

# COMPUTACION BLANDA PRIMERA PREVIA

JUAN PABLO LOPEZ RAMIREZ

# Índice

1. Contenido del GitHub.
2. Introducción al Paper.
3. Resumen del Paper.
4. Lógica difusa.
5. Resumen de lo realizado en colab.
6. Webgrafía de la presentación.

# 1. CONTENIDO DEL GITHUB

1. Paper sobre Perceptrón y backpropagation.
2. PowerPoint.
3. Archivos de Colab.
4. Archivo de Excalidraw.



## **2. INTRODUCCIÓN AL PAPER**

**Nombre del Paper:** El Perceptrón, Entrenamiento del Perceptrón y Backpropagation.

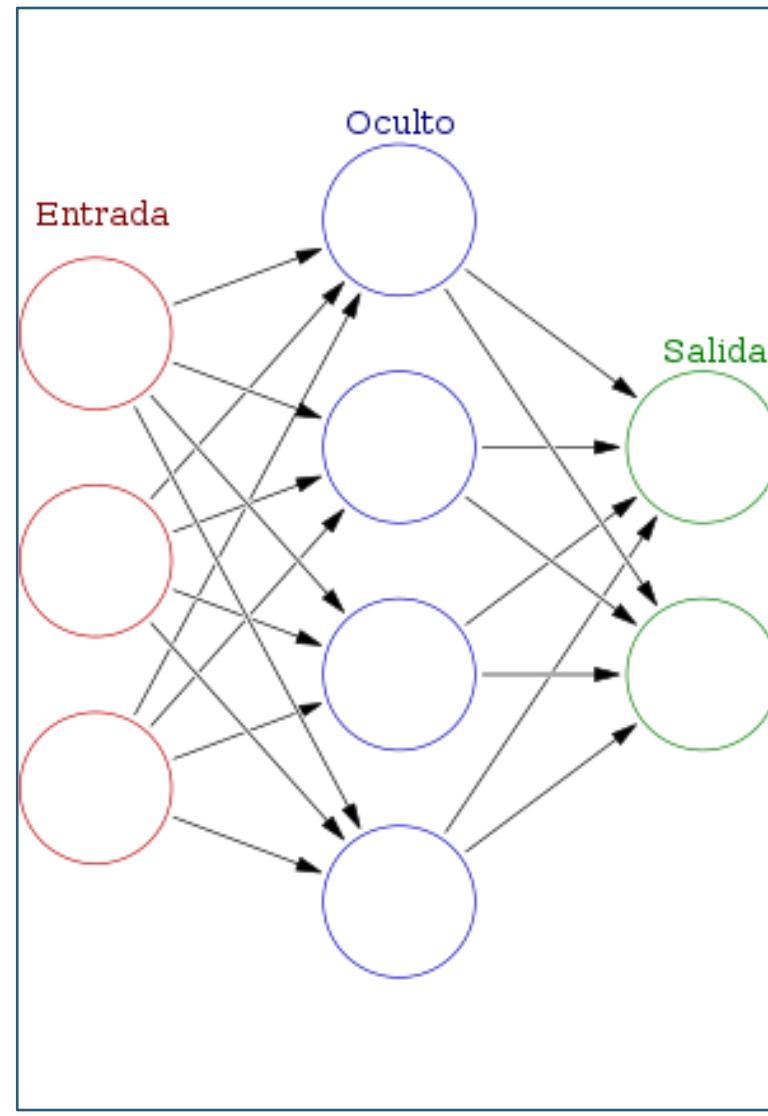
**Introducción:** En este paper podremos ver y leer sobre algunos temas como lo son; redes neuronales , el perceptrón y backpropagation.

A continuación en este documento de PowerPoint se hablara de aspectos importantes referentes al paper, para mas información leer el paper.

### 3. RESUMEN DEL PAPER

El contenido de este punto tratara los siguientes aspectos:

- ¿Qué es una red neuronal artificial?
- Ventajas de una red neuronal.
- Algunos usos de una red neuronal
- Introducción al Perceptrón.
- Partes de un Perceptrón.
- En que consiste el entrenamiento de un perceptrón.
- ¿Que es backpropagation?



# ¿QUÉ ES UNA RED NEURONAL ARTIFICIAL?

Las redes neuronales son un modelo inspirado en el funcionamiento del cerebro humano. Está formado por un conjunto de nodos conocidos como neuronas artificiales que están conectadas y transmiten señales entre sí.

Dicho de otra manera las redes neuronales artificiales (también conocidas como sistemas conexionistas) son un modelo computacional evolucionado que consiste en un conjunto de unidades, llamadas neuronas artificiales, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo unos valores de salida.

# VENTAJAS DE UNA RED NEURONAL

Algunas de las ventajas de una red neuronal son:

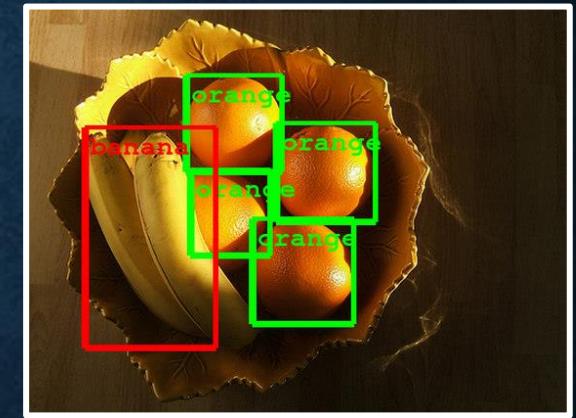
- Aprendizaje Adaptativo. Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.
- Flexibilidad. Una Red Neuronal Artificial puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (ej. si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente).

Además de estas 2 ventajas hay otras muy importantes como la auto organización o la tolerancia a fallos , aspectos que son muy importantes, razón por la cual hoy en dia el uso de las redes neuronales ha aumentado, haciendo que se usen en diversos campos o áreas, algunos de estos campos son el de las finanzas y la biología.

# ALGUNOS USOS DE UNA RED NEURONAL

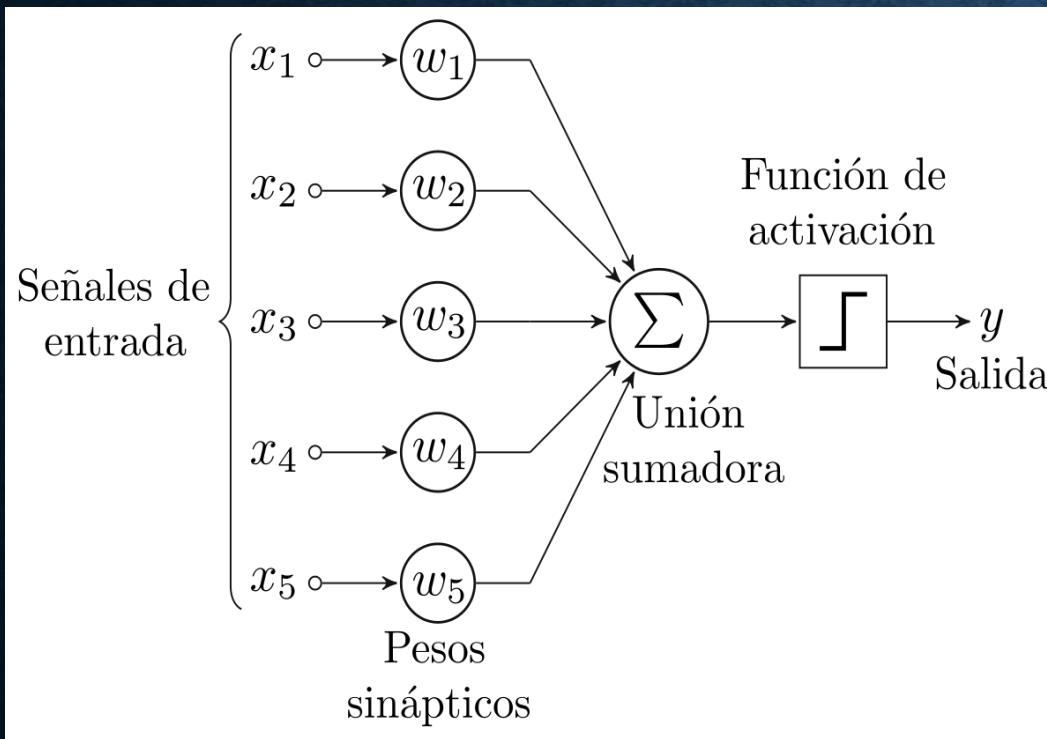
Las redes neuronales como lo mencionamos antes aportan muchas ventajas y son de mucha ayuda en algunas áreas, a continuación voy a decir algunas de las muchas aplicaciones que tienen las redes neuronales:

- Predicción de sucesos y simulaciones.
- Procesamiento de datos y modelización.
- Reconocimiento y clasificación.
- Control inteligente.



Todos estos aspectos sirven para que las redes neuronales se apliquen en el área de las finanzas o en la biología como lo habíamos dicho anterior mente. Gracias a las redes neuronales podemos crear programas de reconocimiento de imágenes, de reconocimiento de sonidos, reconocimiento de patrones entre muchas otras cosas.

# EL PERCEPTRÓN



En el campo de las Redes Neuronales, el perceptrón, creado por Frank Rosenblatt, se refiere a:

- la neurona artificial o unidad básica de inferencia en forma de discriminador lineal, a partir de lo cual se desarrolla un algoritmo capaz de generar un criterio para seleccionar un sub-grupo a partir de un grupo de componentes más grande.

# PARTES DEL PERCEPTRÓN

El perceptrón se compone por:

- Entradas: Es la información que recibe el perceptrón.
- Pesos: Son valores numéricos que se encargan de establecer la influencia de una entrada en la salida deseada.
- Bias: Es un parámetro que tienen algunos modelos de redes neuronales el cual permite encontrar fácilmente la separación entre posibilidades de salida de una red neuronal.
- Función de activación: Es una función matemática que se encarga de determinar un valor de salida una vez se han procesado cada una de las entradas.

# ENTRENAMIENTO DEL PERCEPTRÓN

Para comprender como se realiza el entrenamiento de una red de neuronas, lo primero que debemos de hacer es comprender el entrenamiento de una sola neurona, para luego acoplar varios entrenamientos individuales en uno colectivo para toda la red. A continuación se muestran las formulas que se usan normalmente para entrenar el perceptrón.

$$w_i \leftarrow w_i + \Delta w_i$$

vector de  
pesos asociado

$$\Delta w_i = \eta(t - o)x_i$$

Para cada iteración  
del algoritmo se  
actualizará el vector  
de pesos

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Función de Error

$$\nabla E(\vec{w}) \equiv \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$

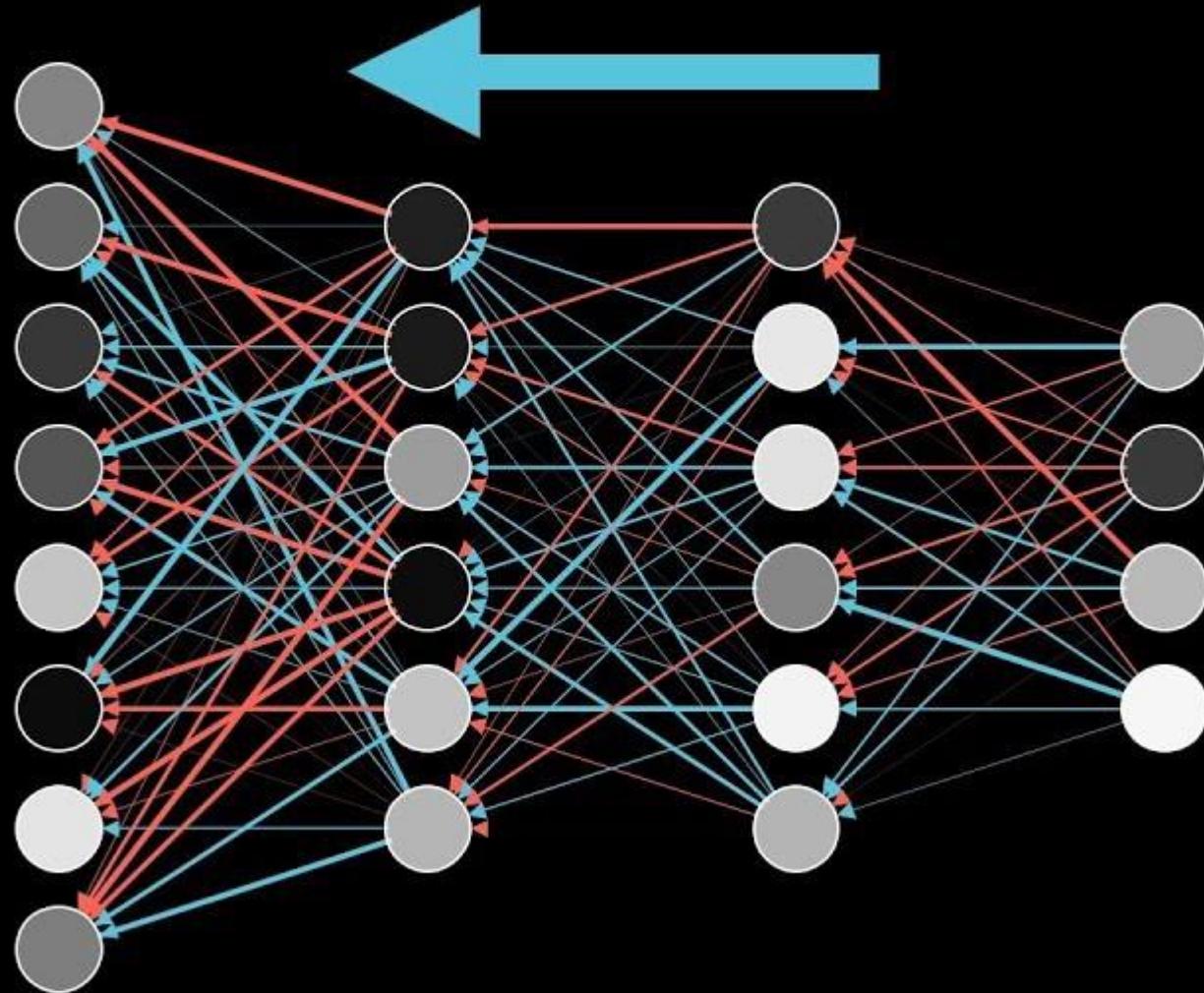
Gradiente  
Descendente

# ENTRENAMIENTO DEL PERCEPTRÓN

Pasos para el entrenamiento:

- Para el entrenamiento de un perceptrón lo primero que debemos hacer es inicializar aleatoriamente su vector de pesos asociado, y después ir actualizando este para conseguir mejores resultados.
- Siendo  $\eta$  un parámetro al que denominaremos tasa de entrenamiento con un valor positivo y que hará la función de hacer que la convergencia sea más o menos rápida. Es recomendable que la tasa de entrenamiento sea pequeña (entre 0.1 y 0.2) para que los resultados sean correctos.
- Aplicar la función de error. La función de error mide de una manera numérica la diferencia existente entre la salida ofrecida por la red y la salida correcta.
- Usamos el método del gradiente. El método del gradiente descendente se basa en buscar la dirección en la que una pequeña variación del vector de pesos hace que el error decrezca más rápidamente. Para encontrar esta dirección usamos la que nos marca el gradiente de la función de error con respecto al vector de pesos.

# Backpropagation



## ¿QUÉ ES BACKPROPAGATION?

La propagación hacia atrás o retro propagación (del inglés backpropagation) es un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado utilizados para entrenar redes neuronales artificiales.

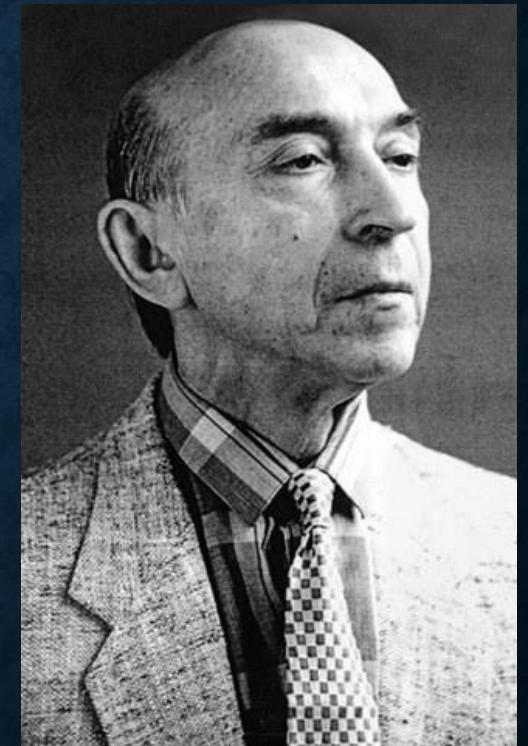
El Backpropagation es uno de los algoritmos de aprendizaje más estudiados y utilizados en las redes neuronales artificiales, pero por su misma naturaleza presenta notables deficiencias, como por ejemplo la lentitud en su proceso de aprendizaje.

# 4. LÓGICA DIFUSA

## HISTORIA

En la década de 1920, el lógico polaco Jan Lukasiewicz trabajó los principios de la lógica multivaluada, en éstos las proposiciones pueden tomar valores verdaderos fraccionales entre los unos (1) y los ceros (0) de la lógica clásica. En 1965 Lotfi A. Zadeh, en aquel entonces director del Departamento de Ingeniería Eléctrica de la Universidad de California en Berkeley, publicó *Fuzzy Sets*. Este artículo describe las matemáticas de los conjuntos difusos y por extensión de la lógica difusa, y este trabajo le dio nombre a su campo. Zadeh aplicó la lógica de Lukasiewicz a cada objeto en un conjunto y creó un álgebra completa para conjuntos difusos. Esta teoría propone funciones de pertenencia (o los valores falso y verdadero) sobre el rango [0.0, 1.0].

La lógica difusa se aplicó a mediados de la década de 1970 por Ebrahim H. Mamdani en el Queen Mary Collage en Londres. Mamdani diseñó un controlador difuso para un motor a vapor. Desde entonces el término lógica difusa es sinónimo de cualquier sistema matemático o computacional que razona con lógica difusa. La noción de sistemas difusos consiste en que los valores verdaderos (en lógica difusa) o valores de pertenencia (en conjuntos difusos) se indican en un número entre [0.0, 1.0], donde 0.0 representa falsedad total y 1.0 significa verdad absoluta.



Lotfi A. Zadeh

# CONCEPTO DE LÓGICA DIFUSA

La forma en que la gente piensa es, inherentemente, difusa. La forma en que percibimos el mundo está cambiando continuamente y no siempre se puede definir en términos de sentencias verdaderas o falsas. Consideremos como ejemplo el conjunto de vasos del mundo, que pueden estar vacíos o llenos de agua. Ahora tomemos un vaso vacío y comencemos a echar agua poco a poco, ¿en qué momento decidimos que el vaso pasa de estar vacío a estar lleno?

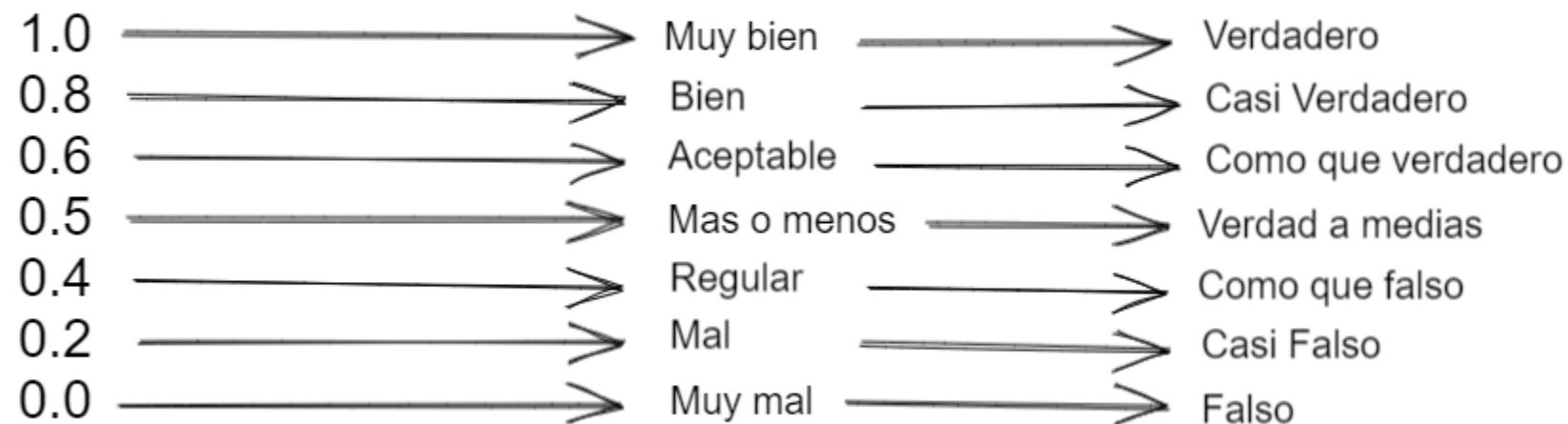


Entonces teniendo en cuenta el ejemplo de los vasos podemos decir lo siguiente. Un conjunto difuso permite a sus elementos tener un grado de pertenencia. Si el valor 1 se asigna a los elementos que están completamente en el conjunto, y 0 a los que están completamente fuera, entonces los objetos que están parcialmente en el conjunto tendrán un valor de pertenencia estrictamente entre 0 y 1. Por tanto, si un vaso completamente lleno tiene un grado de pertenencia a los vasos llenos de valor 1, y un vaso completamente vacío un grado de pertenencia a los vasos llenos de valor 0, entonces al añadir una gota a este último, su grado de pertenencia a los vasos llenos sería ligeramente superior a 0.



# EJEMPLO REALIZADO EN CLASE

## CONCEPTUALIZACIÓN DIFUSA:

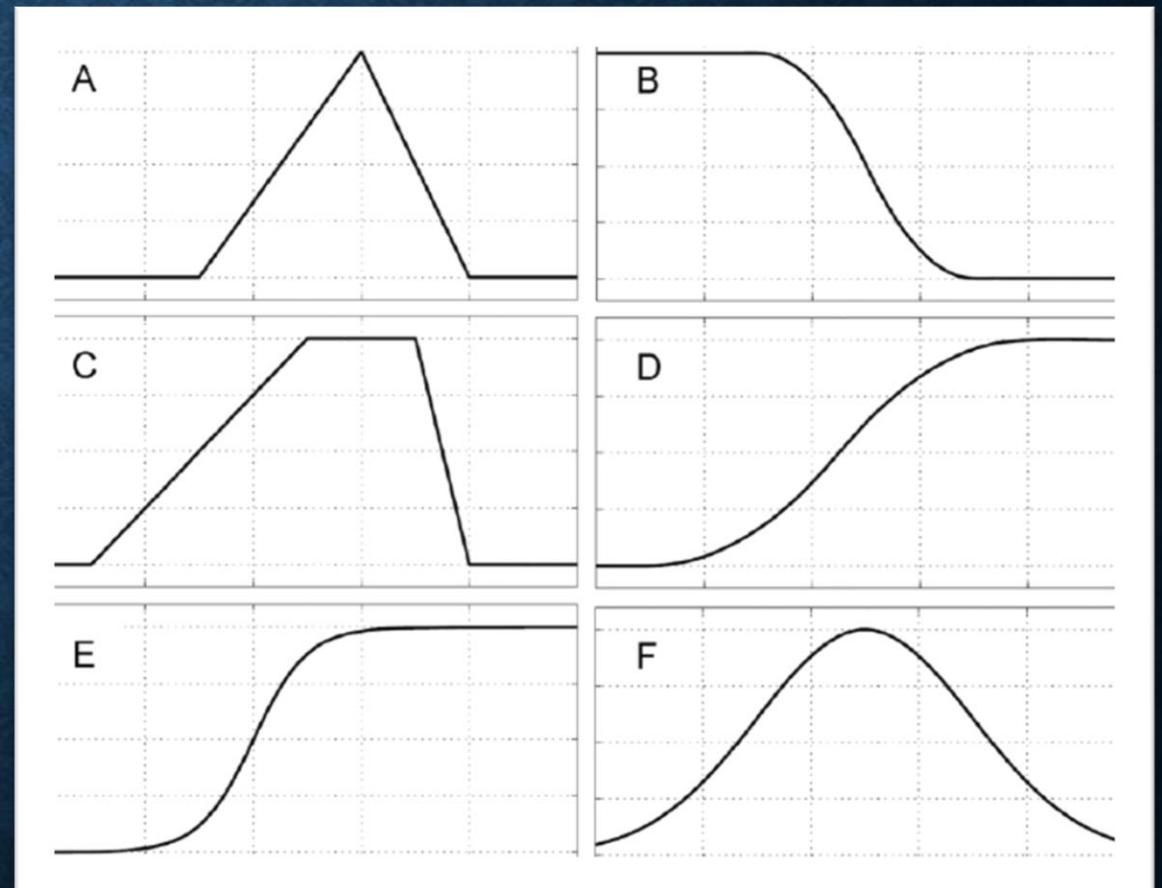


Tópico: Que tan bien se siente una persona

# FUNCION DE PERTENENCIA

Cómo ha de ser la función de pertenencia?

Eso dependerá de las características propias del conjunto real que se quiere representar, pero suelen usarse algunas funciones clásicas comunes como las que se muestran a continuación:



# ALGUNAS APLICACIONES DE LA LÓGICA DIFUSA

La lógica difusa se utiliza cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos, para procesos altamente no lineales y cuando se envuelven definiciones y conocimiento no estrictamente definido (impreciso o subjetivo). A continuación se citan algunos ejemplos de su aplicación:

- Sistemas de control de acondicionadores de aire.
- Sistemas de foco automático en cámaras fotográficas .
- Control y optimización de procesos y sistemas industriales.
- Vehículos y conducción autónoma

Como podemos ver la lógica difusa se puede aplicar en múltiples áreas, aunque también es importante hablar de cuando no aplicar o de cuando no usar lógica difusa y eso es cuando algún modelo matemático ya soluciona eficientemente el problema, cuando los problemas son lineales o cuando no tienen solución.

## 5. RESUMEN DE LO REALIZADO EN COLAB

Bueno en esta parte primero voy a hablar de lo que hice , entonces primero revise cada uno de los archivos de colab que habían sido enviados durante el semestre, especialmente revise los de lógica difusa que son los que mas me interesaba y comencé a probarlos , probe cada uno donde pude ver como se hacia para construir cada una de las graficas que se mostraban y como se hacia para representar los datos que se querían. También dentro de las cosas que hice al ver cada uno de los programas fue buscar las funciones que se estaban usando , funciones que provenían de las librerías (numpy, skfuzzy , matplotlib) , estas librerías son bastante esenciales para el uso de funciones que nos permiten hacer cálculos matemáticos y graficar también. Inicial mente eso fue lo que hice, también me fije en como se construían las funciones de membresía trapezoidal y triangular, entre otras cosas. Y bueno para finalizar modifique algunos de los códigos e intente aplicar la lógica de uno de los códigos para resolver un problema que me encontré en internet , sobre el piloto automático de un avión. A continuación se mostraran algunos ejemplos gráficos de los archivos que analice y que modifique .

## Código del Triangulo (Membresía Triangular)

```
# Función de Membresía Triangular

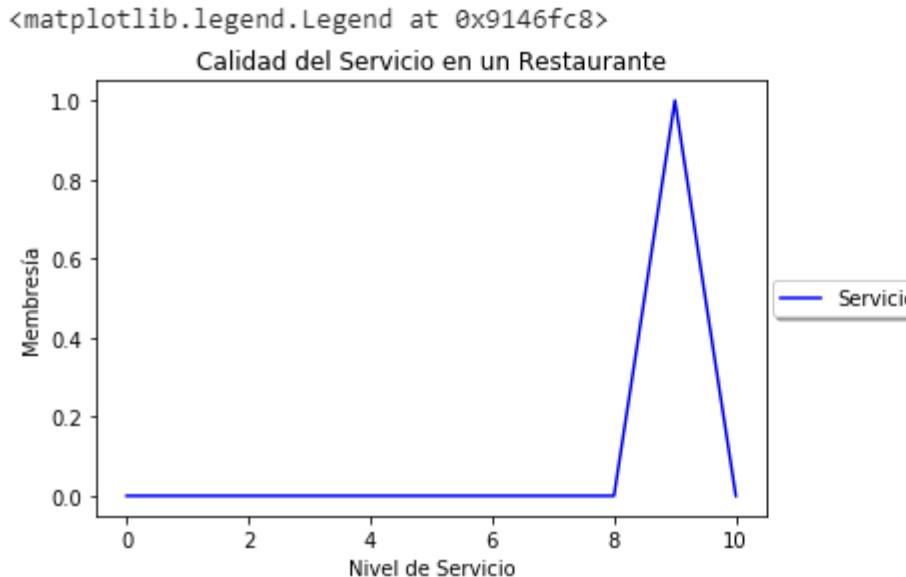
import numpy as np
import skfuzzy as sk
import matplotlib.pyplot as plt

# Se define un array x para el manejo del factor de calidad en un restaurante
x = np.arange(0, 11, 1)

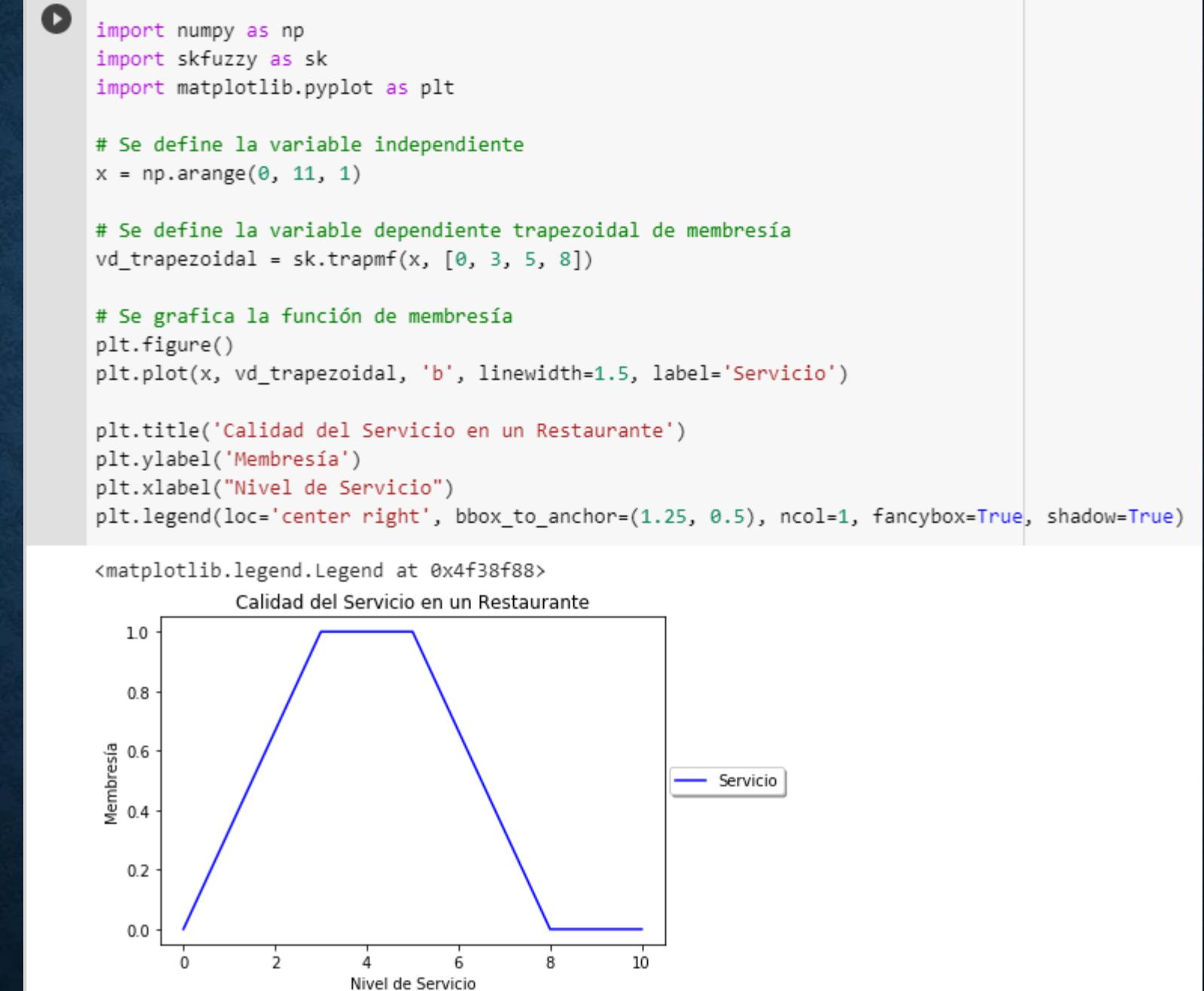
# Se define un array para la función miembro de tipo triangular
calidad = sk.trimf(x, [9, 9, 10])

# Se grafica la función de membresía
plt.figure()
plt.plot(x, calidad, 'b', linewidth=1.5, label='Servicio')

plt.title('Calidad del Servicio en un Restaurante')
plt.ylabel('Membresía')
plt.xlabel("Nivel de Servicio")
plt.legend(loc='center right', bbox_to_anchor=(1.25, 0.5), ncol=1, fancybox=True, shadow=True)
```



## Código del trapezoide (Membresía trapezoidal)



## Figura Campana de Gauss (Membresía gaussiana Bell)

```
# Función de membresía gaussiana BELL

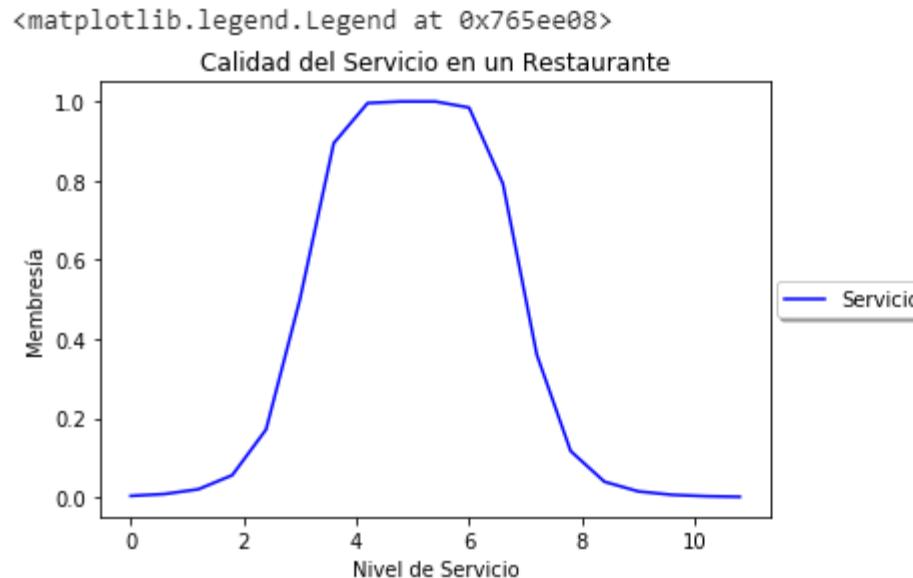
import numpy as np
import skfuzzy as sk
import matplotlib.pyplot as plt

# Se define la variable independiente
x = np.arange(0, 11, 0.6)

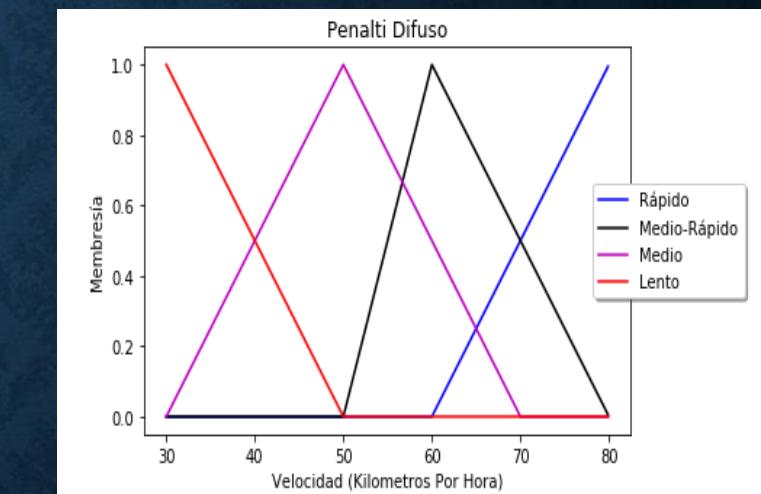
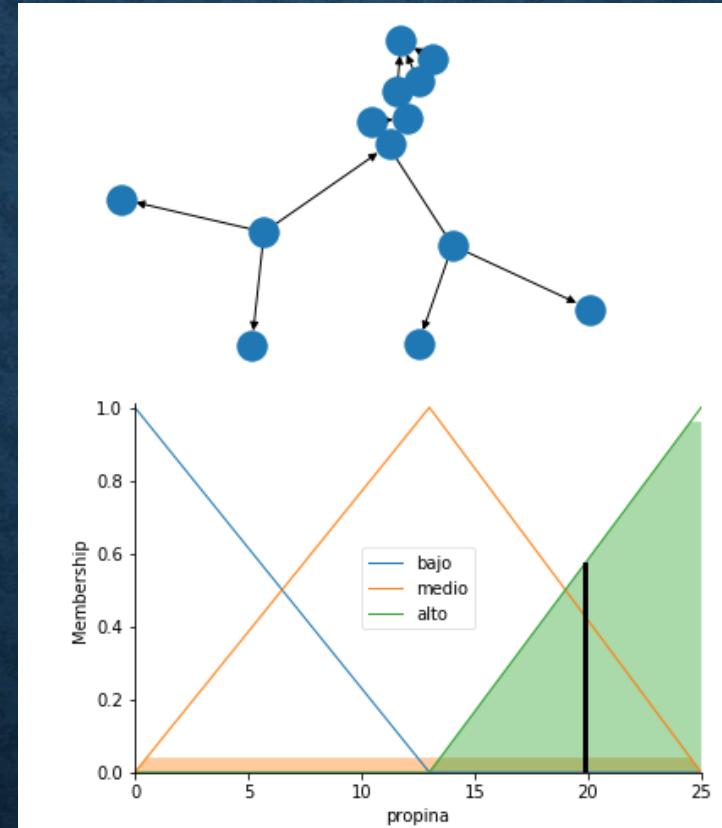
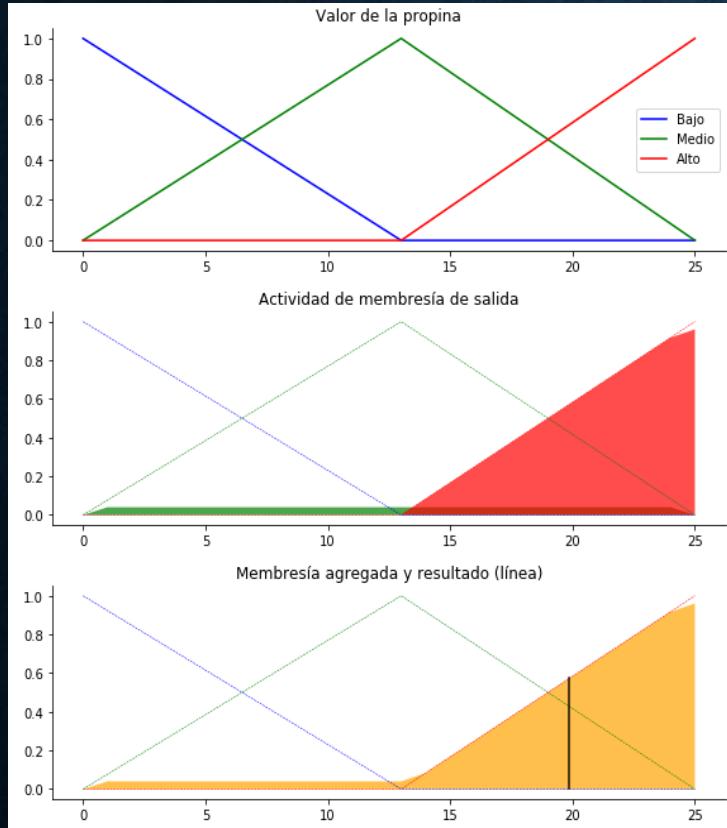
# Se define la variable dependiente gaussiana de membresía
vd_gaussiana_bell = sk.gbellmf(x, 2, 3, 5)

# Se grafica la función de membresía
plt.figure()
plt.plot(x, vd_gaussiana_bell, 'b', linewidth=1.5, label='Servicio')

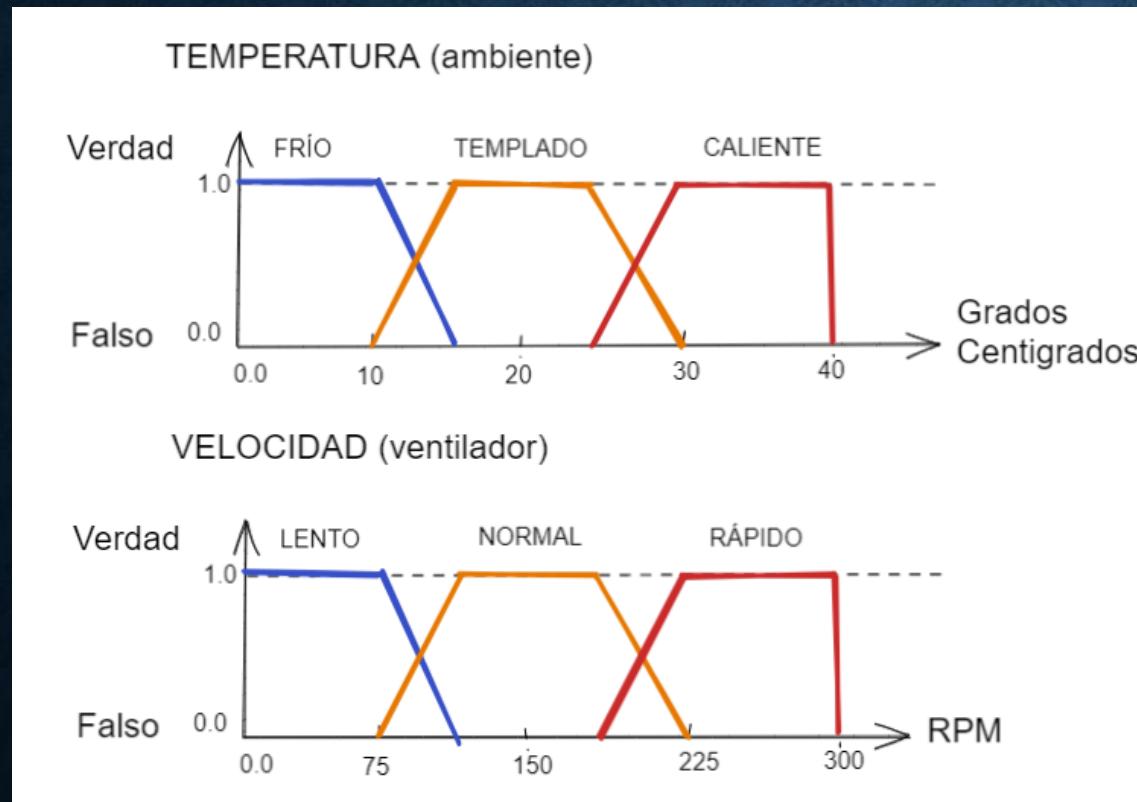
plt.title('Calidad del Servicio en un Restaurante')
plt.ylabel('Membresía')
plt.xlabel("Nivel de Servicio")
plt.legend(loc='center right', bbox_to_anchor=(1.25, 0.5), ncol=1, fancybox=True, shadow=True)
```



# IMÁGENES DE OTROS CÓDIGOS ESTUDIADOS

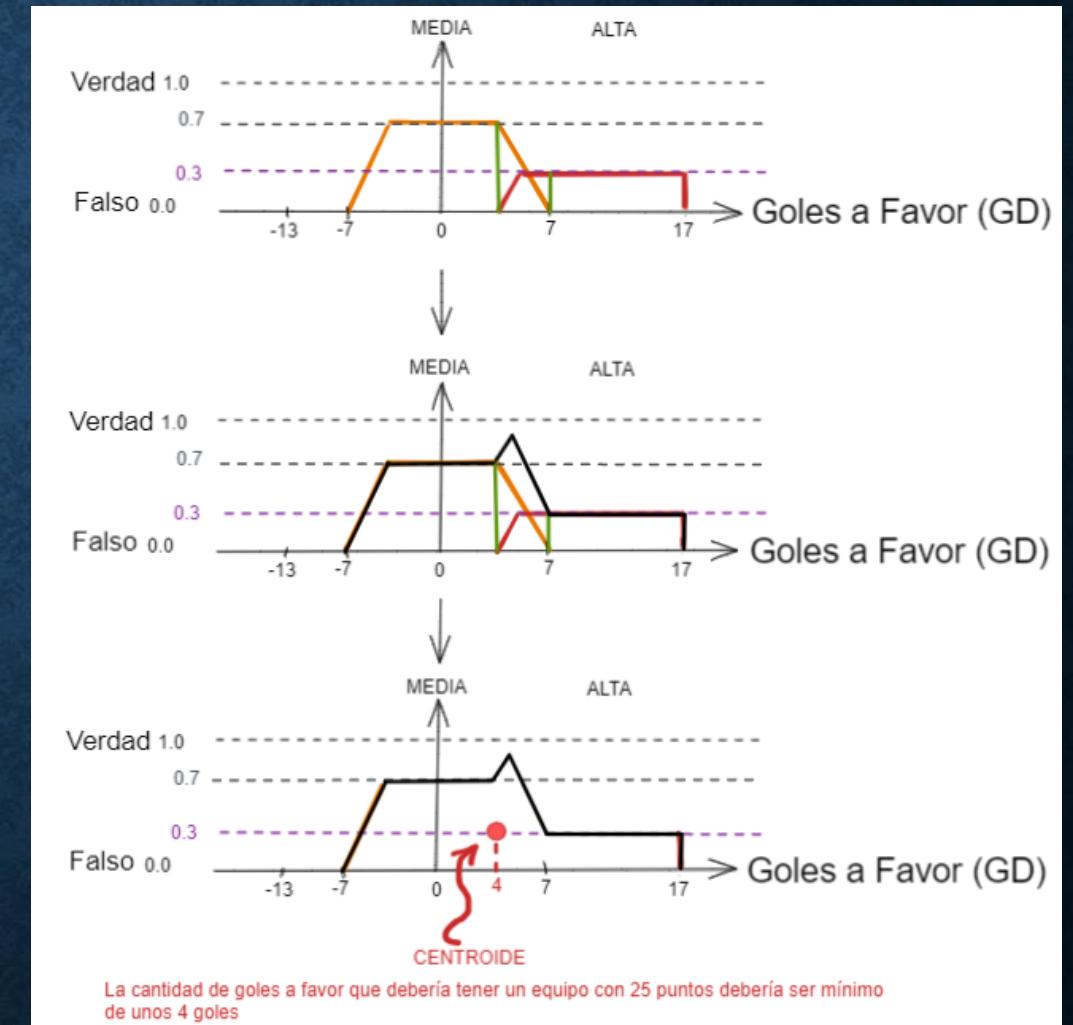
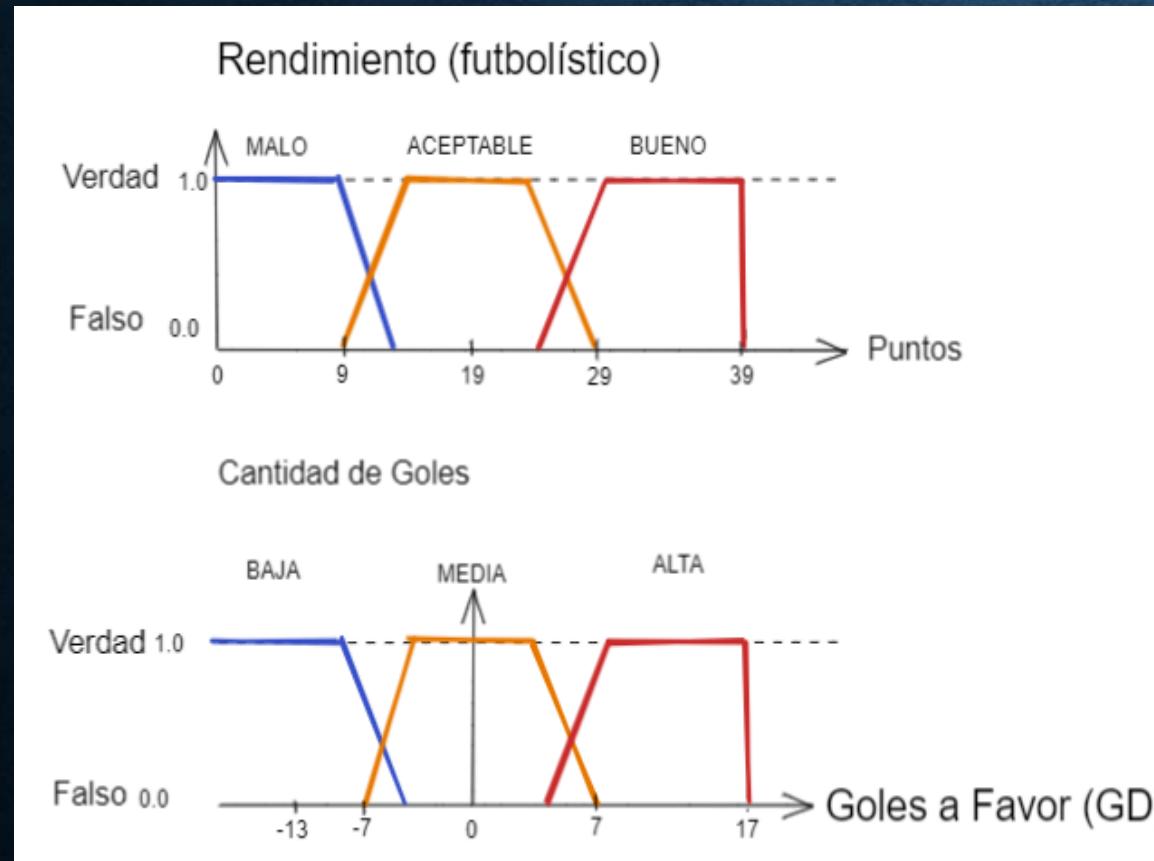


Bueno después de haber analizado lo anterior intente realizar un ejemplo con algo de lo visto en clase, para esto analice un documento de excalidraw y de la misma forma intente hacer un ejercicio aplicando código de los antes vistos y aplicando un poco de la lógica del documento de excalidraw. A continuación imágenes de lo mencionado:



Este es uno de los ejemplos vistos en clase explicados de una manera muy sencilla (nada de cogido). Entonces basado en este ejemplo yo trate de aplicar algo igual pero con un tema referente al futbol. Al final creo que me faltó mas información respecto al tema pero se consiguió algo interesante que veremos en la siguiente diapositiva.  
Los documentos de excalidraw usados y creados estarán en el GitHub.

Entonces primero hice un especie de análisis en excalidraw de la misma forma en que fue explicado en clase y luego trate de llevarlo a código .



Este seria mas o menos el código y las graficas. Muy parecido a lo realizado con lo de la calidad , servicio y propina, solo que para este caso yo use membresía trapezoidal.

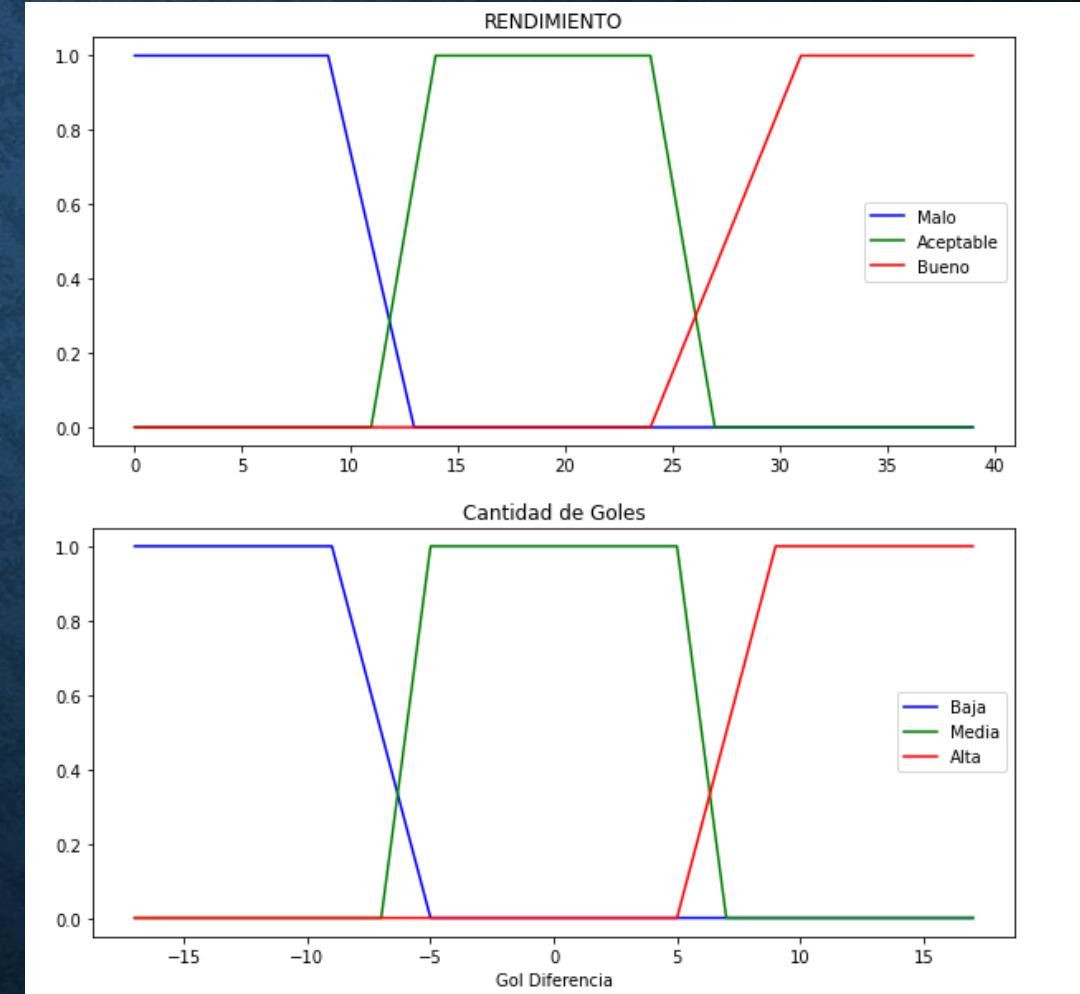
```
# Importar librerías
import numpy as np
import skfuzzy as sk
import matplotlib.pyplot as plt

# Generar variables del universo
# * Rendimiento sujeto a el rango de [0, 10]
# * Los Goles tienen un rango de [-17,17] en unidades de puntos porcentuales
x_rendimiento = np.arange(0, 40, 1)
x_cantidad_goles = np.arange(-17, 18, 1)

# Definiendo las funciones miembro trapezoidales
rendimiento_malo = sk.trapmf(x_rendimiento, [0, 0, 9, 13])
rendimiento_aceptable = sk.trapmf(x_rendimiento, [11, 14, 24, 27])
rendimiento_bueno = sk.trapmf(x_rendimiento, [24, 31, 39, 39])
cantidad_goles_bajo = sk.trapmf(x_cantidad_goles, [-17,-17,-9,-5])
cantidad_goles_medio = sk.trapmf(x_cantidad_goles, [-7,-5,5,7])
cantidad_goles_alto = sk.trapmf(x_cantidad_goles, [5,9,17,17])

# Visualizar estos universos y funciones de pertenencia.
fig, (ax0, ax1) = plt.subplots(nrows=2, figsize=(10, 10))
ax0.plot(x_rendimiento, rendimiento_malo, 'b', linewidth=1.5, label='Malo')
ax0.plot(x_rendimiento, rendimiento_aceptable, 'g', linewidth=1.5, label='Aceptable')
ax0.plot(x_rendimiento, rendimiento_bueno, 'r', linewidth=1.5, label='Bueno')
ax0.set_title('RENDIMIENTO')
plt.xlabel("Puntos")
ax0.legend()

ax1.plot(x_cantidad_goles, cantidad_goles_bajo, 'b', linewidth=1.5, label='Baja')
ax1.plot(x_cantidad_goles, cantidad_goles_medio, 'g', linewidth=1.5, label='Media')
ax1.plot(x_cantidad_goles, cantidad_goles_alto, 'r', linewidth=1.5, label='Alta')
ax1.set_title('Cantidad de Goles')
plt.xlabel("Gol Diferencia")
ax1.legend()
```



# PARA TERMINAR

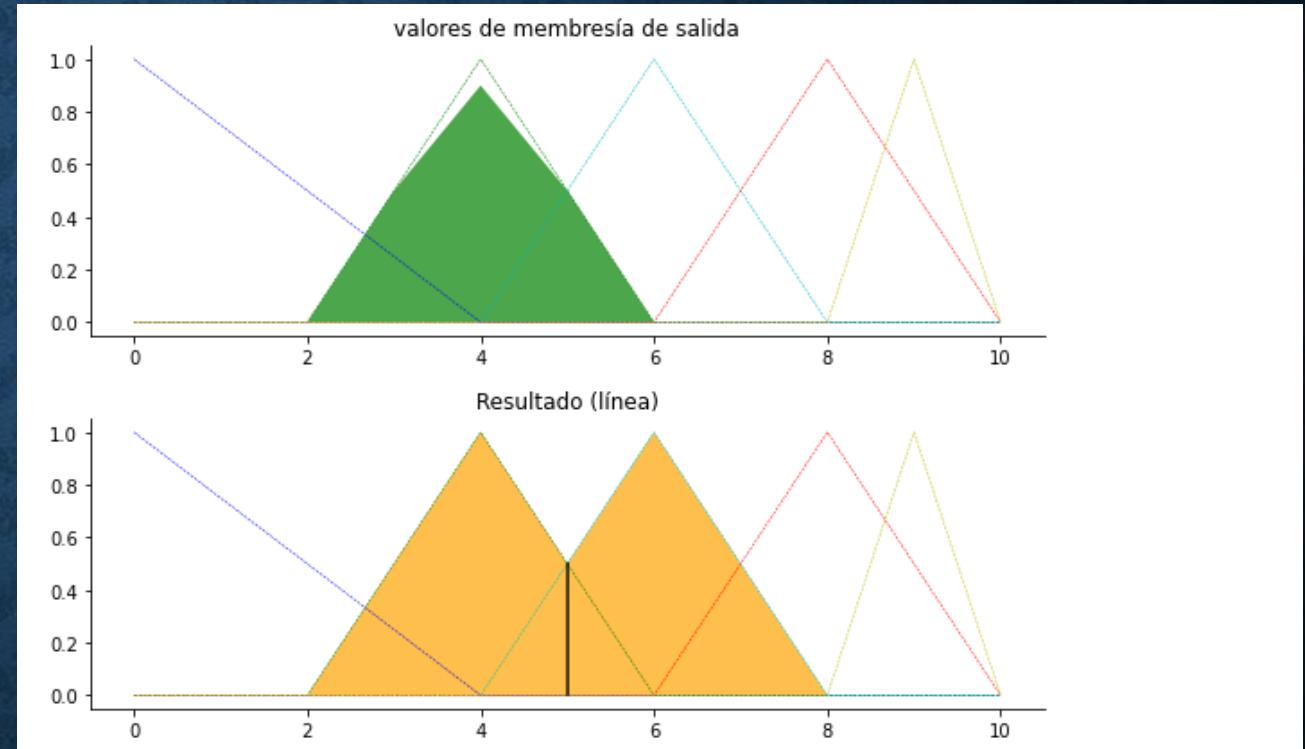
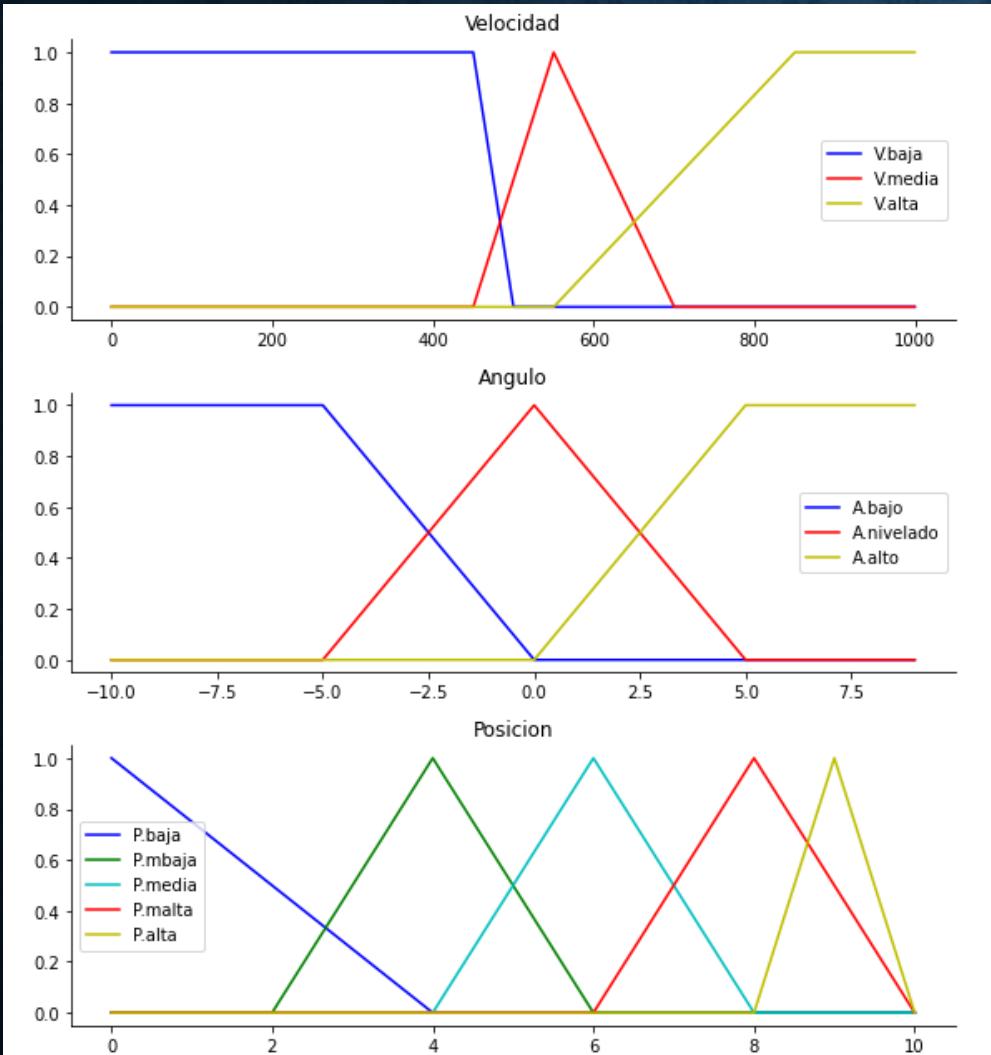
Y bueno ya para terminar intente aplicar la lógica vista en clase y los códigos vistos y estudiados para intentar resolver la mejor manera un problema el cual es el siguiente.

**Problema:** un avión esta sujeto a turbulencias, las que causan que el avión baje o suba bruscamente formando un Angulo respecto de su línea de vuelo. se quiere diseñar un sistema de control difuso para que un piloto automático responda al problema de turbulencia ajustando la posición del timón de la aeronave.

Se pide calcular la posición del timón si el avión vuela a 515kmph y por la turbulencia se formo un Angulo de -2.5 grados respecto a su línea de vuelo.

Nota: el problema fue sacado de un video en YouTube el cual mostraba la lógica del problema y donde se daba información importante para un mejor entendimiento, eso si , aclarar que en el video no se muestra código en Python o algo por el estilo. En el github se colocara un archivo de texto donde estará el link del video por si se quiere ver .

Este seria el resultado del problema mencionado anteriormente.



```
# BUENO RECORDEMOS QUE ERA LO QUE ESTABAMOS BUSCANDO
# LA PREGUNTA ERA:
# Se pide calcular la posicion del timon si el avion vuela a 515 Kilometros por hora
# y por la turbulencia se formo un angulo de -2.5 grados respecto a su linea de vuelo.

# entonces en la ultima grafica podemos ver el resultado , en este caso la grafica nos dice
# que la posicion del timon del avion debe estar a ( 5cm ).
```

## 6. WEBGRAFÍA

<https://artyco.com/que-son-las-redes-neuronales-y-cual-es-su-aplicacion-en-el-marketing/>

[https://es.wikipedia.org/wiki/Red\\_neuronal\\_artificial](https://es.wikipedia.org/wiki/Red_neuronal_artificial)

<https://www.datacentric.es/blog/insight/red-neuronal-artificial-aplicaciones/>

<http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm>

<https://sites.google.com/site/logicadifusaingindustrialpaita/logica-difusa/historia-de-la-logica-difusa>

<http://www.cs.us.es/~fsancho/?e=97>

[https://es.wikipedia.org/wiki/L%C3%B3gica\\_difusa#Ventajas\\_e\\_inconvenientes](https://es.wikipedia.org/wiki/L%C3%B3gica_difusa#Ventajas_e_inconvenientes)

[https://colab.research.google.com/drive/1UXjOjfBLMkyPWiMn\\_ZsZk\\_c6XGxT1UCq](https://colab.research.google.com/drive/1UXjOjfBLMkyPWiMn_ZsZk_c6XGxT1UCq)

<https://colab.research.google.com/drive/1LiTUtyarsKHRPcb4JOLmvkM1naDfQJtW>

Y BUENO ESO SERIA  
TODO, MUCHAS GRACIAS  
POR LLEGAR HASTA ACA

