Universidad Carlos III
Grado en ingeniería informática
Curso Inteligencia Artificial 2023-24
Práctica final (Informe)
Curso 2023-24

Fecha: **15/05/2024** - ENTREGA: **FINAL**

GRUPO: **81**

Alumnos: **Mario Ramos Salsón (100495849), Miguel Fidalgo García (100495770) y Víctor Martínez de las Heras (100495829)**

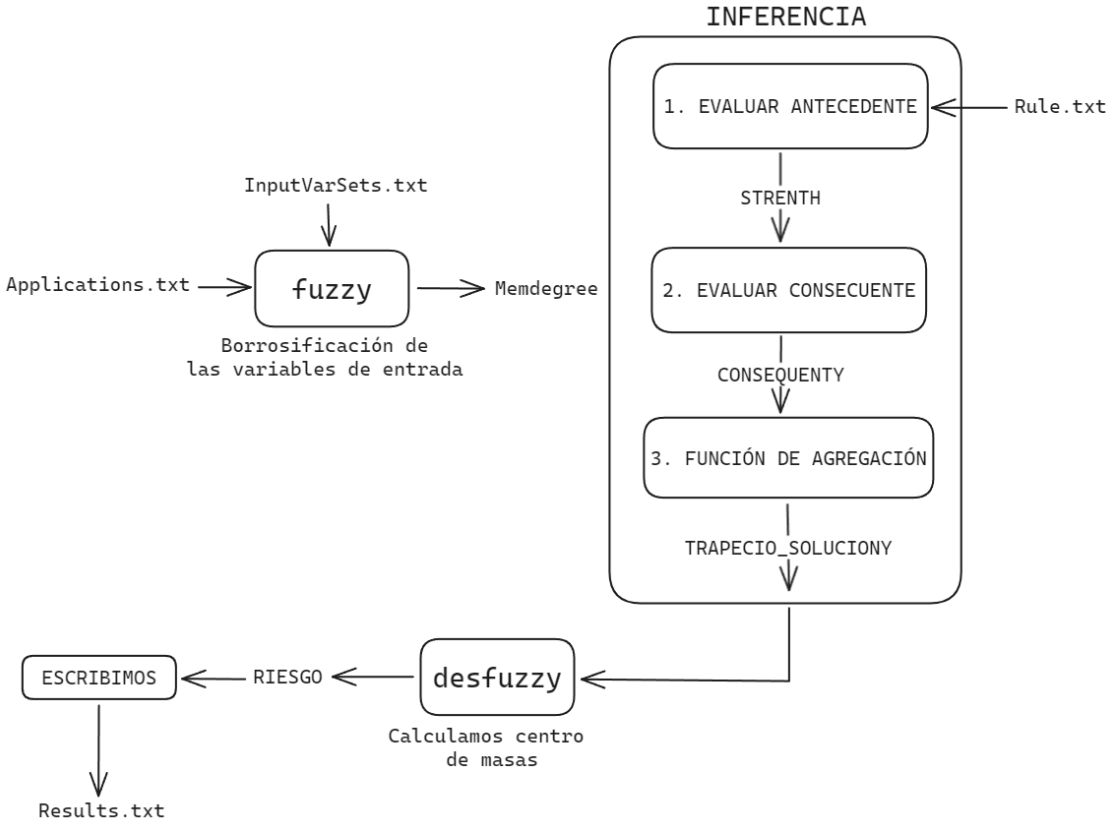
ÍNDICE

1. INTRODUCCIÓN

Desde la empresa C3L hemos realizado un programa informático para cubrir las necesidades del Banco Pichin. Esto les ayudará a calcular los riesgos de cada cliente de forma personalizada para así determinar si les otorgan un préstamo o no en base a los criterios proporcionados.

2. RESUMEN EJECUTIVO

El programa informático que hemos desarrollado tiene como objetivo principal el cálculo del riesgo a partir de los datos clave proporcionados para cada cliente: edad, nivel de ingresos, bienes de posesión, cantidad prestada en relación a los ingresos, estabilidad del empleo y historial de préstamos y pagos. A partir de estos datos el sistema utiliza el sistema de inferencia borrosa de Mandani que contiene la estructura que se muestra en el diagrama a continuación:



Aquí podemos ver que las entradas se le pasan al sistema en el archivo de Applications.txt a partir del cual se realiza la borrosificación de estas variables en conjuntos borrosos. Una vez actualizados los grados de pertenencia para las variables leemos las reglas del archivo correspondiente, que contiene la información lógica sobre cómo influyen cada uno de las informaciones proporcionadas sobre la decisión final de si conceder el préstamo.

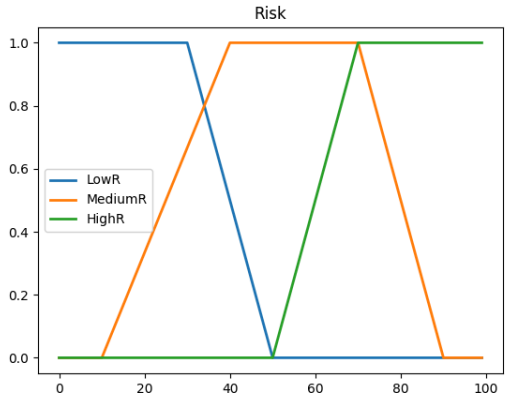
Por último, una vez tenemos la gráfica unida con el resultado de todas estas reglas (en forma de vector) calculamos mediante una función matemática el centro de masas para poder calcular de forma precisa el riesgo que hay para cada solicitud de forma personalizada y poder así facilitar la decisión final al banco.

3. DESCRIPCIÓN DEL SISTEMAS DE INFERENCIA

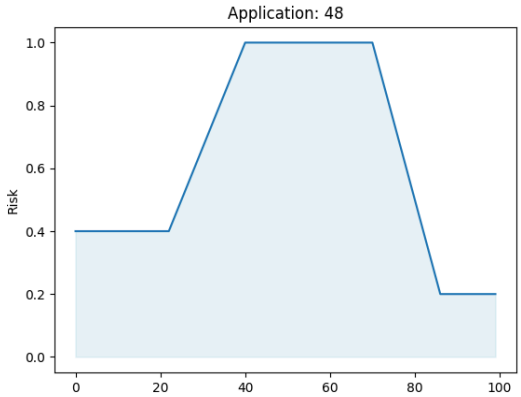
Nuestro sistema de inferencia recibe como parámetro los datos de un cliente que quiere pedir un préstamo. Dependiendo de su edad, salario, patrimonio, cantidad prestada, estabilidad de empleo e historial de pagos, indicaremos cómo de arriesgado es darle dicho préstamo.

Para ello, varios expertos del banco nos han proporcionado las reglas que tiene que seguir el sistema de inferencia. Para cada cliente, evaluamos cada una de las reglas siguiendo los conjuntos borrosos proporcionados.

Con la ayuda de estos conjuntos borrosos evaluamos los antecedentes de las reglas y cogemos el mínimo grado de pertenencia, ya que todas las reglas son de tipo ***'and'***. Este mínimo será la fuerza que tiene el consecuente de la regla, que es la máxima altura que tendrá el trapecio de los riesgos proporcionados.



Sumando los consecuentes de todas las reglas con sus respectivas fuerzas, nos quedará una figura compuesta como por ejemplo la siguiente:



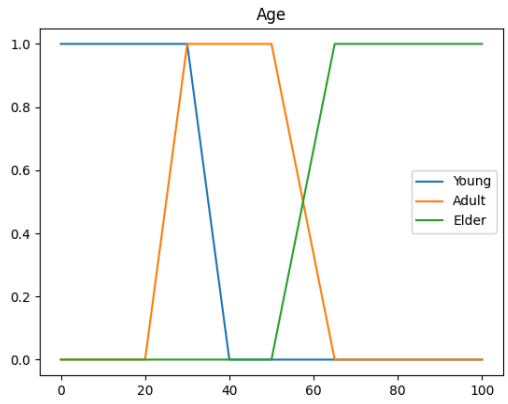
De estas figuras obtendremos el centro de masas. Donde caiga el centro de masas de la figura en el eje **'x'** será el nivel de riesgo que tiene el cliente a la hora de pedir el préstamo. En esta figura el centro de masas estaría en x = 49.15

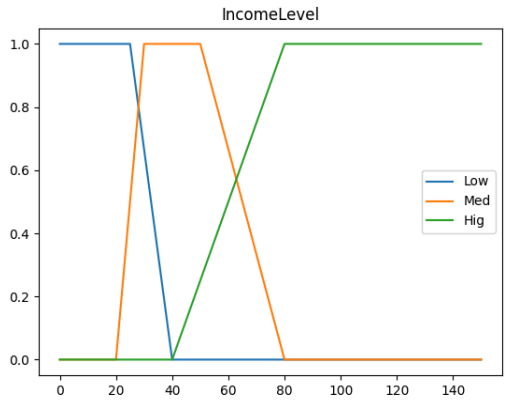
4. METODOLOGÍA

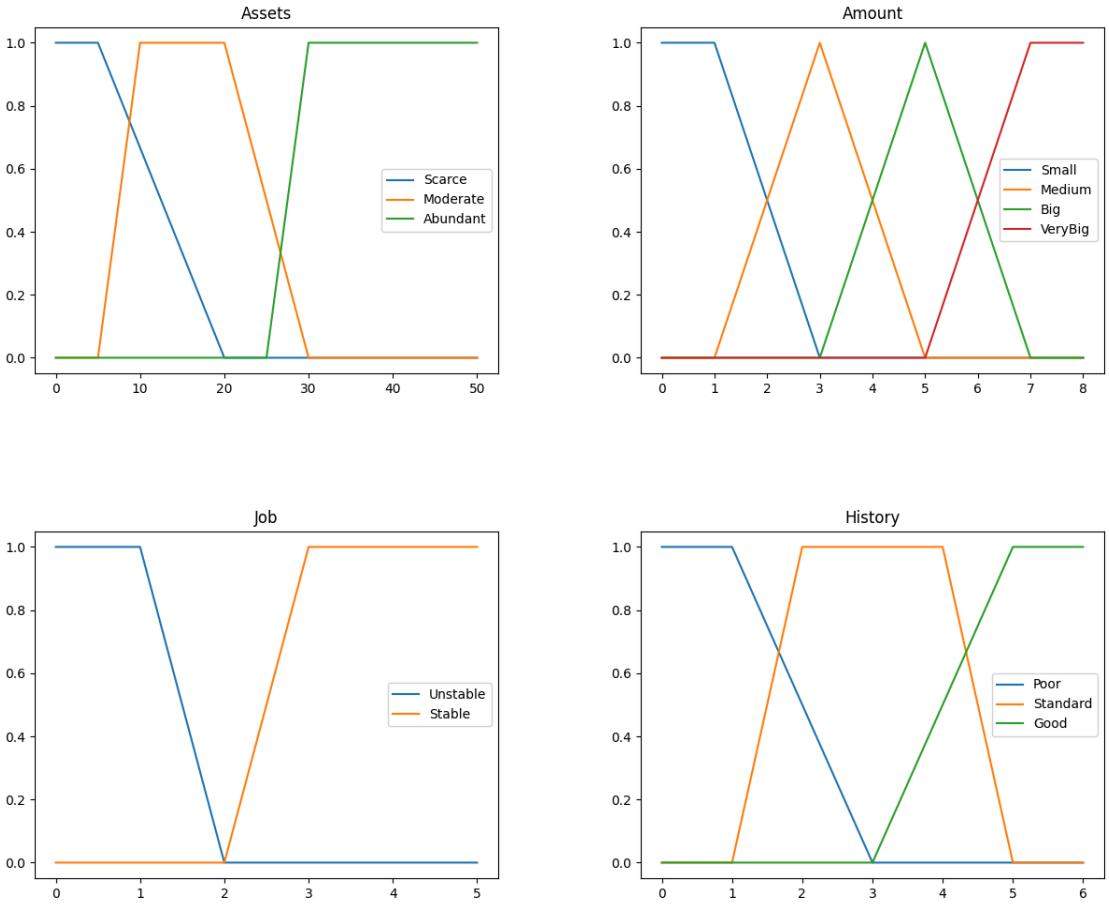
A continuación, vamos a especificar y desarrollar las fases del proyecto definidas en el diagrama del apartado 2. Cabe destacar que en la implementación y para mantener la atomicidad y asegurar la comprensión del código hemos pensado que la mejor manera era realizar cada lógica en una función distinta, es decir, cada rectángulo que realiza una lógica distinta irá en una función independiente.

4.1. Borrosificación

En el apartado de borrosificación nos encargamos de analizar la información recibida de cada una de las solicitudes (***Applications.txt***) y a partir de los conjuntos borrosos definidos en ***InputVarSets.txt***, buscamos la altura que tiene en dicha gráfica el valor de la **"x"** dado y eso corresponde con el grado de pertenencia de cada conjunto borroso. Mediante la función de python ***matplotlib*** hemos elaborado las siguientes gráficas sobre cada uno de los conjuntos borrosos para facilitar la comprensión de estos y entender cómo están definidos.







4.2. Reglas de inferencia

En la inferencia realizamos una serie de pasos a partir del archivo ***Rules.txt***. Este contiene las reglas del sistema, por lo que realizaremos las siguientes acciones para cada una de ellas:

EVALUACIÓN DE ANTECEDENTES:

En este apartado se evalúa el antecedente añadiendo el grado de pertenencia de todos los antecedentes a una lista y seleccionando el mínimo

de esta con el que se actualiza el strength de cada regla (grado de pertenencia al conjunto borroso riesgo).

EVALUACIÓN DE CONSECUENTES:

En este apartado se evalúa el consecuente actualizando los valores de sus ejes ***"x"*** e ***"y"*** que tiene cada regla. Para ello recurrimos al método de ***clipping*** que consiste en realizar un corte a la función de pertenencia del consecuente con la similitud del antecedente, es decir, todos aquellos valores de la "y" que sean mayor al strength tomarán el valor del strength.

FUNCIÓN DE AGREGACIÓN:

En este apartado se realiza el último paso de la inferencia que consiste en la unión de los resultados de las reglas, trazando el trapecio solución. Para ello, se analiza cada consecuente de cada regla y se compara constantemente con el valor que hay en el trapecio solución, inicialmente 0. En caso de que haya una regla con un valor mayor que el que tenía el trapecio, se reemplazará, para cada ***"x"***, el valor del trapecio por el consecuente de la regla. De esta forma logramos que el trapecio solución sea la unión ***"or"*** de los resultados de todas las reglas.

4.3. Desborrosificación

El proceso de desborrosificación consiste en convertir los conjuntos borrosos mencionados anteriormente en un valor numérico preciso que se pueden utilizar en la toma de decisiones del banco Pichin. Para ello utilizamos el método Centroid of Area (COA) con el que calculamos el centroide del área bajo la curva de la función del grado de pertenencia del conjunto borroso. Este es el punto donde se considera concentrada el área total de la figura, también conocido como **centro de masas**.

5. PRESUPUESTO

Para la evaluación del capital usado en el proyecto, tenemos que tener en cuenta a todo el personal que se ha visto involucrado en el desarrollo del mismo, junto con los recursos utilizados. Entre este personal, encontramos a 20 trabajadores, de los cuales 15 de ellos son programadores juniors, 4 son managers y uno es un socio.

Los salarios varían dependiendo del trabajador. En el caso de los juniors, cobran 50 euros la hora y han trabajado un total de 100 horas cada uno. Esto eleva el total de dinero gastado a 75.000 euros solo en su salario. Los managers cobran 100 euros la hora y han trabajado 50 horas cada uno. Esto añade un total de 20.000 euros más al total gastado. Para finalizar, tenemos que tener en cuenta al socio. Este cobra 150 euros por hora y ha trabajado 25 horas en total. Por ello el dinero total gastado solo en los salarios de los 20 trabajadores asciende a 98.750 euros.

También, hay que tener en cuenta que hay que generar un beneficio del proyecto y no solo amortizar los gastos. Por ello hemos decidido multiplicar por 1,5 el dinero total gastado. Esto permite a nuestra empresa crecer tanto a nivel económico como a nivel de calidad para los trabajadores. Ya que con estos beneficios, podemos mejorar el material, la infraestructura y el software que utilizamos para el desarrollo de los proyectos entre otras cosas.

Para finalizar, tenemos que tener en cuenta el 21% de impuestos que hay que pagar al estado. Esto asciende el total del presupuesto a 181.500 euros, pero lo aproximamos a 180.000 para dar un número más exacto.

6. ANEXO

6.1. Código (main.py)

Python

```
from MFIS_Read_Functions import *


def process_app(fuzzy_file, rules_file, values):

    fuzzy(fuzzy_file, values)

    # Inicializamos el trapecio solución a 0 tanto su x como su y
    trapecio_solucion_x = np.arange(0, 101, 1)
    trapecio_solucion_y = np.zeros(101)

     # Hallar el resultado para cada regla y su valor strength (min de
antecedentes)
    for rule in rules_file:

        # Actualizamos el antecedente
        eval_antecedente(fuzzy_file, rule)

             # Lo cargamos de nuevo para cada regla y evaluamos el
consecuente
        risks_file = read_fuzzy_sets_file('Files/Risks.txt')
        eval_consecuente(risks_file, rule)

        # Unión de todos los resultados de las reglas
        agregacion(rule, trapecio_solucion_y)

    # Cálculo del centro de masas y lo devolvemos
    return skf.centroid(trapecio_solucion_x, trapecio_solucion_y)


def fuzzy(fuzzy_file, values):
     """La función fuzzy realiza la borrosificación, es decir, actualiza
el
    grado de pertenencia para cada conjunto borroso (memDegree)"""
```

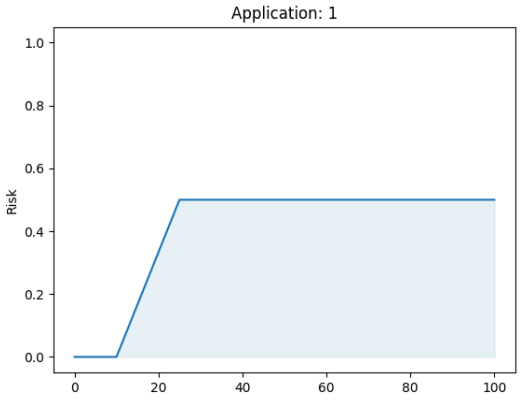
```
    age, income_level, assets, amount, job, history = \
        values[0], values[1], values[2], values[3], values[4], values[5]
    # Borrosificación de los antecedentes según valores de InputVarSets
    for fuzzy_item in fuzzy_file.items():
        if fuzzy_item[1].var == 'Age':
            y_value = fuzzy_item[1].y[age]
            fuzzy_item[1].memDegree = y_value
        elif fuzzy_item[1].var == 'IncomeLevel':
            y_value = fuzzy_item[1].y[income_level]
            fuzzy_item[1].memDegree = y_value
        elif fuzzy_item[1].var == 'Assets':
            y_value = fuzzy_item[1].y[assets]
            fuzzy_item[1].memDegree = y_value
        elif fuzzy_item[1].var == 'Amount':
            y_value = fuzzy_item[1].y[amount]
            fuzzy_item[1].memDegree = y_value
        elif fuzzy_item[1].var == 'Job':
            y_value = fuzzy_item[1].y[job]
            fuzzy_item[1].memDegree = y_value
        elif fuzzy_item[1].var == 'History':
            y_value = fuzzy_item[1].y[history]
            fuzzy_item[1].memDegree = y_value


def eval_antecedente(fuzzy_file, rule):
       """Esta función evalúa el antecedente añadiendo el grado de
pertenencia de
      todos los antecedentes y seleccionando el mínimo con el que se
actualiza el
     strength de cada regla (grado de pertenencia del conjunto borroso
riesgo)"""
    memdegree_antecedents = []
    for antecedente in rule.antecedent:
        for fuzzy_item in fuzzy_file.items():
            if antecedente == fuzzy_item[0]:
                memdegree_antecedents.append(fuzzy_item[1].memDegree)
                break
    rule.strength = min(memdegree_antecedents)
```

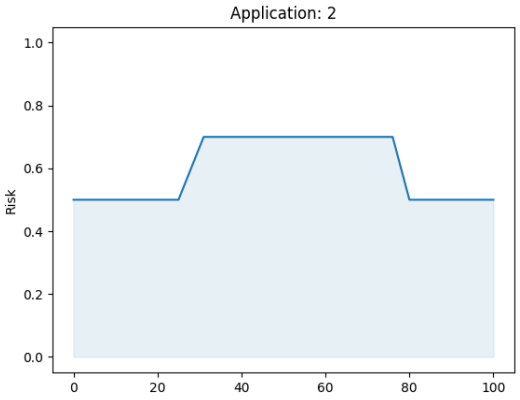
```
def eval_consecuente(risks_file, rule):
     """Esta función evalúa el consecuente rellenando la lista de los
valores de
    "y" que toma el consecuente para cada regla"""
    # Borrosificación del consecuente según los valores de Risks.txt
    for risk in risks_file.items():
        if rule.consequent == risk[0]:
            rule.consequentX = risk[1].x
            for i in range(len(risk[1].y)):
                if risk[1].y[i] > rule.strength:
                    risk[1].y[i] = rule.strength
            rule.consequentY = risk[1].y


def agregacion(rule, trapecio_solucion_y):
       """Esta función de agregación realiza el último paso de la
inferencia que
     consiste en la unión de los resultados de las reglas, trazando el
trapecio solución"""
    for i in range(len(trapecio_solucion_y)):
        if rule.consequentY[i] > trapecio_solucion_y[i]:
            trapecio_solucion_y[i] = rule.consequentY[i]


# ---------------Applications iteration---------------

def main():
    app_file = read_applications_file()
    fuzzy_file = read_fuzzy_sets_file('Files/InputVarSets.txt')
    rules_file = read_rules_file()
    outputfile = open('Results.txt', 'w')

    values = []
    for app in app_file:
        for i in range(len(app.data)):
            values.append(app.data[i][1])
        riesgo = process_app(fuzzy_file, rules_file, values)
        values = []

        # Limitamos riesgo a dos decimales
```

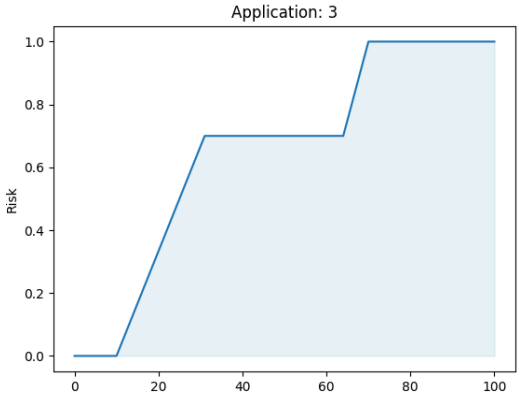
```
        riesgo = "{:.3f}".format(riesgo)
        riesgo = str(riesgo)

        # Abre el archivo en modo escritura ('w')
         outputfile.write("AppId: " + app.appId + ", Risk: " + riesgo +
"\n")

    outputfile.close()


# --------------------------------------------------------

if __name__ == '__main__':
    main()
```

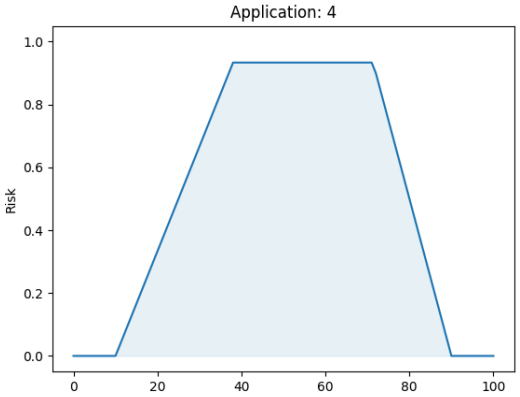
Cabe destacar que en el transcurso del desarrollo de la práctica, nos encontramos un error y por eso hemos tenido que entregar todo el código. Este problema estaba dentro de las funciones otorgadas para la lectura de ficheros, ya que la función ***"read_fuzzy_sets_file"*** no leía bien la información y nos devolvía un array más corto del que debería. Por ejemplo si el valor máximo de "Amount" era 8, el array correspondiente debería ser del 0 al 8 ambos incluidos. Pero debido a este error, se nos devolvía un array del 0 al 7. Por ello hemos modificamos ligeramente la función y ya todo funciona correctamente.

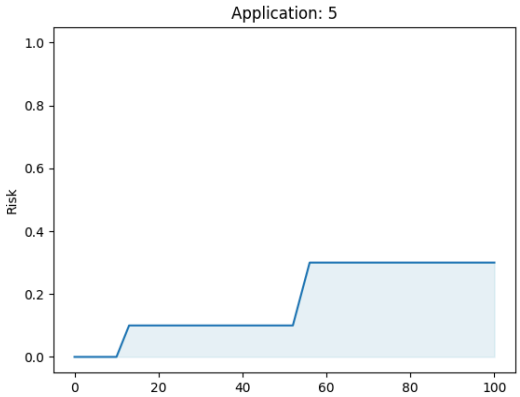
6.2. Gráficas desborrosificación

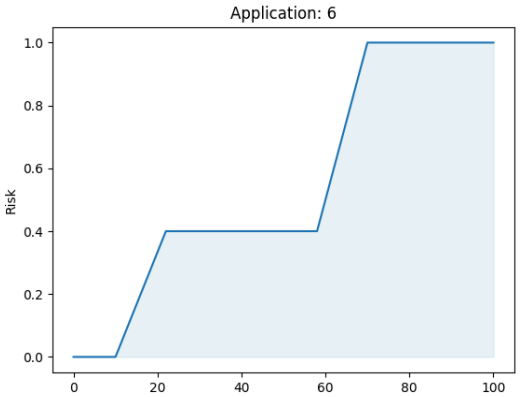


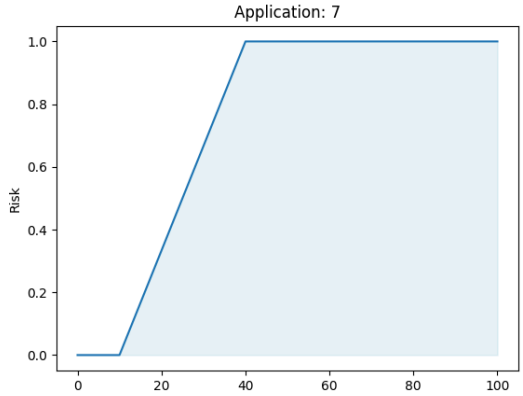


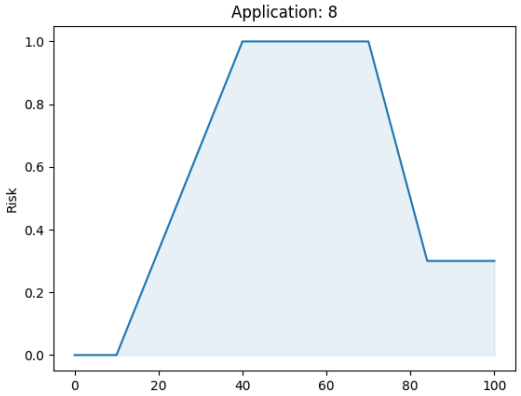


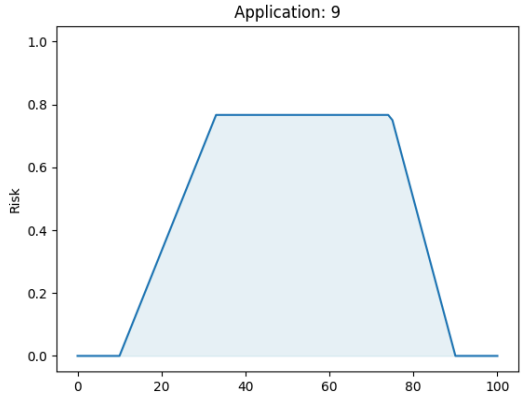


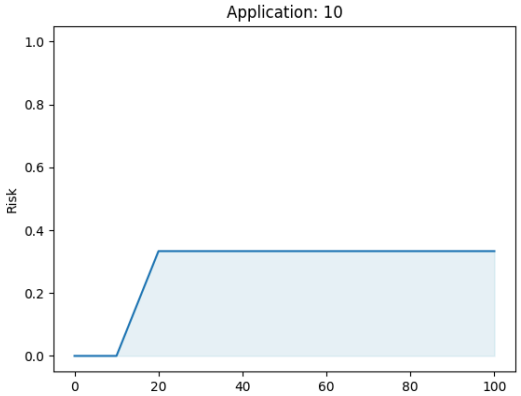


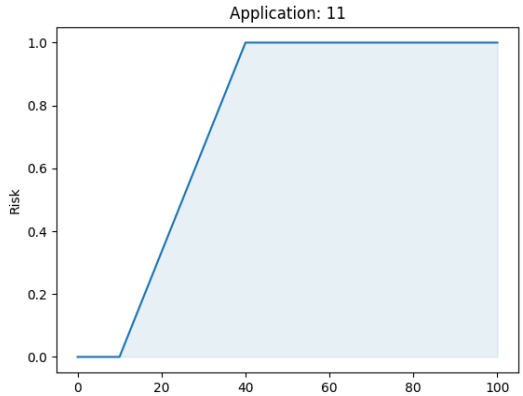


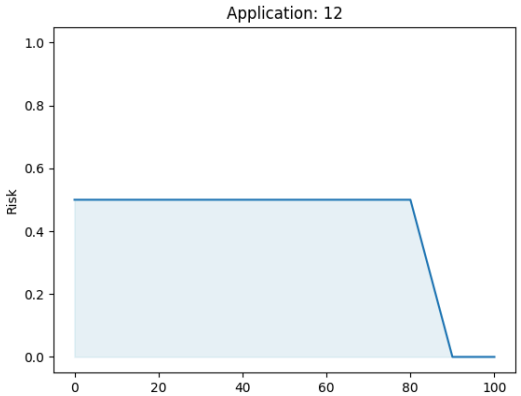

Application: 7
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

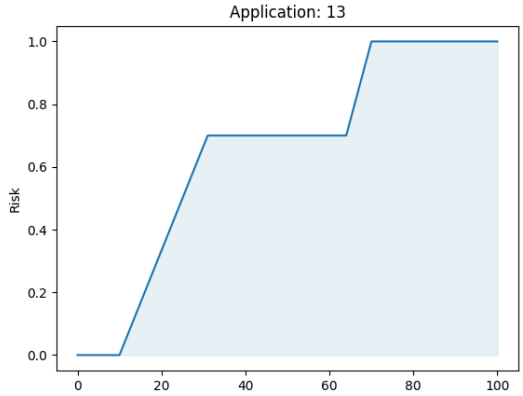

Application: 8
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

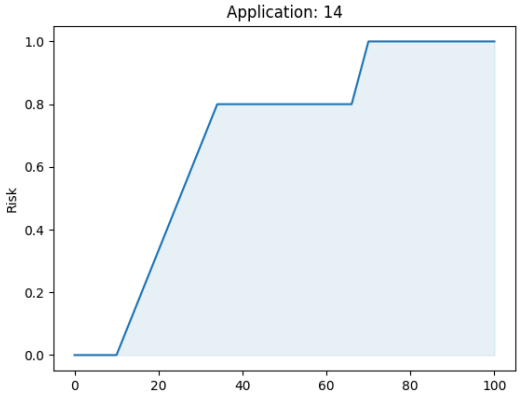

Application: 9
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

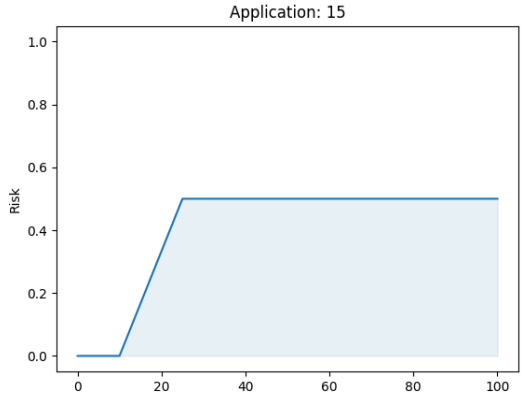

Application: 10
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

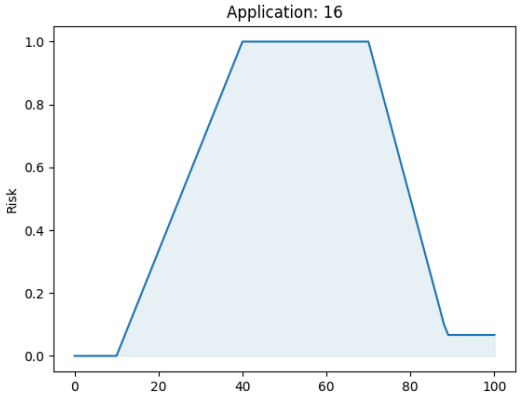

Application: 11
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

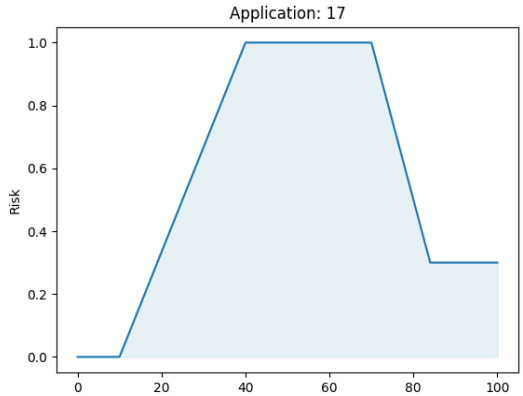

Application: 12
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

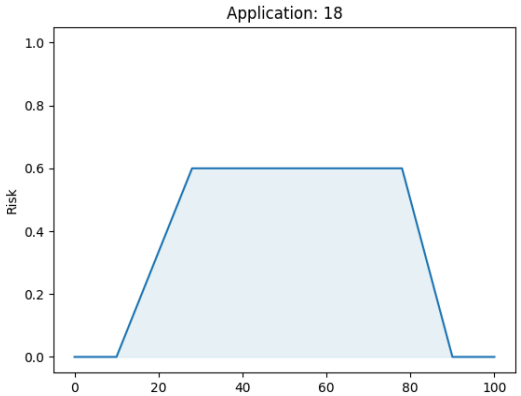

Application: 13
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

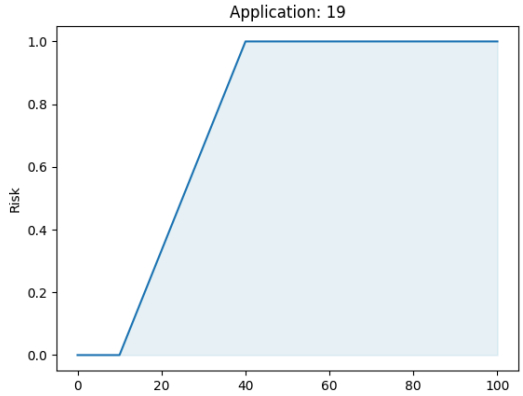

Application: 14
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

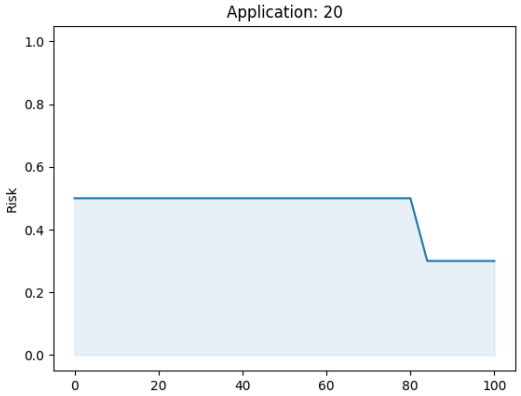

Application: 15
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

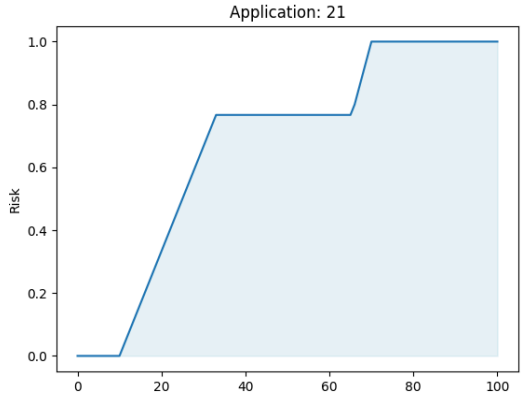

Application: 16
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

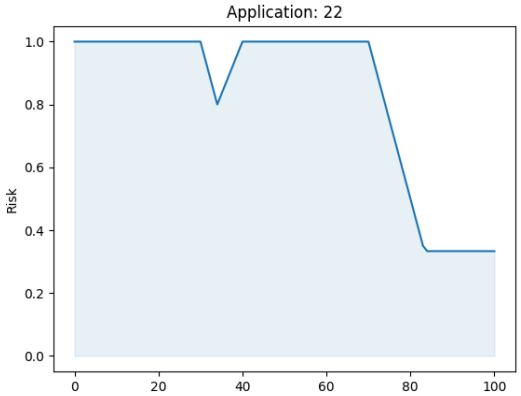

Application: 17
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

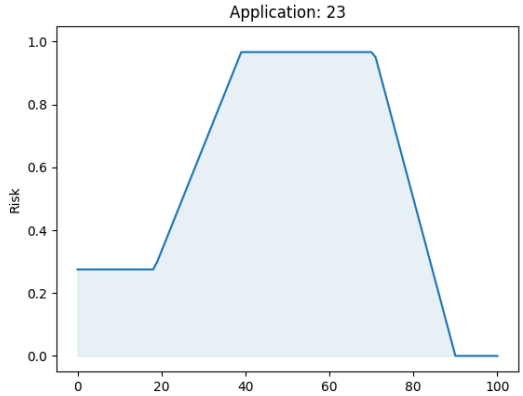

Application: 18
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

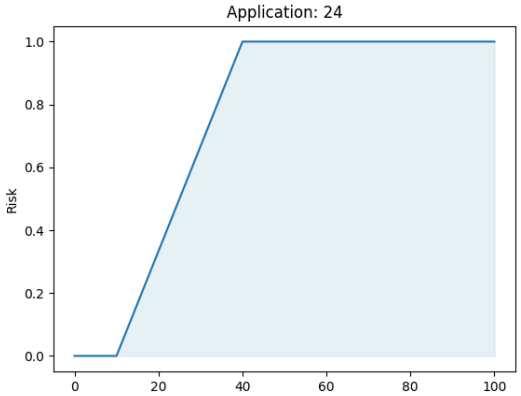

Application: 19
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

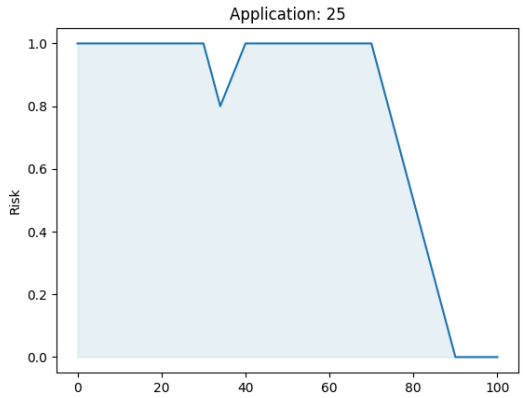

Application: 20
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

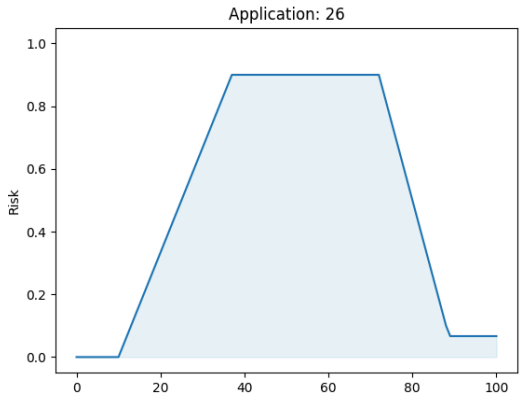

Application: 21
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

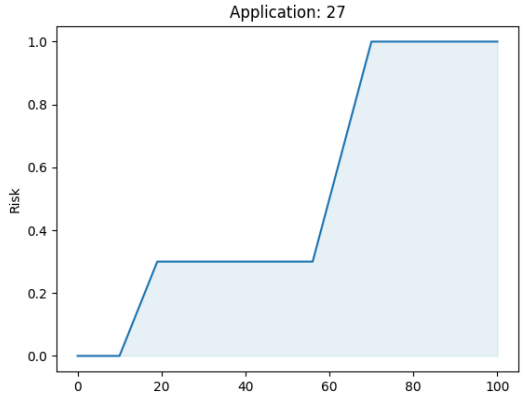

Application: 22
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

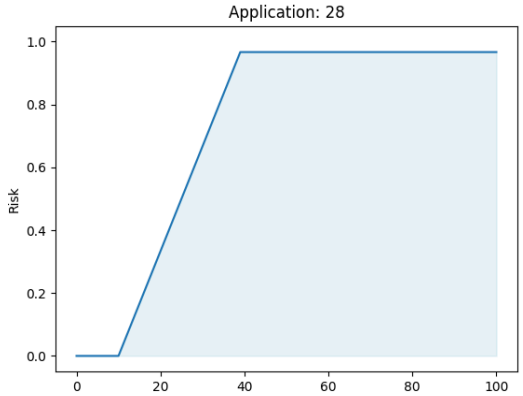

Application: 23
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

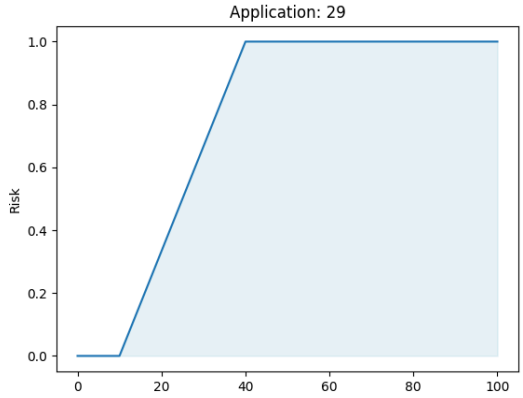

Application: 24
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

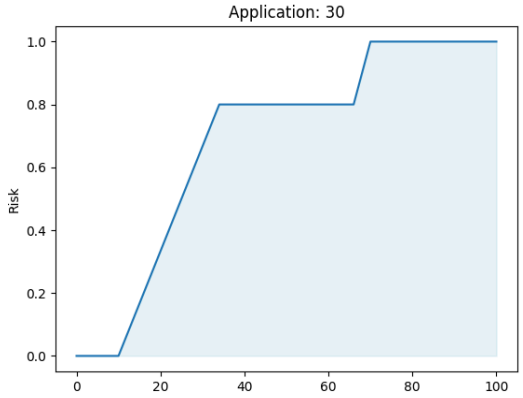

Application: 25
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

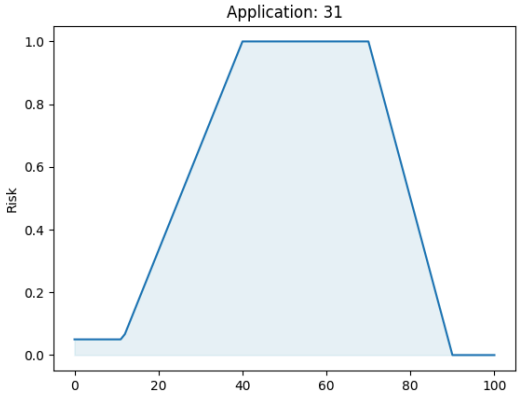

Application: 26
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

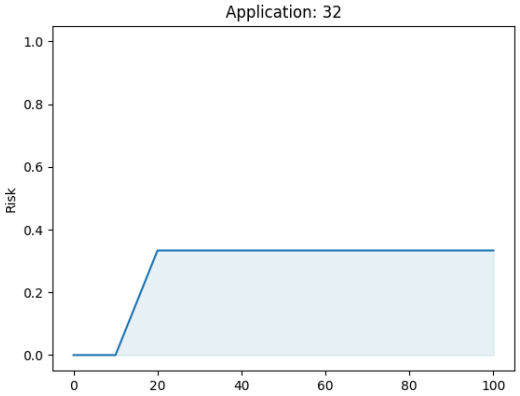

Application: 27
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

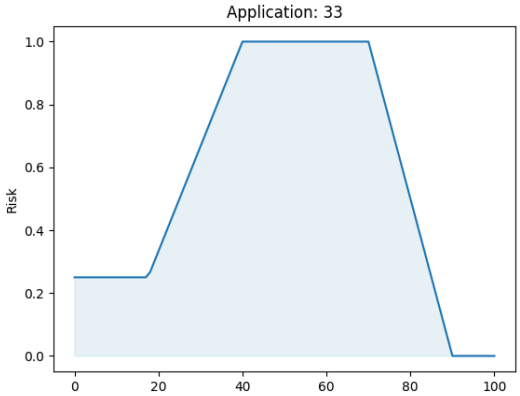

Application: 28
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

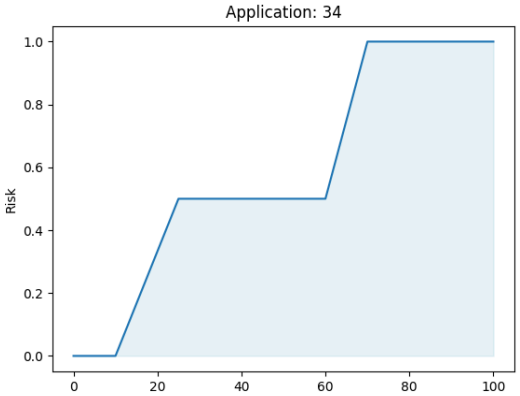

Application: 29
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

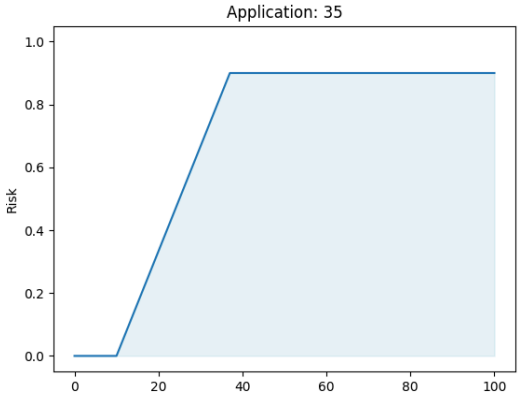

Application: 30
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

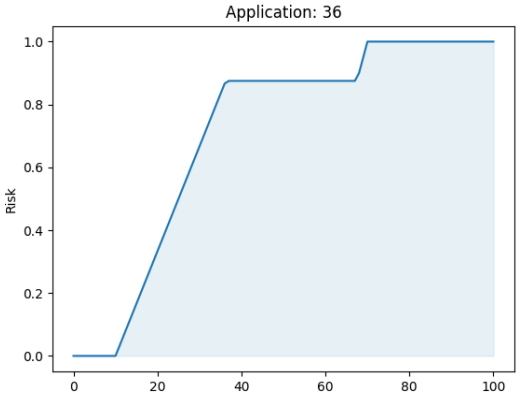

Application: 31
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

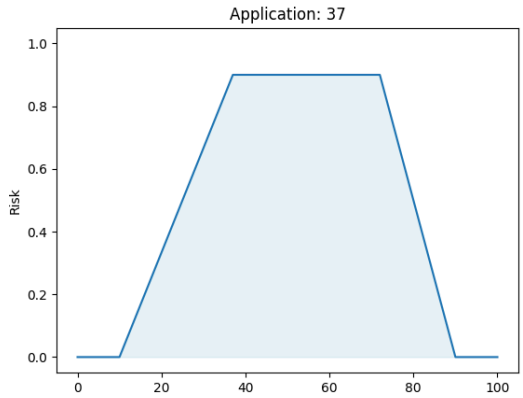

Application: 32
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

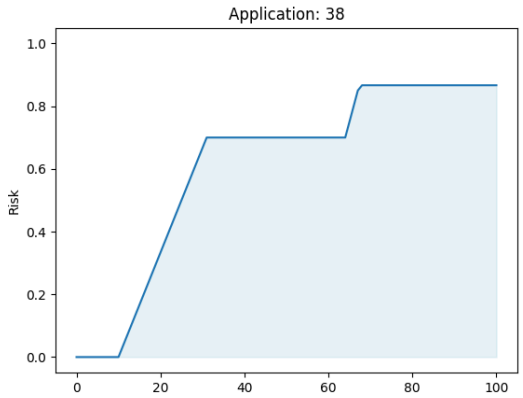

Application: 33
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

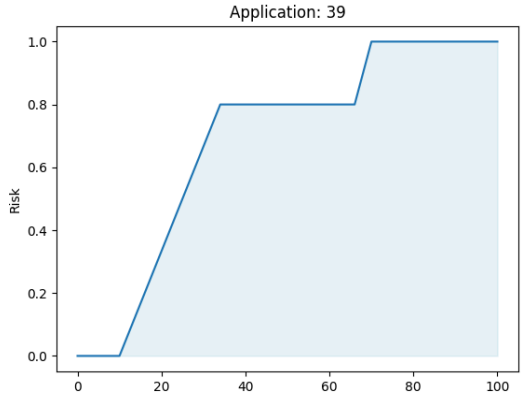

Application: 34
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

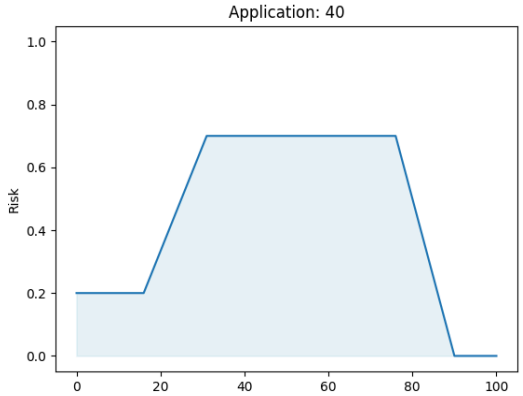

Application: 35
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

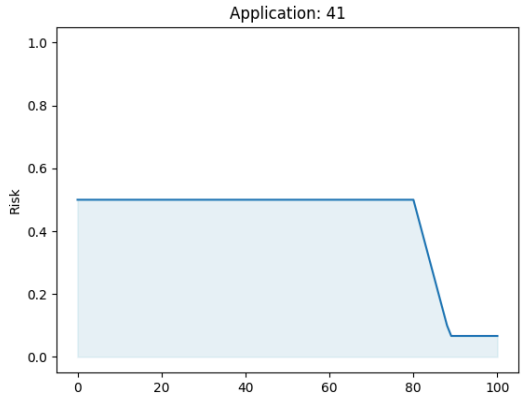

Application: 36
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

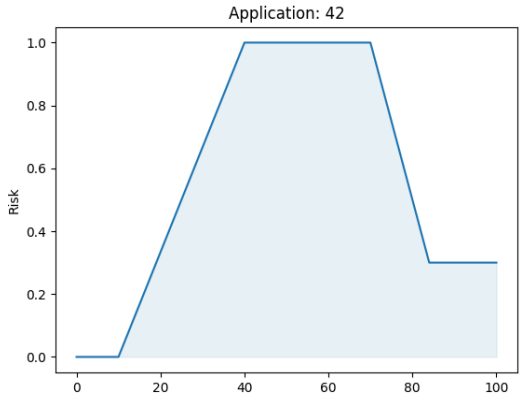

Application: 37
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

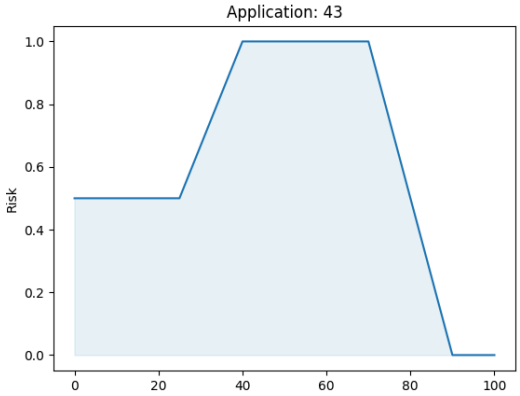

Application: 38
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

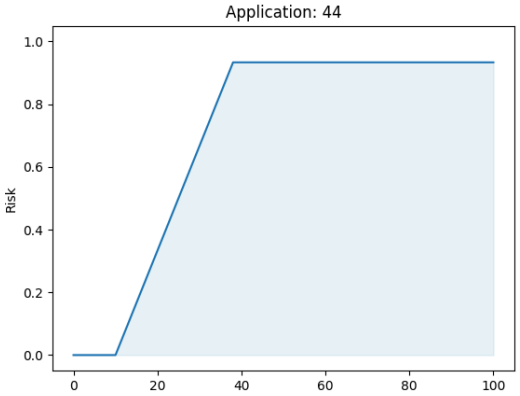

Application: 39
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

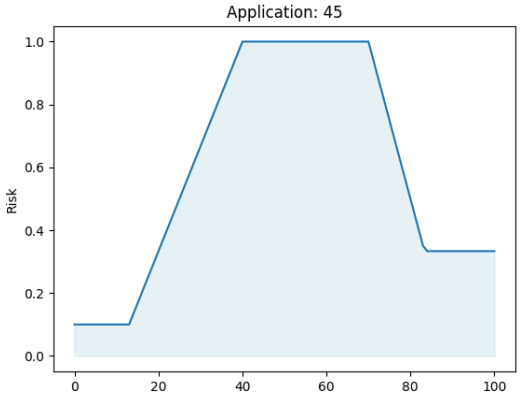

Application: 40
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

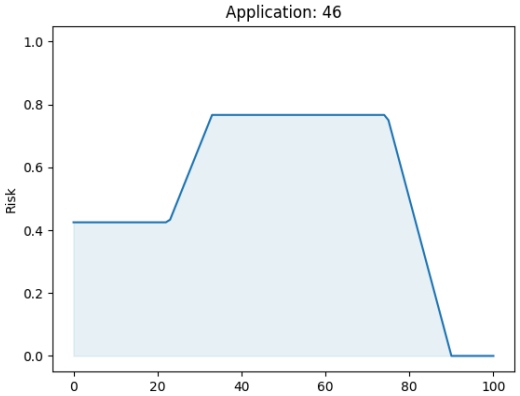

Application: 41
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

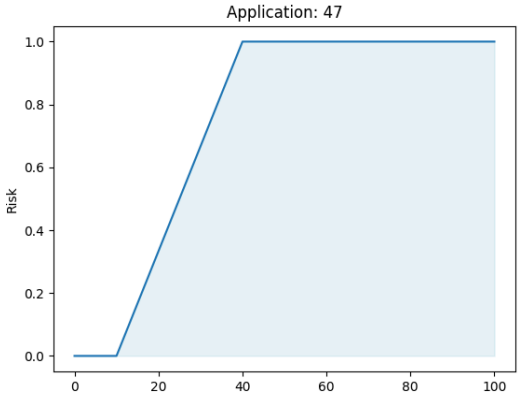

Application: 42
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

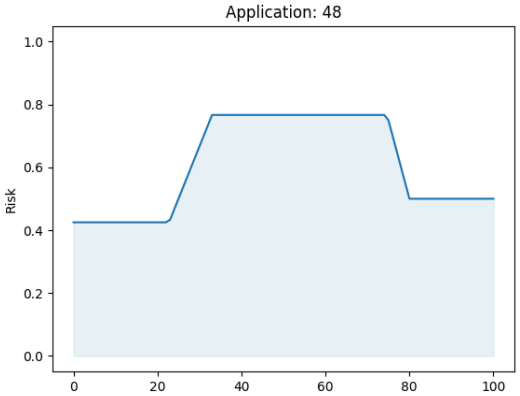

Application: 43
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

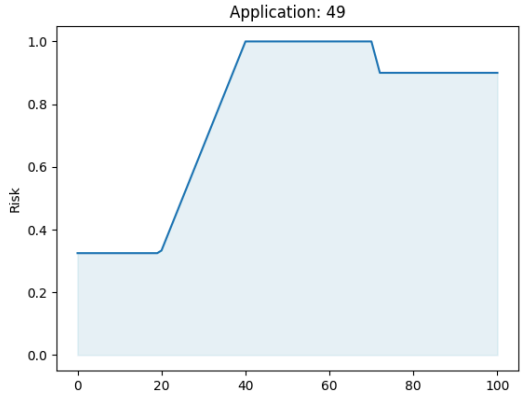

Application: 44
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

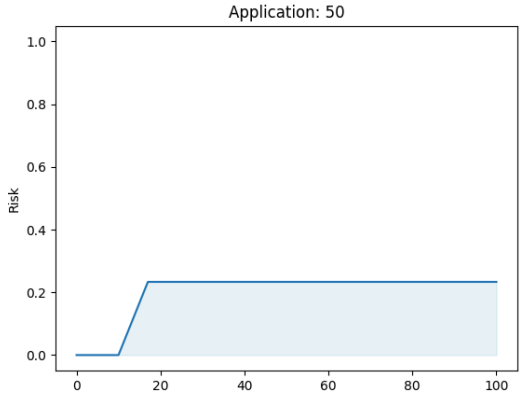

Application: 45
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Application: 46
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Application: 47
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Application: 48
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Application: 49
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100


Application: 50
Risk
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100