Sistemas Expertos
Tema: Basados casos y lógica difusa.

Examen Final

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 recomendador, para lograr esto se describe los pasos a seguir:

1) **Evaluar 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.
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 20 películas y un total de nodos de al menos 250 nodos.
3) Con estos datos aplicar el algoritmo de KNN y Similitud de Coseno para la recomendación de películas, seguir el siguiente tutorial: https://guides.neo4j.com/sandbox/recommendations o https://github.com/MNoorFawi/recommendation-engine-with-neo4j o https://neo4j.com/developer/example-project/.
4) Finalmente realizar alguna interfaz para poder acceder a la recomendación e ingreso de datos y resultados de los procesos.

Generar el Informe en PDF y subir los scripts al repositorio Git para su evaluación.

Fecha de Entrega : **03/08/2021 - 13:55**

Criterios de Evaluación
- Sistema lógico difuso: 30%
- Neo4J Knn: 30%
- Informe y resultados: 20%
- GUI. programación y pruebas: 20%

Nota: Subir el sistema en un cuaderno de Python + scripts + PDF al Git personal.

Cualquier pregunta o inquietud por favor comunicarse por cualquier medio ya sea email, whatssapp o messenger.

Sistemas Expertos
Tema: Basados casos y lógica difusa.

Examen Final

Nombre: Carlos Alvarez

1. Reglas de Inferencia Difusa
    a. Tabla

```
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

#SE GENERA LA TABLA

tabla = (["","Joven","Adulto","Mayor"],["Bajo","Medio","Bajo","Medio"],["Medio","Alto","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
```

    b. Declaración de Funciones

```
class CLASE_NEO4J(object):
    def __init__(self):
        self._driver = GraphDatabase.driver("bolt:neo4j://localhost:7687", auth=("neo4j", "cuenca"), e
    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, nombre, rating)
            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, nombre, edad, por_mar
            print(greeting)
```

Examen Final

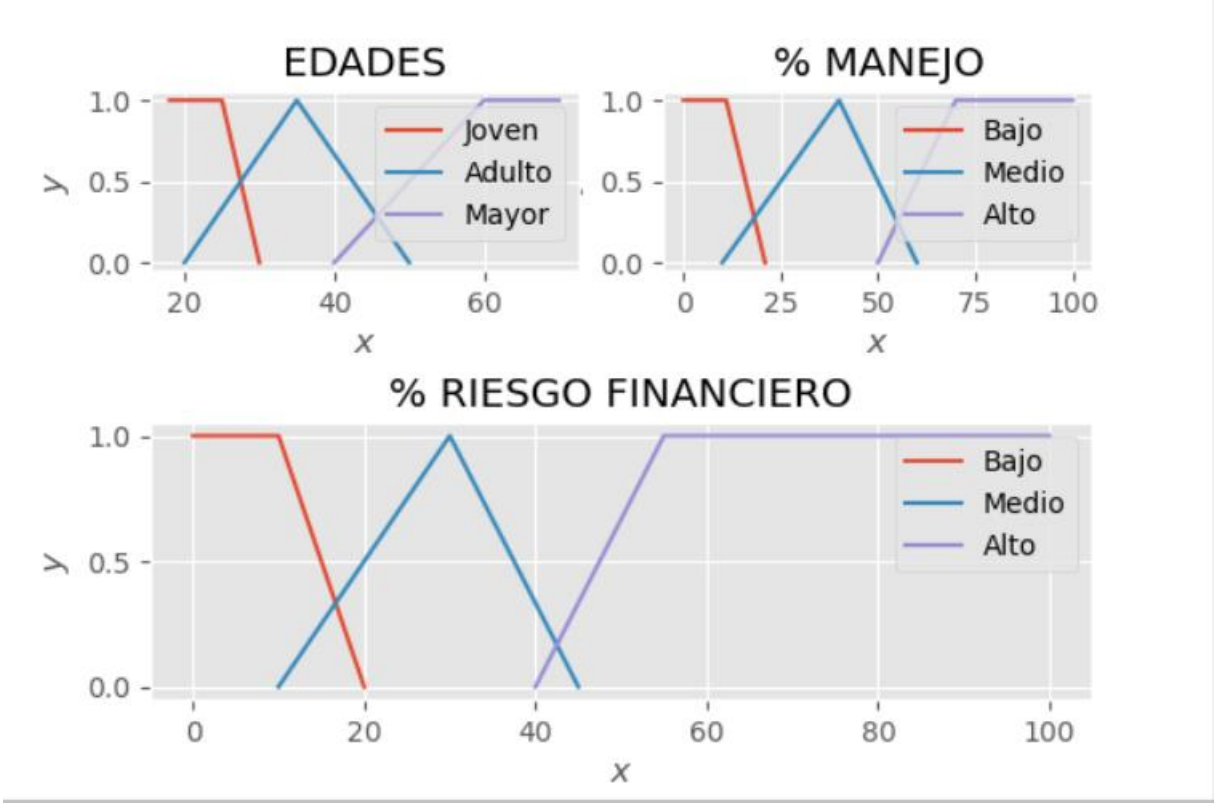
```
def logicadif():
    st = str(combo.get()).split("   :   ")
    print("<-----SE EJECUTA LA LOGICA DIFUSA-----> ")
    ax1.axvline(int(st[1]), label='pyplot vertical line',color='red')
    line.draw()
    ax.axvline(int(st[2]), label='pyplot vertical line',color='blue')
    line.draw()
    #VARIABLE 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",lx])
        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[13:33]
    for lx in listax[13:33]:
        if int(lx) == int(a):
            print("***********ADULTO********",lx,lisyn[cont1])
            resultados.insert(1,["Bajo","Medio","Alto",lx])
        elif int(lx) != int(a):
            pass
```

Examen Final

```
#SE BUSCA EL VALOR INGRESADO EN EL VECTOR DE X ESTA T ALTA
cont=0
lista = listay[34:]
for lx in listax[34:]:
    if int(lx) == int(a):
        print("**********MAYOR*********",lx,lista[cont])
        resultados.insert(2,["Medio","Alto","Alto",lx])
    elif int(lx) != int(a):
        pass
    cont =cont +1
#SE VALIDA  LOS VALORES DE MANEJO
cont2=0
lisyn4 = listay[19:]
for lx in listax[19:]:
    if int(lx) == int(b):
        print("**********PORCENTAJE ALTO*********",lx,lisyn4[cont2])
        resultadosm.insert(0,["Alto","Alto","Alto",lisyn4[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
lista5 = listay[8:19]
for lx in listax[8:19]:
    if int(lx) == int(b):
        print("***********PORCENTAJE MEDIO*********",lx,lista5[cont23])
        resultadosm.insert(1,["Alto","Medio","Alto",lista5[cont23]])
    elif int(lx) != int(b):
        pass
```

c. Graficas de la evaluación del riesgo financiero

Examen Final



2. Generar Números aleatorios para 100 personas y 20 películas

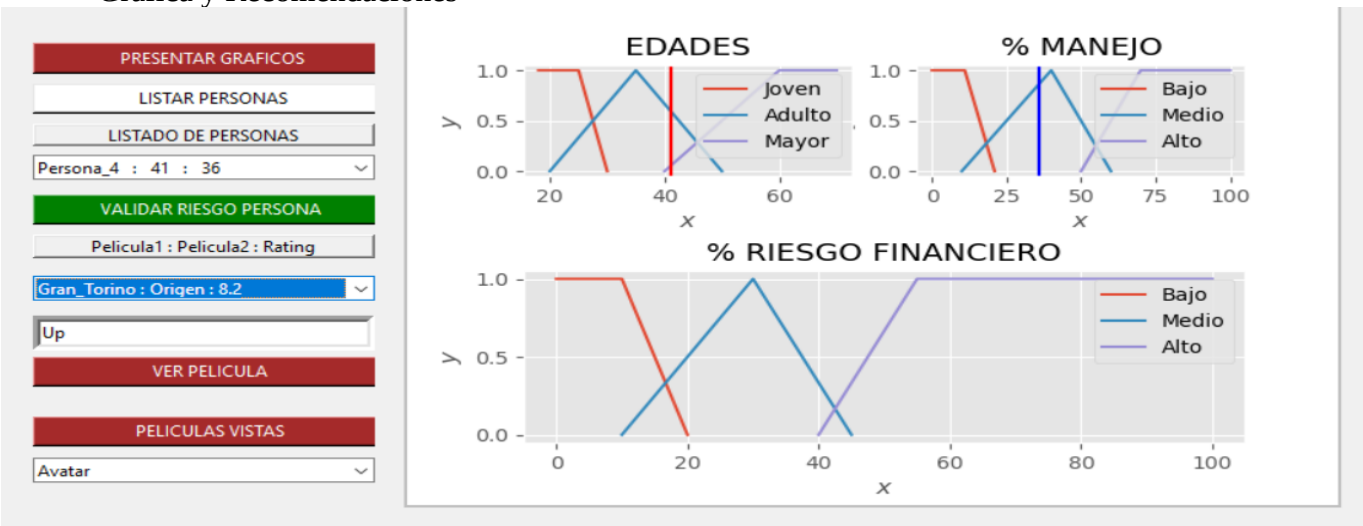
```
ll= []
with open('peliculas.csv', newline='') as File:
    reader = csv.reader(File)
    for row in reader:
        #print(row)
        ll.append(row)
        print(row[0])
        print(row[1])
        print(row[2])
        grafo.CREAR_PELICULA("SE GENERA UN NODO PELICULA EN LA BASE >>>>>>>>> ",str(row[1]),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

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(edad()), str(mar
```

Examen Final

3. Aplicación de los algoritmos

```
@staticmethod
def _EJECUTAR_KNN(tx):
    #SE GENERA EL GRAFO CON LOS NODOS PARA EL ALGORITMO DE KNN
    result1 = tx.run("CALL gds.graph.create('graficoknn17',{Pelicula: {label: 'Pelicula',properties
    print("Resultado 1"+str(result1))
    #SE EJECUTA ALGORITMO KNN
    result = tx.run("CALL gds.beta.knn.stream('graficoknn17', {    topK: 1,    nodeWeightProperty:
    lista =[]
    for io in result:
        var = str(io.get("Pelicula1"))+" : "+str(io.get("Pelicula2"))+" : "+str(io.get("Rating"))
        lista.append(var)
    combo2["values"]=lista
```

- Gráfica y Recomendaciones



4. Interfaz Grafica