



Pablo Moreno González - 100451061 - Grupo 81
Alberto Lozano Veiga - 100495914 - Grupo 81

D01 Informe del Proyecto

- Resumen Ejecutivo

Proyecto de Sistema de Ayuda a la Decisión para la Concesión de Préstamos
Según el enunciado de esta práctica, el Banco Pichin necesita mejorar su proceso de evaluación para la concesión de préstamos personales. Para ello, nosotros como grupo hemos desarrollado un Sistema de Inferencia Borrosa de Mamdani (abreviado MFIS) que permite evaluar el riesgo asociado a cada solicitud. Este sistema utiliza todos los conceptos de la lógica borrosa que vimos en el tema 8 para manejar la incertidumbre y la subjetividad a la hora de tomar decisiones, considerando variables como el nivel de ingresos, los bienes que posee, la estabilidad laboral, la cantidad que pide, el historial de préstamos y la edad.

El procedimiento comienza leyendo las solicitudes en Applications.txt. Luego, usa los conjuntos borrosos definidos en InputVarSet.txt para fuzzificar las

variables. Las reglas de inferencia, que se encuentran en Rule.txt, se utilizan para evaluar los antecedentes y los consecuentes, generando un valor de riesgo borroso que luego se desborrasifica para obtener un valor exacto. La clasificación del riesgo en niveles bajo, medio y alto se guarda en Resultados.txt.

Este sistema aumenta la eficacia y precisión del banco en la concesión de préstamos, permitiendo una gestión de recursos más eficiente y reduciendo el riesgo con las decisiones de crédito.

- Descripción del sistema de inferencia

El Sistema de Inferencia Borrosa de Mamdani (MFIS) desarrollado para el Banco Pichin está diseñado para evaluar el riesgo de conceder préstamos personales a los solicitantes. Este sistema utiliza lógica borrosa para manejar la incertidumbre y subjetividad inherentes a las decisiones de concesión de crédito, permitiendo al banco tomar decisiones más informadas y precisas.

Variables de Entrada:

Las variables de entrada consideradas incluyen el nivel de ingresos, bienes en posesión, estabilidad laboral, cantidad solicitada en relación a los ingresos, historial de préstamos y pagos, y edad. Cada variable se representa mediante conjuntos borrosos con funciones de pertenencia trapezoidales o triangulares. Estos conjuntos se definen en el archivo InputVarSet.txt.

Fuzzificación:

En la etapa de fuzzificación, los valores precisos de las variables de entrada se convierten en grados de pertenencia a los diferentes conjuntos borrosos. Por ejemplo, la edad de un solicitante puede tener un grado de pertenencia a los conjuntos "Joven" y "Adulto".

Reglas de Inferencia:

Las reglas de inferencia, están definidas en Rule.txt. Estas establecen relaciones entre las variables de entrada y el nivel de riesgo. Cada regla tiene una estructura del tipo "SI [antecedente] ENTONCES [consecuente]" .

Evaluación de Reglas:

Durante la evaluación de las reglas, se determinan los grados de pertenencia de los antecedentes y se calcula la fuerza de la regla (strength). Los grados de pertenencia de los consecuentes se determinan en función de la fuerza de la regla.

Agregación y Desborrosificación:

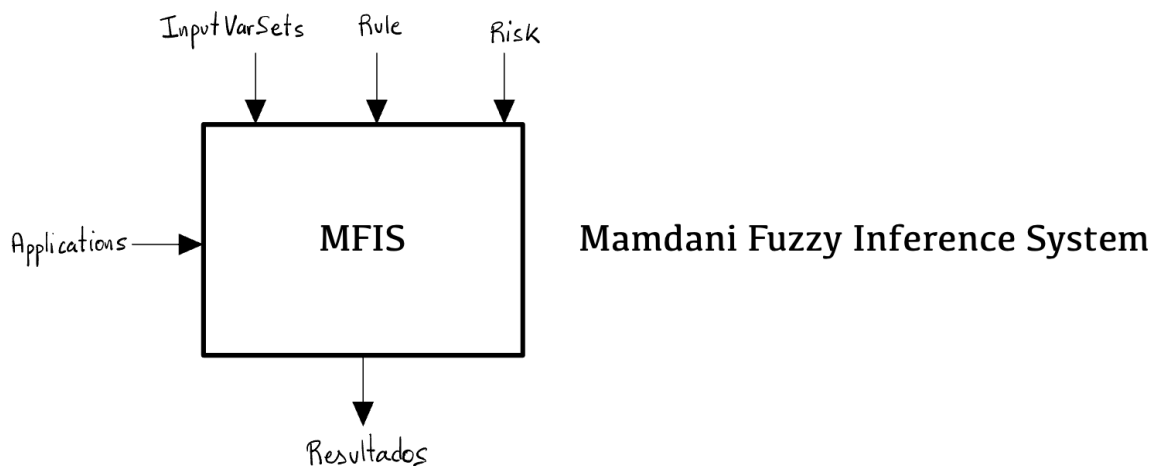
Las salidas borrosas de todas las reglas se agregan para formar un conjunto difuso final.

Este conjunto se desborrosifica para obtener un valor numérico preciso que representa el nivel de riesgo de la solicitud. El método de desborrosificación utilizado es el centroide, que calcula el "centro de masa" del conjunto borroso resultante.

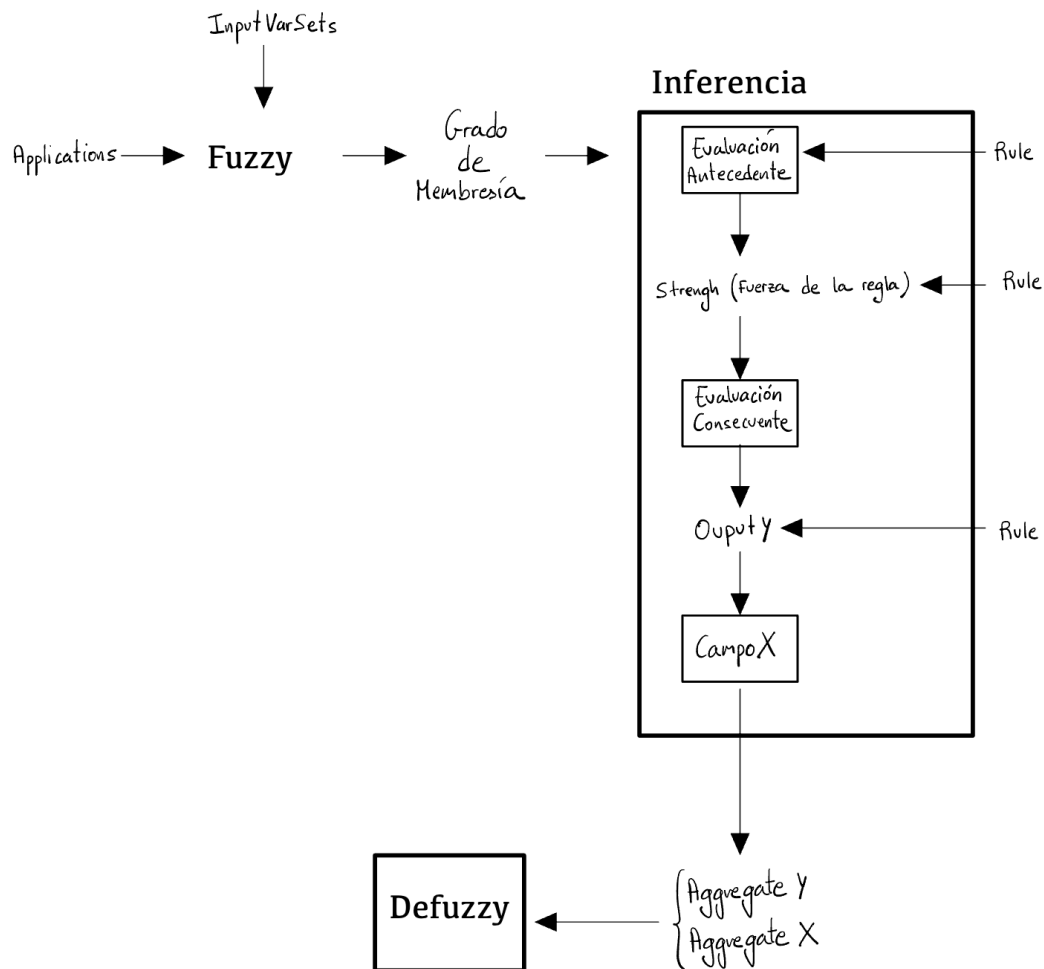
Salida del Sistema:

Los resultados de riesgo se almacenan en el archivo Resultados.txt, donde cada línea contiene el ID de la solicitud y el valor de riesgo correspondiente. Estos resultados ayudan al banco a clasificar las solicitudes en diferentes niveles de riesgo (bajo, medio, alto) y tomar decisiones de concesión de préstamos más informadas.

Descripción visual del diagrama, Diagrama de flujo del MFIS



Detalles del proceso de inferencia



- Metodología

El proyecto se ha llevado a cabo en 5 fases, a continuación, se describen las fases del proyecto, los objetivos y las tareas realizadas en cada una.

Fase 1: Definición de Variables

Objetivo: Identificar y definir las variables de entrada que influirán en la decisión de la concesión de los préstamos.

Tareas Realizadas:

Selección de Variables: Se seleccionaron las siguientes variables del archivo InputVarSets.txt: nivel de ingresos, bienes en posesión, estabilidad laboral, cantidad solicitada, historial de préstamos y pagos, y edad.

Definición de Conjuntos Borrosos: Para cada variable, definimos conjuntos borrosos que permiten representar adecuadamente la incertidumbre y la variabilidad.

1. Nivel de Ingresos:

IncomeLevel=Low, 0, 150, -20, -10, 25, 40; Representa ingresos bajos.

IncomeLevel=Med, 0, 150, 20, 30, 50, 80; Representa ingresos medios.

IncomeLevel=Hig, 0, 150, 40, 80, 160, 170; Representa ingresos altos.

2. Bienes en Posesión:

Assets=Scarce, 0, 50, -2, -1, 5, 20; Representa pocos bienes en posesión.

Assets=Moderate, 0, 50, 5, 10, 20, 30; Representa una cantidad moderada de bienes.

Assets=Abundant, 0, 50, 25, 30, 60, 70; Representa una gran cantidad de bienes.

3. Estabilidad Laboral:

Job=Unstable, 0, 5, -2, -1, 1, 2; Representa empleo inestable.

Job=Stable, 0, 5, 2, 3, 6, 7; Representa empleo estable.

4. Cantidad Solicitada:

Amount=Small, 0, 8, -2, -1, 1, 3; Representa una cantidad pequeña en relación a los ingresos.

Amount=Medium, 0, 8, 1, 3, 3, 5; Representa una cantidad media en relación a los ingresos.

Amount=Big, 0, 8, 3, 5, 5, 7; Representa una cantidad grande en relación a los ingresos.

Amount=VeryBig, 0, 8, 5, 7, 11, 12; Representa una cantidad muy grande en relación a los ingresos.

5. Historial de Préstamos y Pagos:

History=Poor, 0, 6, -2, -1, 1, 3; Representa un historial de pagos pobre.

History=Standard, 0, 6, 1, 2, 4, 5; Representa un historial de pagos estándar.

History=Good, 0, 6, 3, 5, 8, 9; Representa un buen historial de pagos.

6. Edad:

Age=Young, 0, 100, -20, -10, 30, 40; Representa una edad joven.

Age=Adult, 0, 100, 20, 30, 50, 65; Representa una edad adulta.

Age=Elder, 0, 100, 50, 65, 150, 175; Representa una edad mayor.

Fase 2: Definición de Reglas

Objetivo: Crear un conjunto de reglas de inferencia basadas en la lógica borrosa que permitan evaluar el riesgo de cada solicitud de préstamo.

Tareas Realizadas:

Formulación de Reglas: Analizamos el archivo Rules.txt que contiene las reglas. Hay un total de 50 reglas definidas que cubren numerosas posibles combinaciones de variables de entrada y los niveles de riesgo asociados.

Documentación: Registramos todas las reglas en el archivo Rule.txt para su implementación en el sistema de inferencia. Este archivo contiene la representación estructurada de cada regla, indicando los antecedentes y el consecuente.

Fase 3: Implementación del Sistema de Inferencia

Objetivo: Desarrollar el sistema de inferencia borrosa utilizando las definiciones de variables y reglas establecidas en las fases anteriores.

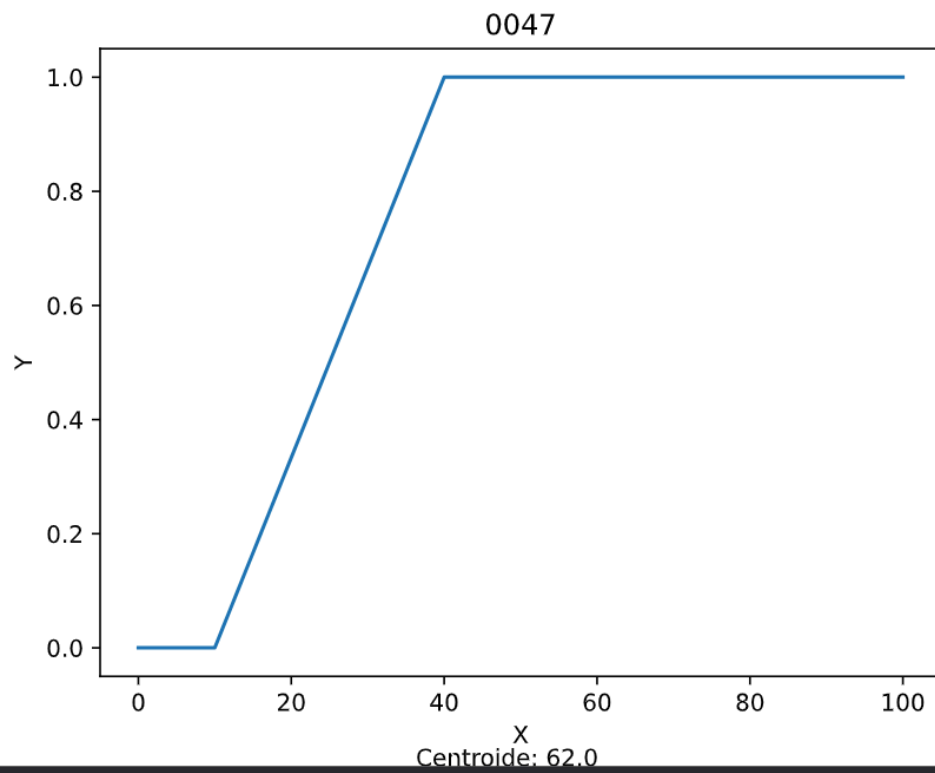
Tareas Realizadas:

Configuración del Entorno de Desarrollo: Instalamos y configuramos las librerías necesarias, como numpy, skfuzzy y matplotlib.

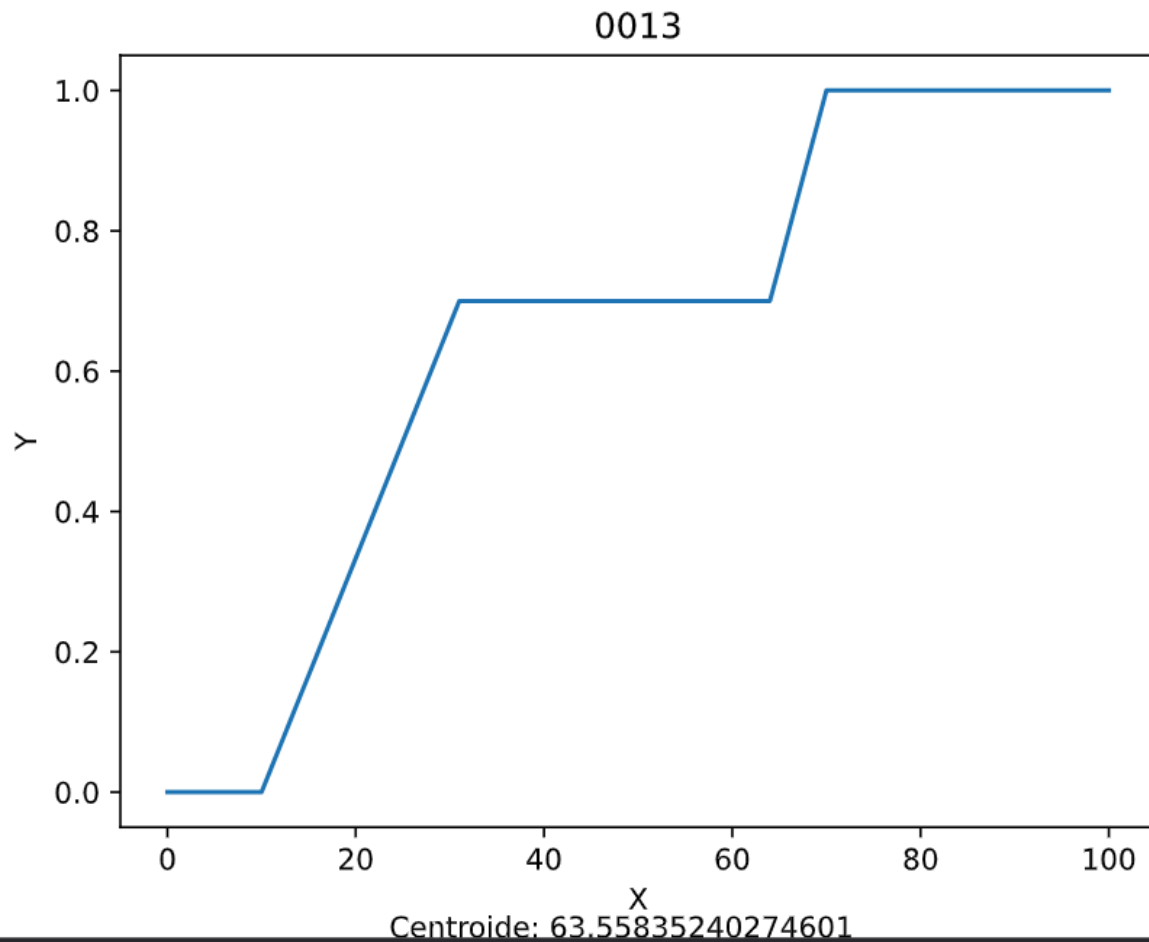
Desarrollo del Sistema: Implementamos el MFIS siguiendo los principios de la lógica borrosa. El proceso incluyó la creación de funciones para leer los archivos de configuración (InputVarSet.txt, Rule.txt, Risk.txt), y estas funciones se implementaron en MFIS_Read_Functions.py. También incluyó las clases para manejar los conjuntos borrosos, reglas y aplicaciones se definieron en MFIS_Classes.py. y por último incluyó la implementación de los algoritmos de fuzzificación, evaluación de reglas, composición, desborrosificación y generación de resultados; realizado en codigo_fuente.py. dichos resultados están contenidos en el archivo Resultados.txt.

Importante ver la carpeta plot, con 50 imágenes (.svj) donde se ve el riesgo, adjuntamos 2 por ejemplo en la memoria para que se vea

Por ejemplo para la ID 47, con 0047, Age, 46, IncomeLevel, 61, Assets, 28, Amount, 7, Job, 4, History, 5 sale un riesgo de 62 para la concesión del préstamo



Por ejemplo para la ID 13, con 0013, Age, 39, IncomeLevel, 59, Assets, 21, Amount, 7, Job, 5, History, 3, sale un riesgo de 63.55835240274601 para la concesión del préstamo



Fase 4: Pruebas y Validación

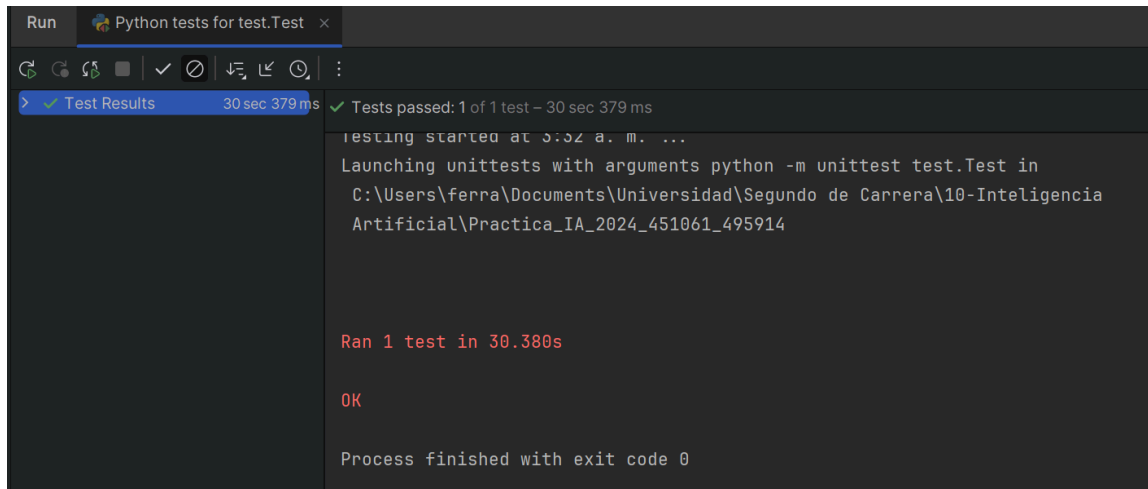
Objetivo: Asegurar que el sistema de inferencia funcione correctamente y proporcione resultados precisos y consistentes.

Tareas Realizadas:

Generación de Datos de Prueba: Creamos un conjunto de datos de prueba representativos de diversas situaciones de solicitud de préstamo. Estos datos incluyeron una variedad de combinaciones de variables de entrada para asegurar que se cubrieran múltiples escenarios posibles.

Pruebas Unitarias: Se implementaron pruebas unitarias para verificar que cada componente del sistema funcionara correctamente. Estas pruebas se centraron en la verificación de los algoritmos de fuzzificación, evaluación de reglas, composición, desborrosificación y generación de resultados.

Ejecución de Pruebas: Ejecutamos el sistema con los datos de prueba y revisamos los resultados para detectar y corregir cualquier error. Los resultados obtenidos de las pruebas se compararon con los valores esperados para asegurar la precisión del sistema.



The screenshot shows a terminal window titled "Python tests for test.Test". The output indicates that the tests passed successfully. The text in the terminal is as follows:

```
testing started at 3:52 a. m. ...
Launching unittests with arguments python -m unittest test.Test in
C:\Users\ferre\Documents\Universidad\Segundo de Carrera\10-Inteligencia
Artificial\Practica_IA_2024_451061_495914

Ran 1 test in 30.380s

OK

Process finished with exit code 0
```

El método `helper_esperado` calcula el riesgo esperado para una solicitud, utilizando conjuntos borrosos y reglas definidas en archivos de configuración. Convierte variables y riesgos en objetos de control, procesa las reglas y simula el sistema de control borroso. Luego, evalúa el riesgo de la solicitud basándose en los datos proporcionados y devuelve el resultado. Esto se compara en el método `test` para validar la precisión del sistema de inferencia borrosa.

Fase 5: Documentación

Objetivo: Entregar la documentación completa del proyecto.

Tareas Realizadas:

Elaboramos la documentación completa del proyecto, incluyendo:

Informe del Proyecto: Una descripción detallada del proyecto, el sistema de inferencia y las metodologías utilizadas.

Código Fuente: Todos los archivos de código necesarios para ejecutar y mantener el sistema, con comentarios detallados.

Resultados: Archivos de salida generados por el sistema de inferencia como respuesta a las solicitudes de préstamo.

Análisis: Respuestas a preguntas y análisis adicionales realizados durante el proyecto.

- Presupuesto

Como estudiantes de segundo de carrera de Ingeniería Informática en un proyecto de Inteligencia Artificial, hemos trabajado en este proyecto de implementación de Inferencia Borrosa de Mamdani. A continuación, presentamos un presupuesto adaptado a nuestra experiencia y nivel académico, considerando la dedicación y esfuerzo invertido en el desarrollo del sistema.

Horas de Trabajo:

Análisis y Planificación:

Horas: 4

Tarifa por hora: 15 EUR

Total: 60 EUR

Desarrollo y Codificación:

Horas: 10

Tarifa por hora: 15 EUR

Total: 150 EUR

Pruebas y Validación:

Horas: 2

Tarifa por hora: 15 EUR

Total: 30 EUR

Documentación y Presentación:

Horas: 8

Tarifa por hora: 15 EUR

Total: 120 EUR

Gastos Adicionales:

Materiales y Software:

Coste estimado: 10 EUR

‘cuenta en sucio sería $(60 + 150 + 30 + 120) = 360$ euros’

Hemos decidido poner estos precios ya que somos solo dos simples estudiantes de segundo de carrera que estamos comenzando a practicar en el mundo real de la ingeniería informática. Aunque nuestra experiencia práctica en la vida real en empresas de verdad es limitada, pues contamos con un sólido conocimiento teórico que nos permite desenvolvernó bien y que tantas horas hemos estado estudiando a lo largo de la carrera. En comparación con profesionales avanzados, este presupuesto es bastante bajo, lo tenemos en cuenta. Consideramos que es un precio razonable y justo para ser nuestro primer proyecto 'real' y una buena manera de empezar a ganar experiencia y que la gente se interese por nosotros.

Subtotal: 360 EUR

IVA (21%): 75.60 EUR

Total con IVA: 435.60 EUR

Total a Pagar: 435.60 EUR

Datos del Banco:

Nombre: Banco Pichin

Dirección: prestamos_clientes_pichin@gmail.com

Datos del Proveedor:

Nombre: Pablo Moreno González y Alberto Lozano Veiga

Dirección: 100451061@alumnos.uc3m.es; 100495914@alumnos.uc3m.es

Método de Pago: Transferencia Bancaria

Número de Cuenta: ES12 3456 7890 12 123456789

Observaciones:

Gracias por confiar en nuestros servicios. Para cualquier consulta, no dude en contactarnos.

Firmado por Pablo Moreno y Alberto Lozano:

Información Legal:

Esta factura cumple con todos los requisitos legales y fiscales aplicables. La tasa de IVA aplicada es del 21%.

Nota: Por favor, realizar el pago dentro de los próximos 30 días.

D02 Vídeo

Adjuntamos en nuestra carpeta dicho video que hemos preparado con una duración de 5:31 minutos

D03 Código fuente

A continuación, adjuntamos el programa en Python que codifica el Sistema de Inferencia Borrosa de Mamdani desarrollado para el Banco Pichin. (En la carpeta adjuntaremos también el código entero, para que lo ejecutéis y comprobéis)

codigo_fuente.py:

```
from pathlib import Path

import matplotlib.pyplot as plt
import numpy
import skfuzzy as skf

import MFIS_Read_Functions as lectura
from MFIS_Classes import *

# Constantes

FICHERO_APLICACIONES = Path('./Applications.txt')
FICHERO_INPUTVAR = Path('./InputVarSets.txt')
FICHERO_RESULTADOS = Path('./Resultados.txt')
FICHERO_RIESGOS = Path('./Risks.txt')
FICHERO_REGLAS = Path('./Rules.txt')
DIRECTORIO_PLOTS = Path('./plots')

def escribir_resultado(archivo: Path, resultados: list[dict]):
    # Escribe el resultado en un fichero
    with open(archivo, "w") as outputFile:
        for app_id, centroide in resultados.items():
            string_de_escritura = f"{app_id} {centroide}\n"
```

```

        outputFile.write(string_de_escritura)

def borrosificacion(aplicacion: Application, lista_varset: dict) -> dict:
    aplicacion_borrosificada = {"app_id": aplicacion.app_id}
    # Itero por las variables de la aplicacion
    for variable, valor in aplicacion.data:
        aplicacion_borrosificada[variable] = {}
        # Busco las varset correspondientes
        for varset in lista_varset.values():
            if varset.variable == variable:
                aplicacion_borrosificada[variable][varset.label] =
                    skf.interp_membership(varset.x, varset.y, valor)
        return aplicacion_borrosificada

def evaluacion_de_reglas(aplicacion_borrosificada: dict, reglas: dict) ->
dict:
    for regla in reglas.values():
        valores_antecedentes = []
        for antecedente in regla.antecedents:
            s = antecedente.split('=')
            variable, conjunto = s[0], s[1]

            valores_antecedentes.append(aplicacion_borrosificada[variable][conjunto])
            regla.strength = min(valores_antecedentes)
        return reglas

def calculo_de_consecuente(riesgos: dict, reglas: dict) -> tuple[dict]:
    # Calculo de activaciones
    activacion_riskL = 0
    activacion_riskM = 0
    activacion_riskH = 0
    for regla in reglas.values():
        s = regla.consequent.split('=')
        tipo_riesgo = s[1]
        match tipo_riesgo:
            case 'LowR':
                activacion_riskL = max(regla.strength, activacion_riskL)
            case 'MediumR':
                activacion_riskM = max(regla.strength, activacion_riskM)
            case 'HighR':
                activacion_riskH = max(regla.strength, activacion_riskH)

    # Recorte de funciones
    riesgoL_ajustado = {"x": riesgos['LowR'].x,
                        "y": numpy.clip(riesgos['LowR'].y, 0,
activacion_riskL)}

```

```

    riesgoM_ajustado = {"x": riesgos['MediumR'].x,
                        "y": numpy.clip(riesgos['MediumR'].y, 0,
activacion_riskM)}
    riesgoH_ajustado = {"x": riesgos['HighR'].x,
                        "y": numpy.clip(riesgos['HighR'].y, 0,
activacion_riskH)}
    return (riesgoL_ajustado, riesgoM_ajustado, riesgoH_ajustado)

def composicion(funciones_riesgo: tuple[dict]) -> dict:
    # Agregacion
    # Nota: Asumo que todas las funciones de riesgo tienen el mismo rango
de x (0, 100)
    funcion_agregada = {"x": funciones_riesgo[0]["x"],
                        "y": numpy.maximum(funciones_riesgo[0]["y"],
numpy.maximum(funciones_riesgo[1]["y"], funciones_riesgo[2]["y"]))}
    return funcion_agregada

def desborrosificacion(x: numpy.ndarray | list | tuple, y: numpy.ndarray
| list | tuple):
    return skf.centroid(x, y)

def imprimir_funcion(nombre: str, x: numpy.ndarray | list, y:
numpy.ndarray | list, centroide: int) -> None:
    plt.clf()
    plt.plot(x, y)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.title(nombre)
    plt.figtext(0.5, 0.0, f'Centroide: {centroide}', ha='center')
    if not DIRECTORIO_PLOTS.exists():
        DIRECTORIO_PLOTS.mkdir(exist_ok=True)
    plt.savefig(DIRECTORIO_PLOTS / (nombre + '.svg'), format='svg')

def main():
    aplicaciones: dict =
lectura.readApplicationsFile(FICHERO_APLICACIONES)
    inputvar: dict = lectura.readFuzzySetsFile(FICHERO_INPUTVAR)
    reglas: dict = lectura.readRulesFile(FICHERO_REGLAS)
    riesgos: dict = lectura.readRisksFile(FICHERO_RIESGOS)
    resultados: dict = {}
    for aplicacion in aplicaciones.values():
        aplicacion_borrosificada = borrosificacion(aplicacion, inputvar)
        reglas = evaluacion_de_reglas(aplicacion_borrosificada, reglas)
        funciones_ajustadas = calculo_de_consecuente(riesgos, reglas)
        funcion_agregada = composicion(funciones_ajustadas)

```

```

        centroide = desborrosificacion(funcion_agregada["x"],
funcion_agregada["y"])
        resultados[aplicacion.app_id] = centroide
        imprimir_funcion(aplicacion.app_id, funcion_agregada['x'],
funcion_agregada['y'], centroide)
        escribir_resultado(FICHERO_RESULTADOS, resultados)

if __name__ == '__main__':
    main()

```

D04 Resultados

El archivo Resultados.txt contiene los valores de riesgo calculados por el Sistema de Inferencia Borrosa de Mamdani (MFIS) para cada solicitud de préstamo, basados en los datos proporcionados en Applications.txt. Cada línea del archivo incluye el identificador de la solicitud seguido del valor de riesgo correspondiente.

Cada linea sigue el formato:
 <ID de solicitud> <Valor de riesgo>

ID de solicitud: Identificador único para cada solicitud de préstamo (por ejemplo, 0001, 0002).

Valor de riesgo: Valor numérico que representa el nivel de riesgo asociado con la solicitud, calculado por el sistema de inferencia borrosa.

Estos valores de riesgo permiten al Banco Pichin clasificar las solicitudes de préstamos en diferentes niveles de riesgo (bajo, medio, alto) y tomar decisiones más informadas sobre la concesión de préstamos.

```

0001  58.63636363636363
0002  50.497222222222184
0003  63.55835240274601
0004  52.01103057043807

```


0005	66.57156048014765
0006	67.28260869565217
0007	62.0
0008	54.85342388228637
0009	51.71265189421013
0010	57.45098039215693
0011	62.0
0012	42.54901960784315
0013	63.55835240274601
0014	62.85852713178293
0015	58.63636363636363
0016	52.66776348589084
0017	54.85342388228637
0018	51.384615384615344
0019	62.0
0020	46.82183908045982
0021	63.06889763779529
0022	43.099593901870726
0023	49.056175729204135
0024	62.0
0025	40.27848101265818
0026	52.539178314273585
0027	69.30326543808599
0028	61.785871964679885
0029	62.0
0030	62.85852713178293
0031	51.66586646661661
0032	57.45098039215693
0033	49.46741765293024
0034	65.70921985815603
0035	61.35294117647064
0036	62.46270928462709
0037	51.956521739130444
0038	62.14430563099765
0039	62.85852713178293
0040	49.06400575332611
0041	43.412392338346805
0042	54.85342388228637
0043	46.24183006535944
0044	61.570175438596564

```
0045 54.2570590392372
0046 46.11295965171422
0047 62.0
0048 51.77323876423654
0049 57.70100189547788
0050 56.7263969171484
```

D05 Análisis

Q1 - Problema:

Los expertos observan que los resultados no son siempre los deseados. Ello se debe a que algunas reglas son muy importantes, que ellos consideran “reglas de oro”, mientras que otras pueden servir en algunos casos, pero tienen menos relevancia en el riesgo. ¿Cómo se podría modificar el sistema de inferencia de Mamdani para solucionar este problema?

Para solucionar el problema de que algunas reglas son consideradas "reglas de oro" y tienen más importancia que otras, se puede modificar el sistema de inferencia de Mamdani introduciendo ponderaciones o pesos para las reglas. Aquí está cómo se podría hacer:

1. Asignación de Pesos a las Reglas:

- Identificación: Identificar y categorizar las reglas en términos de su importancia (por ejemplo, reglas de oro y reglas secundarias).
- Asignación de Pesos: Asignar un peso a cada regla basado en su importancia. Las reglas de oro recibirán pesos más altos en comparación con las reglas secundarias.

2. Modificación del Proceso de Evaluación:

- Evaluación de Antecedentes: Calcular los valores de pertenencia para los antecedentes como siempre.
- Aplicación de Pesos: Multiplicar la fuerza de cada regla (strength) por su peso correspondiente antes de calcular el consecuente.

3. Actualización de la Composición y Desborrosificación:

- Asegurarse de que los pesos sean considerados durante la agregación y desborrosificación de los resultados.

```
def evaluacion_de_reglas(aplicacion_borrosificada:
dict, reglas: dict) -> dict:

    for regla in reglas.values():

        valores_antecedentes = []

        for antecedente in regla.antecedents:

            s = antecedente.split('=')

            variable, conjunto = s[0], s[1]

valores_antecedentes.append(aplicacion_borrosificada[v
variable][conjunto])

            regla.strength = min(valores_antecedentes) *
regla.weight # Aplicar el peso aquí

    return reglas
```

Las reglas de oro tendrían un peso mayor, lo que aumentará su influencia en la inferencia del riesgo.

Q2 - Problema:

Otra observación consiste en que nunca se obtienen valores extremos de riesgo. ¿A qué se puede deber? ¿Cuál es el valor máximo que se puede obtener con el sistema descrito?

El problema de no obtener valores extremos de riesgo (muy bajos o muy altos) podría deberse a:

1. Reglas y Funciones de Pertenencia: Las reglas difusas y las funciones de pertenencia podrían no estar configuradas para cubrir adecuadamente los valores extremos.

2. Superposición de Conjuntos: Los conjuntos difusos para las diferentes categorías de riesgo podrían estar demasiado solapados, lo que lleva a una predominancia de valores medios.
3. Evaluación y Composición: El proceso de evaluación de reglas y composición podría estar suavizando demasiado las contribuciones individuales, resultando en valores promedio.

Valor Máximo que se Puede Obtener

Con el sistema descrito, el valor máximo de riesgo se define por el conjunto difuso Risk=HighR, cuyo valor máximo en la función de pertenencia es 100, pero puede llegar a extenderse ligeramente más allá de este valor debido a la configuración de la función (100 a 111). Por lo tanto, el valor máximo que se puede obtener es 111.

La causa probable de no obtener valores extremos de riesgo está en cómo se evalúan y combinan las reglas. Ajustando las reglas y asegurándose de que los extremos de las funciones de pertenencia se usan correctamente, se pueden obtener valores extremos de riesgo más frecuentemente.

Q3 - Problema:

El Banco Pichin ha sido adquirido por otro mucho mayor. En el plazo de un mes, el sistema deberá procesar cientos de veces de solicitudes más que ahora y se quiere obtener el resultado en muy pocos segundos para impresionar al nuevo dueño. No se puede gastar nada en hardware y todas las optimizaciones posibles al software ya han sido realizadas. ¿Cómo podría conseguirse?

Podríamos conseguirlo reduciendo la complejidad del problema de lógica borrosa:

1. Reduciendo el número de variables de entrada: Cogemos solo las variables que tengan más peso en el resultado final
2. Reducir el número de funciones de pertenencia: Usar funciones menos restrictivas que cubran un mayor número de entradas

3. Reducir el número de reglas borrosas: Usar las reglas que más influyen en el resultado final y combinar reglas similares