redes sociales.

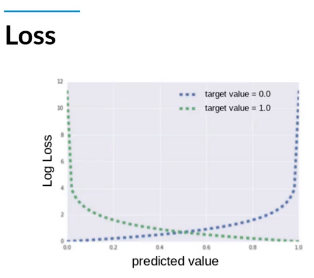
**Cerrando paréntesis**

Ahora pasamos del dominio continuo en la regresión lineal a el domino discreto en la regresión logística (haciendo clasificación binaria)

El **Loss** debe cambiar

Si espero un valor cercano a 0 y me da un valor alto debo castigar

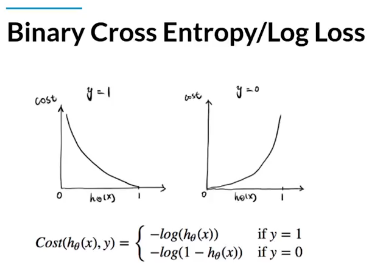
Si espero un valor cercano a 1 y me da un valor bajo debo castigar



El logaritmo nos permite modelar esto

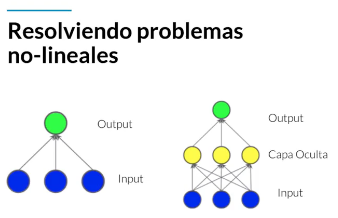
**Nota:** para problemas de probabilidad utilizamos una regresión logística

Para calcular el error nos basamos en la entropía, pero el gradiente sigue siendo útil.



**Neuronas y función de activación**

**Resolviendo problemas no lineales**

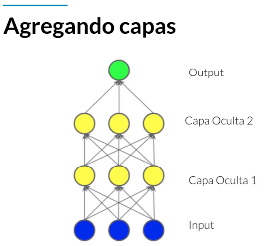


Para ello se van añadir capas ocultas

Estas capas ocultas nos van ayudar a resolver problemas de la no linealidad

En cada capa se agregan unos nodos (neuronas)

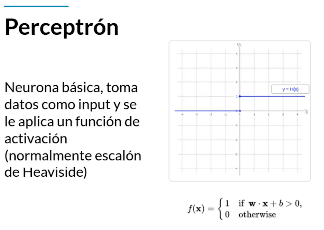
Se van agregar capas según las necesidades del problema, 12 (shadow) o miles (Deep)

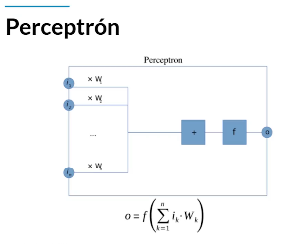


Cada neurona va a tener una función de activación y esta misma va a permitir hacer las conexiones

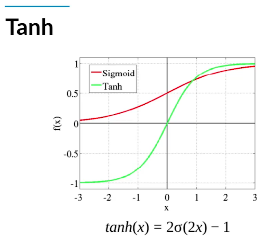
**Funciones de activación**

* Perceptrón: neurona básica, generalmente trabaja con el escalón de heaviside)

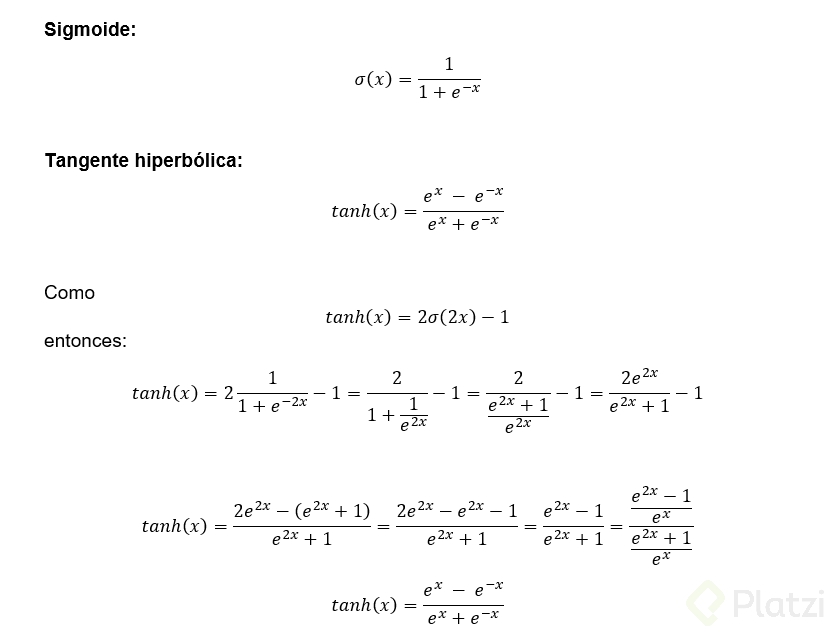




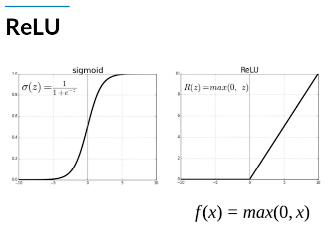
* Sigmoid
* Tanh: es un sigmoid ajustado, este va entre -1 y 1



**Desarrollo de ecuación**



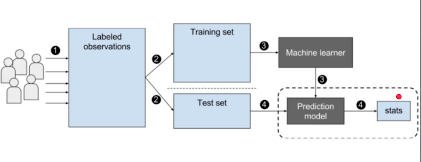
* Relu (rectificador lineal): evita el problema de vanishing gradient (cuando los gradientes se van haciendo muy pequeños). Solo se puede usar en capas ocultas



Podemos modelar un problema no lineal incluyendo capas ocultas para abstraer ciertos detalles, en cada capa oculta vamos a incluir múltiples neuronas, cada neurona va a incluir una función de activación

**Los algoritmos más usados en machine learning**

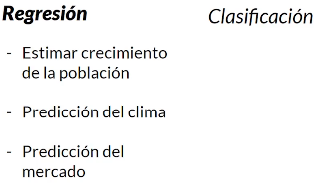
**Supervised Learning**



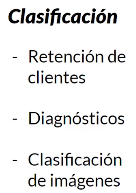
Las observaciones tienen que estar etiquetadas

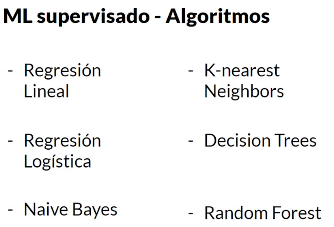
Stats: estadísticas

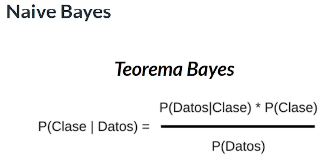
**Ejem regre**



Ejem clas

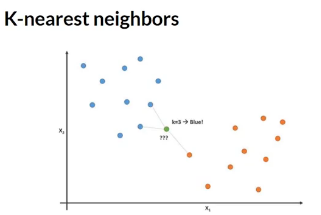






Es usado para clasificar, haciendo uso de las probabilidades

Este algoritmo agrupa los datos en base a los pesos, no por sus relaciones

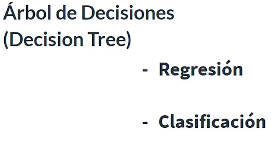


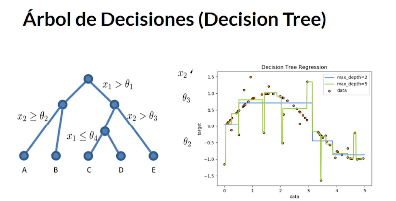
Este algoritmo permite hacer un predicción, cada vez que se mete un dato al sistema se puede hacer la predicción de cual es el grupo al cual debe pertenecer

Este algoritmo también se ocupa de hacer clasificaciones

Las formas de medir las distancias son:

* Distancia euclidiana
* Distancia Hamming
* Distancia Manhattan

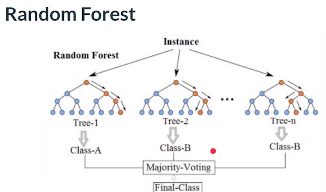




La forma de hacer clasificación/predicción se hace con los condicionales de programación

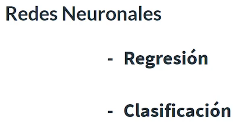
En la gráfica a la derecha se puede apreciar la ecuación representada en forma de grafica que representa el set de datos

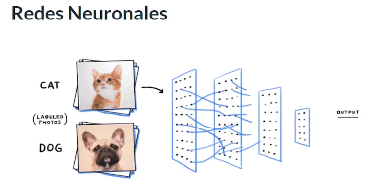
Si trabajamos con esta algoritmo hay que tener cuidado de no sobreajustarlo al modelo dado que si entra un dato nuevo no podría predecir bien, la idea es que se pueda generalizar



Este algoritmo sirve para clasificar

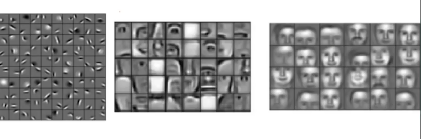
La magia del algoritmo esta en que genera una clase final para tomar la decisión final y que podemos tener control sobre la profundidad de los árboles



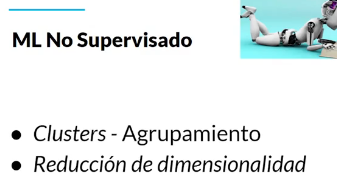


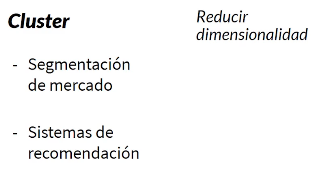
Las redes funcionan rebotando información entre capas

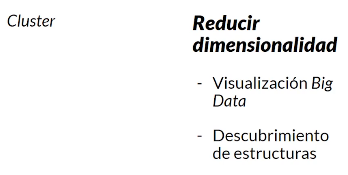
**Unsupervised Learning**

****

en este caso el algoritmo es el que da las etiquetas a través de la búsqueda de patrones



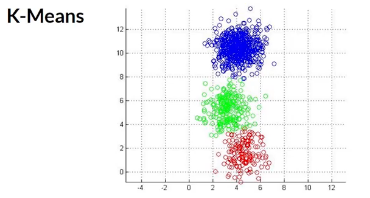




Es utliizado en big data porque tenemos muchas dimensiones (features \* targets )

Lo que busca es agrupar los features para tratarlos más facil en algoritmos supervidados o en redes neuronales

**Algoritmos**



**Que hay detrás de una red neuronal**

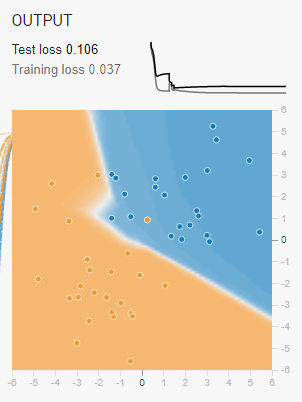
Batch size: es el tamaño del conjunto de datos usado en cada iteración

Ratio of training to test data: cantidad de datos de entrenamiento

Bias: es un parámetro de sesgo que proporciona grados de liberta al modelo, es usado en la combinación lineal de una neurona

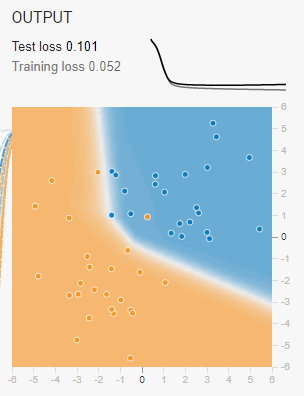
**Overfiting:** es un problema en machine learning, sucede por ejemplo cuando se esta haciendo el entrenamiento para clasificar y el modelo en el entrenamiento se sobreajusta/se adapta/acomoda muy bien al modelo, lo cual aparentemente es bueno pero no es así, porque cuando se haga la clasificación con los datos de entrenamiento no va a clasificar bien, la idea es que permita generalizar

Ejem



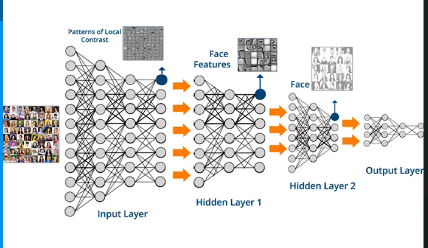
Para solucionar esto se utiliza las funciones de regularización

**Funciones de regularización**: son usadas para impedir que suceda el overfiting



Otra forma es para el entrenamiento cuando el loss este en un rango aceptable

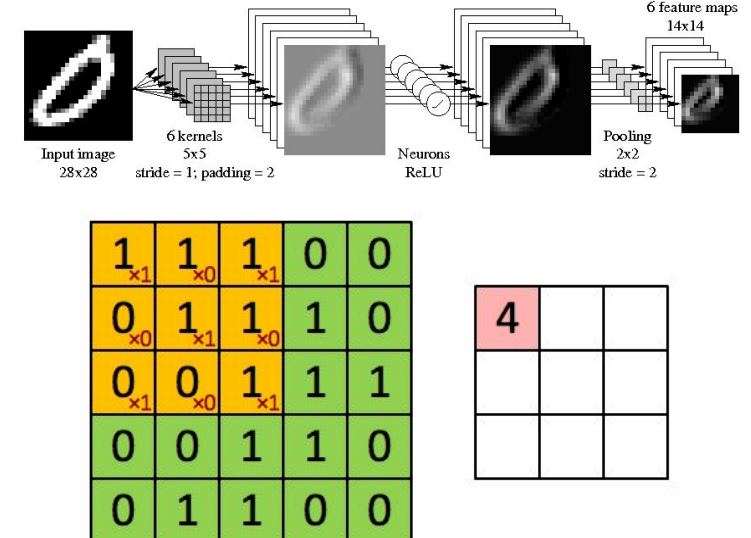
**Redes convolucionales y por que son tan buenas con imágenes**



Ejem - reconocimiento de caras: En las primeras capas va a detectar contrastes y en capas siguientes estas serán características (nariz, ojos, boca)

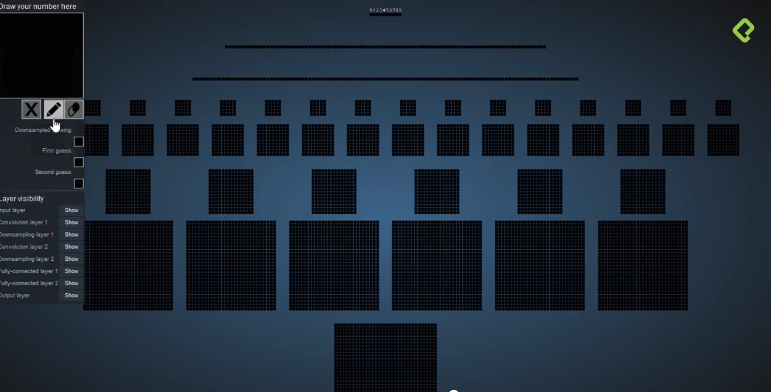
**Matemática detrás de las redes convolucionales**

El objetivo es encontrar patrones de contraste comparando el gradiente entre los pixeles, la arquitectura no solo debe tener presente el valor del pixel sino también su contexto al resto de pixeles

La convolución son productos matriciales para extraer las características de contraste 

el pooling lo que hace es comprimir la información para tratarla más rápido

**como funciona la convolución – forma gráfica**



Esta compuesta por

* una capa de entrada – parte inferior
* Una capa de convolución
* Una de pooling
* Una de convolución
* Una de pooling
* Una red neuronal por ejemplo para predecir -parte superior

Red convolucional: es una operación matemática de escaneo que se encarga de buscar patrones y características de las imágenes (bordes, narices etc)

**Redes generativas**

Son redes que generan imágenes a partir de otras

Auto encoder: es una red para reconstruir imágenes, esto se hace llevando la img a un espacio latente de características (características que componen la img inicial) en este punto la info esta codificada (para un mejor tratamiento), para hacer una app de este tipo necesitaríamos una img a la cual vamos añadir un feature. La última parte del auto encoder es el decodificador, el cual lleva la info codificada a un plano real.

**Nota:** sin embargo la mayoría de img falsas se generan con un red llamadas GANS (redes generaticas adversarias) en esta se enfrentan dos tipos de redes, la que intenta falsificar y otra que intenta verificar que no pase imgs falsas, ambas redes van mejorando en cada entrenamiento.

**Colab**

**Comandos**

* **Ejecutar una celda:** ctrl + enter
* **Ejecutar una celda y abrir otra:** shift + enter
* **Crear una celda:** ctrl + ( m b)
* **Eliminar una celda:** ctrl + (m d)

**Unknow words**

* Regresión lineal
* Normalización: es la transformación de escala de la distribución de una variable, esto con el objetivo de poder comparar elementos de diferentes variables, unidades de medida

En otras palabras son proporciones sin unidad de medida (adimensionales o invariantes de escala)

Def wiki: la normalización de índices significa ajustar los valores medidos en diferentes escalas respecto a una escala común

Def me: son transformaciones que me permiten hacer comparaciones con elementos de diferentes unidades

**Enfermedades en el café**

La gotera, mancha de hierro

Trabajar predicción de cosechas de café de acuerdo a históricos y variables,(temperatura, abono, época del año, región )