Objetivo:

- Consolidar los conocimientos adquiridos en clase de los sistemas expertos.

Enunciado:

Se desea generar un sistema de recomendación de películas, por tal motivo se va a utiliza una base de datos orientada a grafos y un control de lógica difusa para clasificar el riesgo financiero, el mismo que sera ingresado como atributo del cliente en el sistema re comendador, para lograr esto se describe los pasos a seguir:

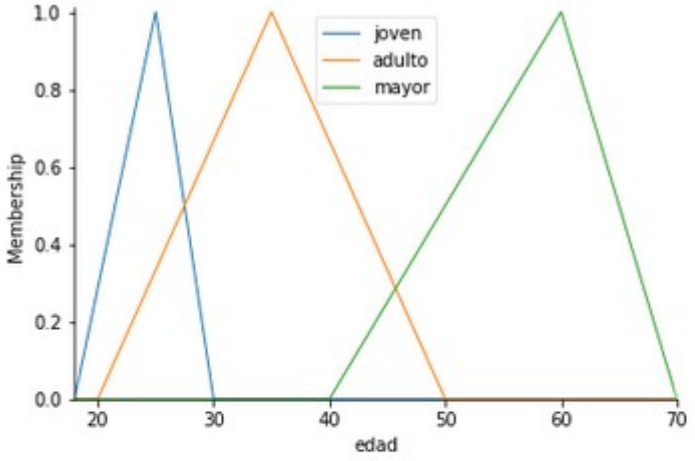
Caja de herramientas de lógica difusa para Python.
Este paquete implementa muchas herramientas y funciones útiles para computación y proyectos que involucran lógica difusa, también conocida como lógica gris.

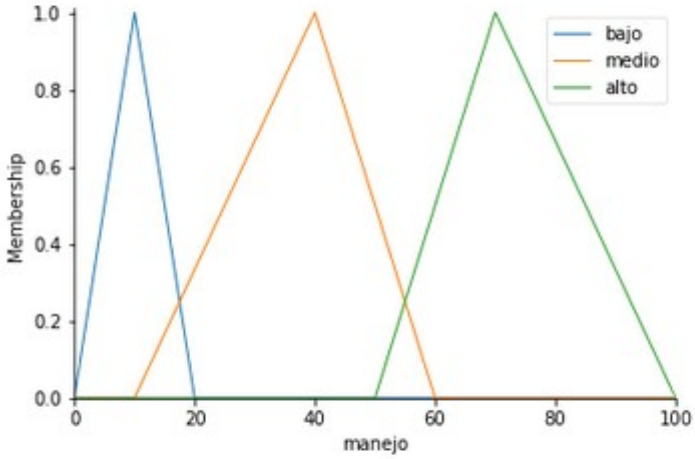
1) E**valuar el riesgo financiero** de sus clientes que requieren la recomendación de películas. Para evaluar el riesgo financiero se toma en cuenta **la edad** del asegurado y su **porcentaje de manejo** durante el año. Para ello se tiene las siguientes reglas y la función de pertinencia. El proceso seguir se describe en el siguiente link: https://medium.com/@javierdiazarca/l%C3%B3gica-difusa-ejercicios-propuestos-b99603ef1bc0.

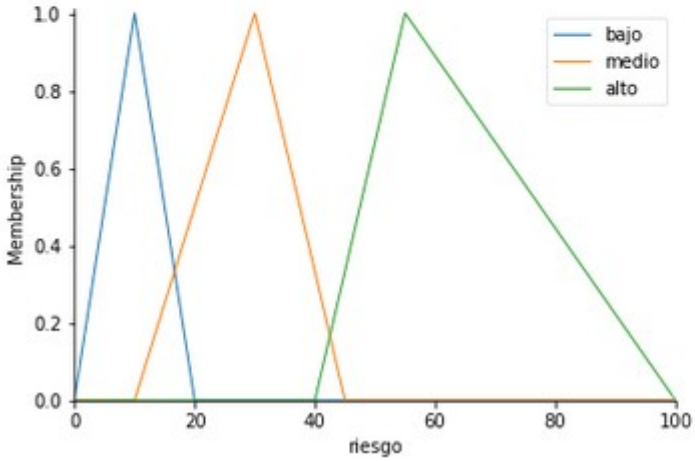
Implementación de las reglas

```
riesgo = ctrl.Consequent(np.arange(0, 101, 1), 'riesgo')
manejo = ctrl.Antecedent(np.arange(0, 101, 1), 'manejo')
edad = ctrl.Antecedent(np.arange(18, 71, 1), 'edad')

riesgo['bajo'] = fuzz.trimf(riesgo.universe, [0, 10, 20])
riesgo['medio'] = fuzz.trimf(riesgo.universe, [10, 30, 45])
riesgo['alto'] = fuzz.trimf(riesgo.universe, [40, 55, 100])

manejo['bajo'] = fuzz.trimf(manejo.universe, [0, 10, 20])
manejo['medio'] = fuzz.trimf(manejo.universe, [10, 40, 60])
manejo['alto'] = fuzz.trimf(manejo.universe, [50, 70, 100])

edad['joven'] = fuzz.trimf(edad.universe, [18, 25, 30])
edad['adulto'] = fuzz.trimf(edad.universe, [20, 35, 50])
edad['mayor'] = fuzz.trimf(edad.universe, [40, 60, 70])


regla1 = ctrl.Rule(manejo['bajo'] and edad['joven'], riesgo['medio'])
regla2 = ctrl.Rule(manejo['medio'] and edad['joven'], riesgo['alto'])
regla3 = ctrl.Rule(manejo['alto'] and edad['joven'], riesgo['alto'])

regla4 = ctrl.Rule(manejo['bajo'] and edad['adulto'], riesgo['bajo'])
regla5 = ctrl.Rule(manejo['medio'] and edad['adulto'], riesgo['medio'])
regla6 = ctrl.Rule(manejo['alto'] and edad['adulto'], riesgo['alto'])

regla7 = ctrl.Rule(manejo['bajo'] and edad['mayor'], riesgo['medio'])
regla8 = ctrl.Rule(manejo['medio'] and edad['mayor'], riesgo['alto'])
regla9 = ctrl.Rule(manejo['alto'] and edad['mayor'], riesgo['alto'])

regla10 = ctrl.Rule(edad['joven'] and manejo['bajo'], riesgo['medio'])
regla11 = ctrl.Rule(edad['joven'] and manejo['medio'], riesgo['alto'])
regla12 = ctrl.Rule(edad['joven'] and manejo['alto'], riesgo['alto'])

regla13 = ctrl.Rule(edad['adulto'] and manejo['bajo'], riesgo['bajo'])
regla14 = ctrl.Rule(edad['adulto'] and manejo['medio'], riesgo['medio'])
regla15 = ctrl.Rule(edad['adulto'] and manejo['alto'], riesgo['alto'])

regla16 = ctrl.Rule(edad['mayor'] and manejo['bajo'], riesgo['medio'])
regla17 = ctrl.Rule(edad['mayor'] and manejo['medio'], riesgo['alto'])
regla18 = ctrl.Rule(edad['mayor'] and manejo['alto'], riesgo['alto'])

riesgo_ctrl = ctrl.ControlSystem([regla1,regla2,regla3,regla4,regla5,regla6,regla7,regla8,regla9,regla10,regla11,re
riesgos = ctrl.ControlSystemSimulation(riesgo_ctrl)
```

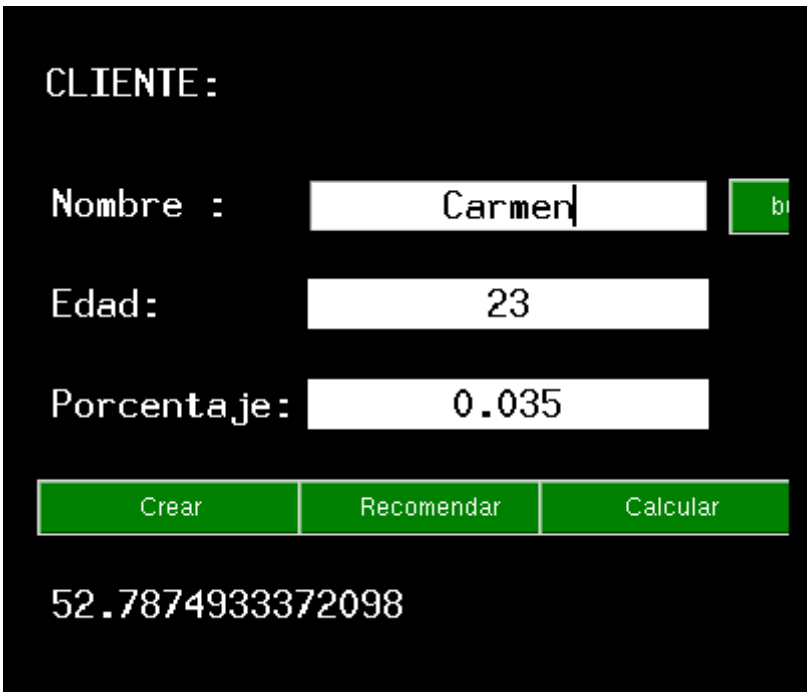
Graficas







- evaluación del riesgo financiero



2) Generar números aleatorios para la edad y el porcentaje de manejo con el objetivo de generar al menos 100 personas y ademas incluir el listado de películas vistas y el valor del rating de cada película. Al menos 50 películas y un total de nodos de al menos 350 nodos.

```
def crear():
    porcen1=(random.randrange(10, 100))/100
    edadn1=random.randrange(10, 80)
    neo4j = Neo4jService('bolt://localhost:7687', 'neo4j', 'pruebadata')
    with neo4j._driver.session() as session:
        session.write_transaction(neo4j.crear_usuario,usuario1.get(),edadn1,porcen1)
```

3) Con estos datos aplicar el algoritmo de KNN y Similitud de Coseno para la recomendación de películas, seguir el siguiente tutorial: https://neo4j.com/graphgist/movie-recommendations-with-k-nearest-neighbors-and-cosine-similarity.

```
def recomendaciones(self,tx,reco,vjuego):
    result = tx.run("MATCH    (b:Persona)-[r:RATED]->(m:Pelicula), (b)-[s:SIMILARITY]-(a:Persona {name:$recomend
                    "WHERE    NOT((a)-[:RATED]->(m))\n"
                    "WITH     m, s.similarity AS similarity, r.rating AS rating\n"
                    "ORDER BY m.name, similarity DESC\n"
                    "WITH     m.name AS pelicula, COLLECT(rating)[0..3] AS ratings\n"
                    "WITH     pelicula, REDUCE(s = 0, i IN ratings | s + i)*1.0/20 AS reco\n"
                    "ORDER BY reco DESC\n"
                    "RETURN   pelicula AS Pelicula, reco AS Recommendation",recomendara=reco)
```


CLIENTE:
Nombre :
Abel
Edad:
0
Porcentaje:
0
Crear
Recomendar
Calcular
Grafico Edad
Grafico Manejo
Grafico Riesgo
TE RECOMENDAMOS :
Witness'
Out of Africa
Honey I Shrunk the Kids
Little Mermaid'
Ninja Turtles
Ferris Bueller'
Dances With Wolves'
Driving Miss Daisy'