In [1]:

```
from neo4j import GraphDatabase
from tkinter import messagebox, ttk
from tkinter import *
import tkinter
import matplotlib.pyplot as plt
import numpy as np
import csv
import random
# https://medium.com/@javierdiazarca/l%C3%B3gica-difusa-ejercicios-propuestos-b99603ef1
```

In [2]:

```
#SE GENERA LA TABLA

tabla = (["","Joven","Adulto","Mayor"],["Bajo","Medio","Bajo","Medio"],["Medio","Alto",
for t in tabla:
    print("|",t[0],"|",t[1],"|",t[2],"|",t[3])
```

```
|  | Joven | Adulto | Mayor
| Bajo | Medio | Bajo | Medio
| Medio | Alto | Medio | Alto
| Alto | Alto | Alto | Alto
```

In [3]:

```
#CLASE PAR CREAR NODO CENTRAR-PARQUE CENTRAL
class CLASE_NEO4J(object):
    def __init__(self):
        self._driver = GraphDatabase.driver("bolt:neo4j://localhost:7687", auth=("neo4
    def close(self):
        self._driver.close()
    def LISTAR(self):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._LISTAR_PERSONAS)

    def KNN(self):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._EJECUTAR_KNN)
    def VER_PELICULA(self,persona,pelicula):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VER,persona,pelicula)

    def VISTAS(self,persona):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._LISTAR_VISTAS,persona)

    def CREAR_PELICULA(self, message, nombre, rating):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_PELICULA, message, nomb
            print(greeting)
    def CREAR_PERSONA(self, message, nombre, edad, por_manejo, pel_vista, pel_rating):
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_PERSONA, message, nombr
            print(greeting)

    #METODO PARA CREAR LOS NODOS DE LUGARES
    @staticmethod
    def _VALIDAR_PELICULA(tx, message, nombre, rating):
        #SE BUSCA SI EL LUGAR DEL ARREGLO EXISTE EN LA BASE NEO4J
        result2 = tx.run("match(l:Pelicula {nombre:'"+nombre+"'}) return l.nombre").da
        #CONDICION PARA VERIFICAR SI EXISTE
        if int(len(result2)) == 0:
            print("SE CREA LA PELICULA EN LA BASE.....")
            #SE CREA NODO LUGAR
            result = tx.run("CREATE("+nombre+":Pelicula {nombre:'"+nombre+"' ,rating:
                        "SET "+nombre+".message = $message "
                        "RETURN "+nombre+".message + ', from node ' + id("+nombre+")",
        elif int(len(result2))  == 1:
            print("EL NODO PELICULA YA EXISTE, INGRESAR OTRA PELICULA.....")
    @staticmethod
    def _VALIDAR_PERSONA(tx, message,nombre, edad, por_manejo, pel_vista, pel_rating):
        #SE BUSCA SI EL LUGAR DEL ARREGLO EXISTE EN LA BASE NEO4J
        result2 = tx.run("match(l:Personas {nombre:'"+nombre+"'}) return l.nombre").da
        #CONDICION PARA VERIFICAR SI EXISTE
        if int(len(result2)) == 0:
            print("SE CREA LA PERSONA EN LA BASE.....")
            #SE CREA NODO LUGAR
            result = tx.run("CREATE("+nombre+":Personas {nombre:'"+nombre+"' ,edad: "+
                        "SET "+nombre+".message = $message "
                        "RETURN "+nombre+".message + ', from node ' + id("+nombre+")",
        elif int(len(result2))  == 1:
            print("EL NODO PERSONA YA EXISTE, INGRESAR OTRO PERSONA.....")
    @staticmethod
    def _VER(tx,persona,pelicula):
```

```
        result2 = tx.run("match("+str(persona)+":Personas {nombre:'"+str(persona)+"'})
        print("valor resultados es  >",result2)
        print("SE GENERA LA RELACION VER PELICULA",persona,pelicula)


    @staticmethod
    def _LISTAR_PERSONAS(tx):
        result = tx.run("match (p:Personas) return p.nombre as nombre,p.edad as edad,p
        lista =[]
        for io in result:
            var = str(io.get("nombre"))+"   :   "+str(io.get("edad"))+"   :   "+str(io
            lista.append(var)
        combo["values"]=lista
    @staticmethod
    def _EJECUTAR_KNN(tx):
        #SE GENERA EL GRAFO CON LOS NODOS PARA EL ALGORITMO DE KNN
        result1 = tx.run("CALL gds.graph.create('graficoknn12',{Pelicula: {label: 'Pel

        #SE EJECUTA ALGORITMO KNN
        result = tx.run("CALL gds.beta.knn.stream('graficoknn12', {    topK: 1,    nod
        lista =[]
        for io in result:
            var = str(io.get("Pelicula1"))+" : "+str(io.get("Pelicula2"))+" : "+str(io
            lista.append(var)
        combo2["values"]=lista
    @staticmethod
    def _EJECUTAR_COSTO(tx):
        #SE GENERA EL GRAFO CON LOS NODOS PARA EL ALGORITMO DE KNN
        print("SE EJECUTA EL COSTO")
#         result1 = tx.run("CALL gds.graph.create('graficoknn12',{Pelicula: {label: 'P

#         #SE EJECUTA ALGORITMO KNN
#         result = tx.run("CALL gds.beta.knn.stream('graficoknn12', {    topK: 1,    n
#         lista =[]
#         for io in result:
#             var = str(io.get("Pelicula1"))+" : "+str(io.get("Pelicula2"))+" : "+str(
#             lista.append(var)
#         combo2["values"]=lista
    @staticmethod
    def _LISTAR_VISTAS(tx,persona):
        result = tx.run("match(p:Personas) -[:VER_PELICULA]->(pe:Pelicula) where p.nom
        lista =[]
        print(result)
        for io in result:
            var = str(io.get("pe.nombre"))
            lista.append(var)
        combo3["values"]=lista

# MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
#SE INICIALIZA LA CLASE DE LOS METODOS DE NEO4J
grafo=CLASE_NEO4J()
```

In [4]:

```
ll= []
with open('pelis.csv', newline='') as File:
    reader = csv.reader(File)
    for row in reader:
        #print(row)
        ll.append(row)
        print(row[0])
        print(row[1])
        print(row[2])
        grafo.CREAR_PELICULA("SE GENERA UN NODO PELICULA EN LA BASE >>>>>>>>> ",str(row

def randomp():
    i = random.randint(0, 19)
    p = ll[i][1]
    r = ll[i][2]
    return p, r

def edad():
    i = random.randint(18, 70)
    return i
def manejo():
    por_manejo = random.randint(0, 100)
    return por_manejo
p = "Persona"
for i in range(1, 101):
    pp = "{}_{}".format(p, i)
    grafo.CREAR_PERSONA("SE GENERA UN NODO PERSONA EN LA BASE >>>>>>>>> ",str(pp), str(

```

```
1
Renacido
7.2
EL NODO PELICULA YA EXISTE, INGRESAR OTRA PELICULA........
None
2
Parasitos
8.0
EL NODO PELICULA YA EXISTE, INGRESAR OTRA PELICULA........
None
3
Vengadores
7.3
EL NODO PELICULA YA EXISTE, INGRESAR OTRA PELICULA........
None
4
El_caballero_oscuro
8.1
EL NODO PELICULA YA EXISTE, INGRESAR OTRA PELICULA........
```

In [ ]:

```
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


manejo=0.0
edad=0
resultados =([])
resultadosm =([])
listax1 = [18,19,20,21,22,23,24,25,26,27,28,29,30,20,23,26,29,32,35,38,41,44,47,50,40,
listay1 = [1,1,1,1,1,1,1,1,0.8,0.6,0.4,0.2,0,0,0.2,0.4,0.6,0.8,1,0.8,0.6,0.4,0.2,0,0,0
listax = [0,10,11,13,15,17,19,21,10,16,22,28,34,40,44,48,52,56,60,50,54,58,62,66,70,80
listay = [1,1,1,0.8,0.6,0.4,0.2,0,0,0.2,0.4,0.6,0.8,1,0.8,0.6,0.4,0.2,0,0,0.2,0.4,0.6,
listax2 = [0,10,12,14,16,18,20,10,14,18,22,26,30,33,36,39,42,45,40,43,46,49,52,55,60,7
listay2 = [1,1,0.8,0.6,0.4,0.2,0,0,0.2,0.4,0.6,0.8,1,0.8,0.6,0.4,0.2,0,0,0.2,0.4,0.6,0

def listar_pelis():
    st = str(combo.get()).split(" : ")
    #PERSONA
    print("VALOR DE PERSONAS>> ",str(st[0]))
    per = str(st[0])
    grafo.VISTAS(str(per).replace(" ",""))
    tkinter.messagebox.showinfo(title="SE PRESIONA", message="Se Listan las peliculas


def listarp():
    pv = grafo.LISTAR()
    tkinter.messagebox.showinfo(title="SE PRESIONA", message="Se listaron con exitos")

def ver_peli():
    st = str(combo.get()).split(" : ")
    #PERSONA
    per = str(st[0])
    st2 = str(peli.get())
    #PELICULA
    pel = st2
    grafo.VER_PELICULA(str(per).replace(" ",""),str(pel).replace(" ",""))
    tkinter.messagebox.showinfo(title="SE PRESIONA", message="Esta viendo la pelicula

def logicadif():
    st = str(combo.get()).split("   :   ")
    print("<-----SE EJECUTA LA LOGICA DIFUSA-----> ")
    ax1.axvline(int(st[1]), label='pyplot vertical line',color='red')
    line.draw()
    ax.axvline(int(st[2]), label='pyplot vertical line',color='blue')
    line.draw()
    #VARIABLE DE EDAD
    a = int(st[1])
    #VARIABLE DE MANEJO
    b = int(st[2])
    cont=0
    for lx in listax1[:13]:
        if int(lx) == int(a):
            print("*********JOVEN***************",lx,listay1[cont])
            resultados.insert(0,["Medio","Alto","Alto",listay1[cont]])
        elif int(lx) != int(a):
            pass
```

```
        cont =cont +1
    #SE BUSCA EL VALOR INGRESADO EN EL VECTOR DE X ESTA EN T MEDIA
    cont1=0
    lisyn = listay[13:24]
    for lx in listax[13:24]:
        if int(lx) == int(a):
            print("**********ADULTO********",lx,lisyn[cont1])
            resultados.insert(1,["Bajo","Medio","Alto",lisyn[cont1]])
        elif int(lx) != int(a):
            pass
        cont1 = cont1 +1
    #SE BUSCA EL VALOR INGRESADO EN EL VECTOR DE X ESTA T ALTA
    cont=0
    lista = listay[24:]
    for lx in listax[24:]:
        if int(lx) == int(a):
            print("*********MAYOR*********",lx,lista[cont])
            resultados.insert(2,["Medio","Alto","Alto",lista[cont]])
        elif int(lx) != int(a):
            pass
        cont =cont +1



    #SE VALIDA  LOS VALORES DE MANEJO
    cont2=0
    lisyn = listay[19:]
    for lx in listax[19:]:
        if int(lx) == int(b):
            print("*********PORCENTAJE ALTO*********",lx,lisyn[cont2])
            resultadosm.insert(0,["Alto","Alto","Alto",lisyn[cont2]])
        elif int(lx) != int(b):
            pass
        cont2 = cont2 +1
    #SE BUSCA EL VALOR INGRESADO EN EL VECTOR DE X ESTA T ALTA
    cont23=0
    lista = listay[8:19]
    for lx in listax[8:19]:
        if int(lx) == int(b):
            print("***********PORCENTAJE MEDIO*********",lx,lista[cont23])
            resultadosm.insert(1,["Alto","Medio","Alto",lista[cont23]])
        elif int(lx) != int(b):
            pass
        cont23 =cont23 +1
    #PORCNTAJE BAJO
    cont24=0
    lista = listay[:8]
    for lx in listax[:8]:
        if int(lx) == int(b):
            print("*********PORCENTAJE BAJO*******",lx,lista[cont24])
            resultadosm.insert(2,["Alto","Medio","Alto",lista[cont24]])
        elif int(lx) != int(b):
            pass
        cont24 =cont24 +1
    print("VALOR DE LOS RESULTADOS DE EDADES >> ",resultados)
    print("VALOR DE LOS MANEJOS>> ",resultadosm)
    ct=0
    for rs in resultados:
        if str(rs)!="[]":
            print("EDAD------->",rs)
```

```
        if str(resultadosm)!="[]":
            print("MANEJO------->",resultadosm[ct])
        ct=ct+1
    #SE REALIZA LA RECOMENDACIONES BASADAS EN LAS SIMILITUDES
    grafo.KNN()
    tkinter.messagebox.showinfo(title="SE EJECUTARA KNN ", message="Selecione una de l

def B0f():
    ax1.clear()
    ax1.plot(listax1[:13], listay1[:13],label="Joven")
    ax1.legend()
    ax1.plot(listax1[13:24], listay1[13:24],label="Adulto")
    ax1.legend()
    ax1.plot(listax1[24:], listay1[24:], label="Mayor")
    ax1.legend()
    ax1.grid(True)
    ax1.set_xlabel('$x$'),ax1.set_ylabel('$y$')
    ax1.set_title('EDADES')
    line.draw()


    ax.clear()
    ax.plot(listax[:8], listay[:8],label="Bajo")
    ax.legend()
    ax.plot(listax[8:19], listay[8:19],label="Medio")
    ax.legend()
    ax.plot(listax[19:], listay[19:], label="Alto")
    ax.legend()
#     ax.fill_between(listax[:21],listay[:21],color="green")
    ax.grid(True)
    ax.set_xlabel('$x$'),ax.set_ylabel('$y$')
    ax.set_title('% MANEJO')
    line.draw()

    ax2.clear()
    ax2.plot(listax2[:7], listay2[:7],label="Bajo")
    ax2.legend()
    ax2.plot(listax2[7:18], listay2[7:18],label="Medio")
    ax2.legend()
    ax2.plot(listax2[18:], listay2[18:], label="Alto")
    ax2.legend()
#     ax.fill_between(listax[:21],listay[:21],color="green")
    ax2.grid(True)
    ax2.set_xlabel('$x$'),ax2.set_ylabel('$y$')
    ax2.set_title('% RIESGO FINANCIERO')
    line.draw()

#--- Raiz ---
root = tk.Tk()
root.geometry('940x450')
root.title("Tkinter + Matplotlib")
#------------

#-- Frames ---
left_frame = tk.Frame(root)
left_frame.place(relx=0.03, rely=0.05, relwidth=0.25, relheight=0.9)

right_frame = tk.Frame(root, bg='#C0C0C0', bd=1.5)
right_frame.place(relx=0.3, rely=0.05, relwidth=0.65, relheight=0.9)
#---------------
#LABEL--------------
```

```
182
183
184 label = Label(left_frame, text="LISTADO DE PERSONAS", relief=RAISED )
185 label.place(relx=0.05, rely=0.35,relheight=0.03, relwidth=1)
186
187 label2 = Label(left_frame, text="Pelicula1 : Pelicula2 : Rating", relief=RAISED )
188 label2.place(relx=0.05, rely=0.52,relheight=0.03, relwidth=1)
189 #-------------
190 #--- Botones ---
191
192 B0 = tk.Button(left_frame,text="PRESENTAR GRAFICOS", foreground = "white",background =
193 B0.place(relx=0.05, rely=0.08,relheight=0.06, relwidth=1)
194
195 B1 =tk.Button(left_frame,text="LISTAR PERSONAS",background = "orange",command = listar
196 B1.place(relx=0.05, rely=0.25,relheight=0.06, relwidth=1)
197
198 B2 =tk.Button(left_frame,text="VALIDAR RIESGO PERSONA ", foreground = "white",backgrou
199 B2.place(relx=0.05, rely=0.45,relheight=0.06, relwidth=1)
200
201 B3 =tk.Button(left_frame,text="VER PELICULA ", foreground = "white",background = "brow
202 B3.place(relx=0.05, rely=0.68,relheight=0.06, relwidth=1)
203
204 B4 =tk.Button(left_frame,text="PELICULAS VISTAS ", foreground = "white",background = "
205 B4.place(relx=0.05, rely=0.82,relheight=0.06, relwidth=1)
206
207 #----COMBOBOX---------
208 combo = ttk.Combobox(left_frame, state="readonly")
209 combo.place(relx=0.05, rely=0.38,relheight=0.03, relwidth=1)
210
211 combo2 = ttk.Combobox(left_frame, state="readonly")
212 combo2.place(relx=0.05, rely=0.55,relheight=0.03, relwidth=1)
213
214 combo3 = ttk.Combobox(left_frame, state="readonly")
215 combo3.place(relx=0.05, rely=0.90,relheight=0.03, relwidth=1)
216
217
218
219 #-------entry
220
221 peli = Entry(left_frame, bd =5)
222 peli.place(relx=0.05, rely=0.60,relheight=0.03, relwidth=0.5)
223 #------------
224 #--- Agregar figura ---
225 figure = plt.Figure(figsize=(5,6), dpi=100)
226 ax = figure.add_subplot(322)
227 ax.grid(True),ax.set_xlabel('$x$'),ax.set_ylabel('$y(x)$')
228
229 ax1 = figure.add_subplot(321)
230 ax1.grid(True),ax1.set_xlabel('$x$'),ax1.set_ylabel('$y(x)$')
231
232
233 ax2 = figure.add_subplot(212)
234 ax2.grid(True),ax1.set_xlabel('$x$'),ax1.set_ylabel('$y(x)$')
235
236 line = FigureCanvasTkAgg(figure, right_frame)
237 line.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH,expand=1)
238 #----------------------
239
240 root.mainloop()
```

```
<-----SE EJECUTA LA LOGICA DIFUSA----->
*********JOVEN*************** 29 0.2
*********PORCENTAJE BAJO****** 13 0.8
VALOR DE LOS RESULTADOS DE EDADES >>  [['Medio', 'Alto', 'Alto', 0.2]]
VALOR DE LOS MANEJOS>>  [['Alto', 'Medio', 'Alto', 0.8]]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.8]
valor resultados es  > <neo4j.work.result.Result object at 0x00000189E6E7C60
8>
SE GENERA LA RELACION VER PELICULA Persona_22 Dunkerque
VALOR DE PERSONAS>>  Persona_22
[{'pe.nombre': 'Dunkerque'}]
<-----SE EJECUTA LA LOGICA DIFUSA----->
*********JOVEN*************** 29 0.2
*********PORCENTAJE BAJO****** 19 0.2
VALOR DE LOS RESULTADOS DE EDADES >>  [['Medio', 'Alto', 'Alto', 0.2], ['Med
io', 'Alto', 'Alto', 0.2]]
VALOR DE LOS MANEJOS>>  [['Alto', 'Medio', 'Alto', 0.8], ['Alto', 'Medio',
'Alto', 0.2]]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.8]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.2]


Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\ADMINX\anaconda3\lib\tkinter\__init__.py", line 1705, in __
call__
    return self.func(*args)
  File "<ipython-input-6-07d651ea277b>", line 125, in logicadif
    grafo.KNN()
  File "<ipython-input-3-71e535665a74>", line 13, in KNN
    greeting = session.write_transaction(self._EJECUTAR_KNN)
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\simple.py", l
ine 403, in write_transaction
    return self._run_transaction(WRITE_ACCESS, transaction_function, *args,
**kwargs)
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\simple.py", l
ine 309, in _run_transaction
    result = transaction_function(tx, *args, **kwargs)
  File "<ipython-input-3-71e535665a74>", line 79, in _EJECUTAR_KNN
    result = tx.run("CALL gds.beta.knn.stream('graficoknn12', {    topK: 1,
nodeWeightProperty: 'rating',    randomSeed: 42,    concurrency: 1,    sampl
eRate: 1.0,    deltaThreshold: 0.0}) YIELD node1, node2, similarity RETURN g
ds.util.asNode(node1).nombre AS Pelicula1, gds.util.asNode(node2).nombre AS
Pelicula2, gds.util.asNode(node2).rating AS Rating, similarity ORDER BY simi
larity DESCENDING, Pelicula1, Pelicula2").data()
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\transaction.p
y", line 118, in run
    result._tx_ready_run(query, parameters, **kwparameters)
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\result.py", l
ine 57, in _tx_ready_run
    self._run(query, parameters, None, None, None, **kwparameters)
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\result.py", l
ine 101, in _run
    self._attach()
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\work\result.py", l
ine 202, in _attach
    self._connection.fetch_message()
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\io\_bolt4.py", lin
```

```
e 363, in fetch_message
    response.on_failure(summary_metadata or {})
  File "C:\Users\ADMINX\anaconda3\lib\site-packages\neo4j\io\_common.py", li
ne 179, in on_failure
    raise Neo4jError.hydrate(**metadata)
neo4j.exceptions.ClientError: {code: Neo.ClientError.Procedure.ProcedureCall
Failed} {message: Failed to invoke procedure `gds.graph.create`: Caused by:
java.lang.IllegalArgumentException: A graph with name 'graficoknn12' already
exists.}

<-----SE EJECUTA LA LOGICA DIFUSA----->
*********JOVEN*************** 29 0.2
*********PORCENTAJE BAJO******* 19 0.2
VALOR DE LOS RESULTADOS DE EDADES >>  [['Medio', 'Alto', 'Alto', 0.2], ['Med
io', 'Alto', 'Alto', 0.2], ['Medio', 'Alto', 'Alto', 0.2]]
VALOR DE LOS MANEJOS>>  [['Alto', 'Medio', 'Alto', 0.8], ['Alto', 'Medio',
'Alto', 0.2], ['Alto', 'Medio', 'Alto', 0.2]]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.8]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.2]
EDAD-------> ['Medio', 'Alto', 'Alto', 0.2]
MANEJO-------> ['Alto', 'Medio', 'Alto', 0.2]
VALOR DE PERSONAS>>  Persona_9
[{'pe.nombre': 'Up'}, {'pe.nombre': 'Up'}, {'pe.nombre': 'Gravity'}, {'pe.no
mbre': 'Babel'}, {'pe.nombre': 'Babel'}, {'pe.nombre': 'Babel'}, {'pe.nombr
e': 'El_renacido'}, {'pe.nombre': 'El_renacido'}, {'pe.nombre': 'El_renacid
o'}]
```

In [ ]:

```
1
```

In [ ]:

```
1
```