ANÁLISIS:

Un año después de la entrega del proyecto, surgen una serie de preguntas relacionadas con él o con el sistema de inferencia de Mamdani en general. En este entregable deberá dar respuesta a estas preguntas lo mejor que pueda. Las preguntas son:

Q1. 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?

Q2. 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?

Q3. 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?

Q1: Modificación del sistema para manejar la importancia variable de las reglas

Para abordar el problema de la importancia variable de las reglas en un sistema de inferencia de Mamdani, una buena solución es introducir pesos en las reglas que nuestros clientes consideran más importantes. Los pesos permitirían modificar la influencia de cada regla en la decisión final. Este enfoque se manejaría de la siguiente manera:

- **Asignación de Pesos**: Cada regla podría tener un peso asociado que refleje su importancia o confiabilidad, determinado por los expertos.
- **Modificación del Cálculo de la Fuerza del Antecedente**: Al calcular la fuerza de los antecedentes, este peso se multiplicaría por la fuerza calculada de la regla, ajustando así la contribución de cada regla al conjunto de salida final.
- **Ajuste en la Agregación:** En la etapa de agregación de las salidas de todas las reglas, se tendrán en cuenta estos pesos para priorizar las reglas más importantes.

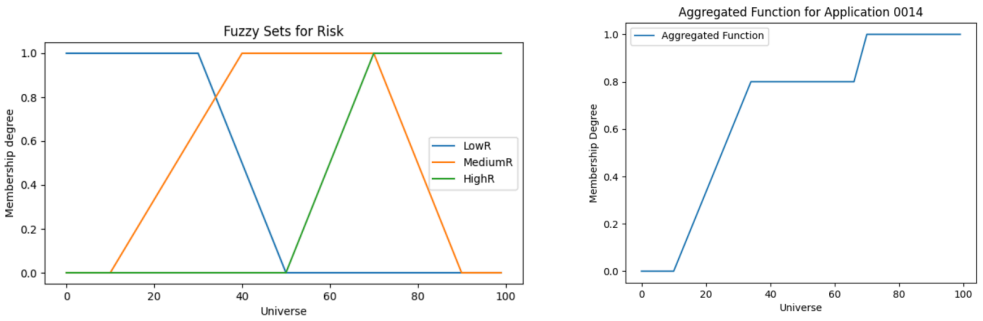
Estos cambios requerirían una modificación del código para incluir el manejo de pesos y su aplicación durante las fases de evaluación y agregación de reglas. Este código tendría que incluir los siguientes cambios:

1. **Modificación de la clase 'Rule'**: necesitamos agregar un nuevo atributo a la clase 'Rule' 'para almacenar el peso de cada regla. Esto implicaría modificar la definición de la clase en el archivo 'MFIS_Classes.py'.
2. **Lectura de Pesos desde Archivo de Reglas**: modificar 'readRulesFile' para leer el peso de las reglas desde el archivo, suponiendo que el formato de las reglas incluya el peso al final.
3. **Uso de Pesos en la Evaluación y Composición de Reglas**: modificar las funciones de evaluación de antecedente y consecuente para usar el peso de la regla. Esto influirá en cómo se calcula la fuerza del antecedente y cómo se combina con el consecuente:

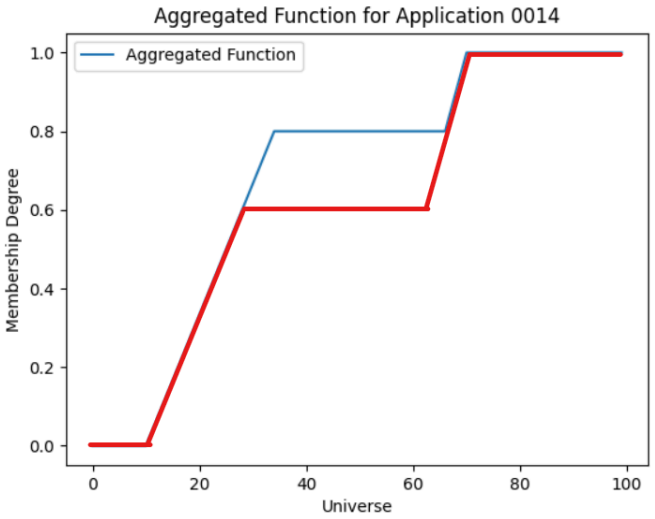
Para la selección de las reglas de oro, vamos a suponer que se trata de un banco conservador, y por tanto, el riesgo se pretende que sea el menor posible, y el riesgo alto nos va a preocupar especialmente. Por ello, vamos a asignar una mayor prioridad a las reglas que desembocan en un riesgo alto, una prioridad media a las de riesgo medio y una prioridad baja a las que llevan asociado un riesgo bajo. Para ello vamos a asignar un peso de 1 a las que el consecuente es riesgo alto, 0'75 a las de riesgo medio y 0'5 a las de riesgo bajo.

De esta forma, las áreas agregadas serán modificadas, achatando aquellas zonas de un riesgo menor, de forma que el centroide se desplazará a la derecha, dando una mayor importancia al riesgo alto.

Por ejemplo, con la siguiente función de riesgo, y la application 0014, en el que el centroide estará aproximadamente en 62,3:



Tras aplicar nuestra propuesta, el riesgo medio (correspondiente al primer escalón), se ve reducido a 0,6 (0,8*0,75), dándole una menor importancia, de forma que el centroide como se puede observar se desplaza aproximadamente a 70.



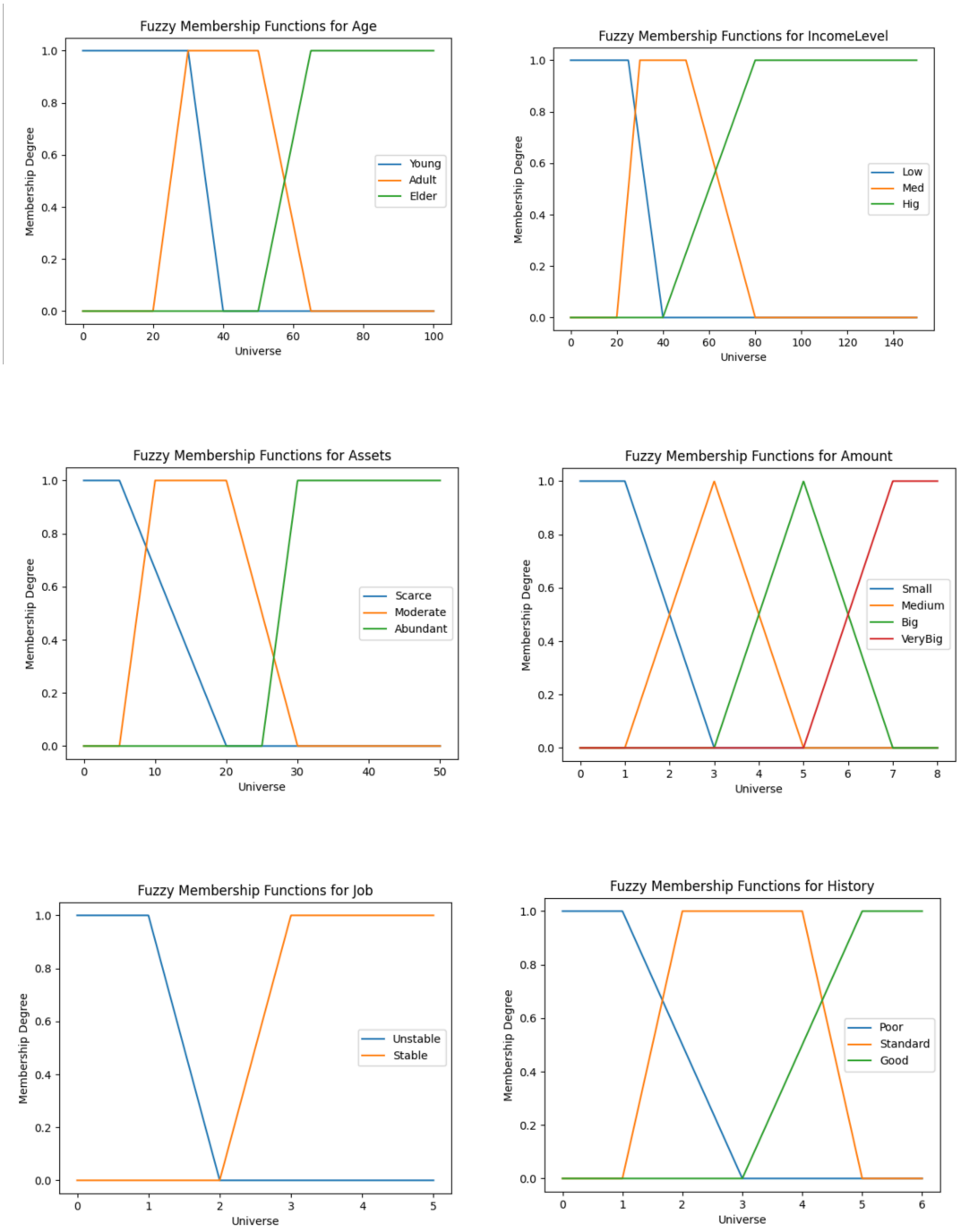
Q2: Falta de valores extremos en los resultados de riesgo

Para abordar esta cuestión, debemos tener un amplio conocimiento sobre el sistema Mamdani y su funcionamiento:

1) **Fuzzificación:** El primer paso en un sistema de inferencia de Mamdani es la fuzzificación de las entradas. En este caso, esto involucra convertir valores numéricos crudos en grados de pertenencia respecto a conjuntos borrosos definidos para cada variable de entrada. Por ejemplo, si estás evaluando el riesgo basado en variables como la edad o los ingresos del solicitante, cada una de estas variables tendrá asociadas funciones de membresía (como 'joven', 'adulto', 'ingresos altos', 'ingresos bajos').

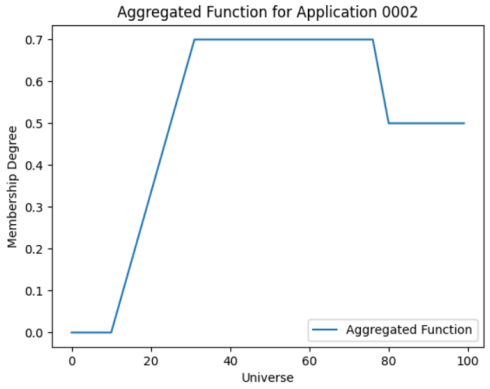
   Cada función de pertenencia asigna un valor entre 0 y 1 que representa qué tan cierto es que la entrada pertenezca a un conjunto borroso específico. Esto se hace usualmente mediante funciones trapezoidales o triangulares especificadas en el archivo ‘InputVarSets’.

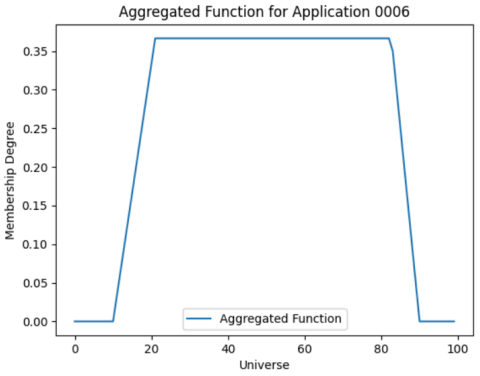
Este es el dibujo de las etiquetas:



2) **Aplicación de las Reglas**: Una vez que las entradas han sido convertidas a valores borrosos, el siguiente paso es evaluar las reglas del sistema. Las reglas en un sistema de inferencia de Mamdani tienen la forma "SI antecedente ENTONCES consecuente", donde el antecedente involucra una combinación de condiciones sobre las variables de entrada y el consecuente especifica un conjunto borroso para la variable de salida. Cada regla es evaluada para determinar su grado de activación basado en los grados de pertenencia de las entradas. El grado de activación de una regla es el mínimo de los grados de membresía de los antecedentes (ya que las reglas usan el operador lógico "Y").

3) **Agregación**: Después de evaluar todas las reglas, los consecuentes de cada regla (cada uno modificado por el grado de activación de su regla correspondiente) deben ser combinados en una única función de membresía de salida. Esto se realiza usando el operador máximo, donde el grado de pertenencia en cada punto del universo de salida es el máximo de los grados de membresía proporcionados por todas las reglas activas para ese punto. Una vez realizado esto podemos visualizar cómo quedarían las funciones agregadas (para consultar todas las agregaciones, ir al *código fuente/figuras_agregadas*):





4) **Defuzzificación (Cálculo del Centroide)**: Finalmente, el conjunto borroso resultante de la etapa de agregación debe ser convertido en un valor numérico único que represente la salida del sistema. Esto se hace a través de la defuzzificación. El método del centroide, que es común en los sistemas de Mamdani, calcula este valor como el centro de gravedad del conjunto borroso resultante.

   Matemáticamente, el centroide se calcula usando la fórmula:

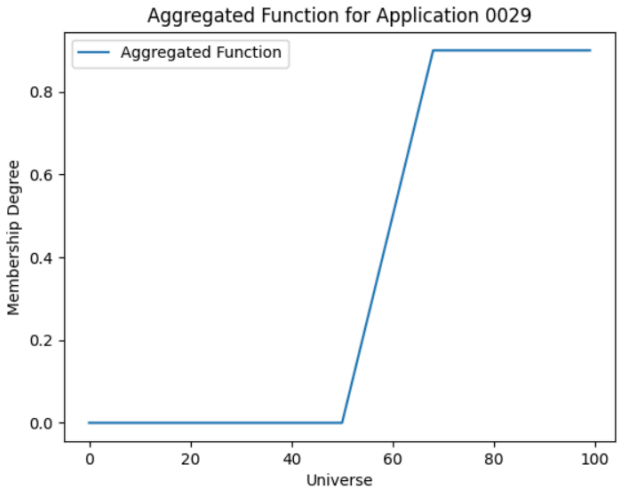
$$\text{Centroide} = \frac{\sum(x \cdot \mu(x))}{\sum \mu(x)}$$

donde x son los puntos en el universo de salida y $\mu(x)$ son los grados de membresía en esos puntos. Este cálculo efectivamente pondera cada punto por su grado de membresía y calcula el promedio ponderado.

Un nivel de riesgo extremo se corresponde con un valor del centroide 100. No obstante, como podemos observar, al hacer el cálculo de las agregaciones el resultado es un área que se encuentra entre las posiciones 0 y 100 que indica el nivel riesgo, al que posteriormente se le calcula su centro de masas, 'centroide', por lo que este valor siempre oscila entre 0 y 100 pero nunca será un valor

extremo, pues como se ha explicado anteriormente, el centro de masas nunca podrá quedar fuera de la figura ni en uno de los extremos.

Para visualizar la explicación anterior, vamos a tomar como ejemplo la Application 29 en la que, como observamos, la agregación se corresponde únicamente con un riesgo alto, por lo que esto sería el mayor riesgo que podríamos observar. No obstante, al realizar el cálculo del centroide el resultado de este riesgo máximo será en torno a 80, que se corresponde con el centro de masas de la figura observada:




```
0029 78.66250000000007
```

Q3: Escalabilidad sin inversión en hardware

Para lograr una mayor escalabilidad, vamos a basarnos en que no podemos realizar inversión en hardware y que el código está lo más optimizado posible. Por lo que debemos evaluar los problemas que presenta Mandami, y qué técnicas podríamos implementar para que sea más eficientes.

Problemas que presenta el método Mandani

1. Alta Complejidad Computacional: El método de Mamdani implica varias operaciones computacionalmente intensivas: las operaciones de fuzzificación, evaluación de reglas, agregación y defuzificación requieren un alto número de operaciones que pueden resultar en un programa poco eficiente.
2. Alto Uso de Memoria: Las funciones de membresía y los conjuntos de salida intermedios deben almacenarse en memoria, lo que puede ser considerable, especialmente con un gran número de reglas y un universo de salida detallado.

Métodos Alternativos para Reducir Tiempos de Ejecución

El modelo de inferencia de Sugeno es generalmente más eficiente que el de Mamdani, principalmente porque simplifica las operaciones de agregación y defuzzificación.

- Consecuentes Constantes o Lineales: En Sugeno, los consecuentes de las reglas pueden ser constantes o funciones lineales de las entradas, en lugar de conjuntos borrosos. Esto elimina la necesidad de defuzzificar un conjunto borroso, reemplazándolo por un cálculo directo y simple del promedio ponderado de los consecuentes de las reglas activadas.
- Evaluación Rápida: Dado que el resultado de cada regla es una función matemática simple de las entradas, el modelo de Sugeno puede ser evaluado más rápidamente que el modelo de Mamdani, especialmente cuando se utilizan funciones lineales o constantes.

Implementación de Tablas de Búsqueda

Otra técnica para mejorar la eficiencia, compatible tanto con Mamdani como con Sugeno, es la precomputación de resultados. Esto es particularmente efectivo cuando las entradas y las reglas no varían dinámicamente:

- Precomputar Resultados: Calcula los resultados de todas las combinaciones posibles de entradas y almacenarlos en una tabla de búsqueda. Durante la ejecución real, simplemente consulta esta tabla para obtener el resultado, eliminando la necesidad de ejecutar el proceso de inferencia.
- Ventajas: Reduce significativamente el tiempo de cálculo en tiempo de ejecución a casi cero, ya que la respuesta se recupera de una tabla en lugar de ser calculada.
- Desventajas: El espacio de almacenamiento necesario puede ser considerable, dependiendo de la granularidad y rango de las entradas.