Prise de décisions multicritères appliquée à l'octroi des crédits par les banques

Aymen Sarhane

Épreuve de TIPE

Session 2023

Introduction

Figure – Crédit bancaire

Plan de l'exposé

La prise de décisions multicritères

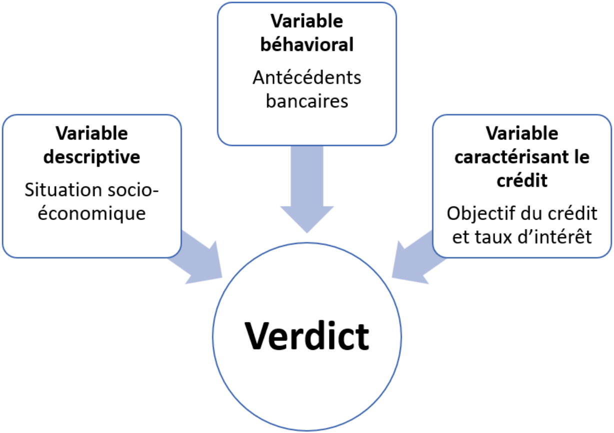

Figure – Les critères de la prise de décisions

La logique floue

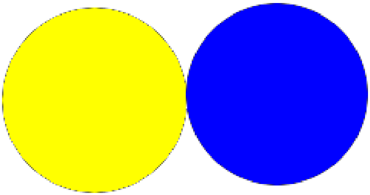

Figure – Logique classique

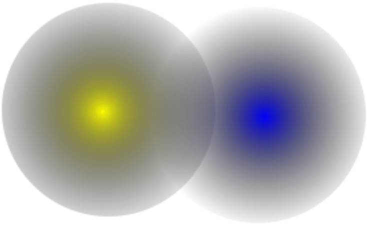

Figure – Logique floue

La logique floue

Emploi de la logique floue dans la prise de décisions

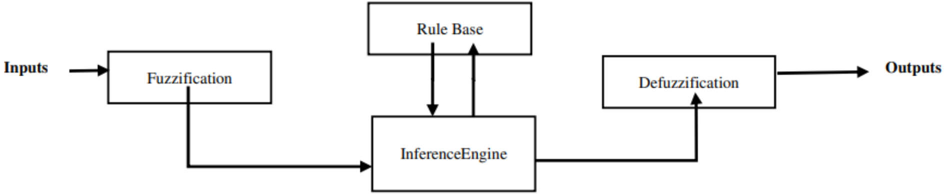

Figure – Étapes de l'application de la logique floue

Fuzzification

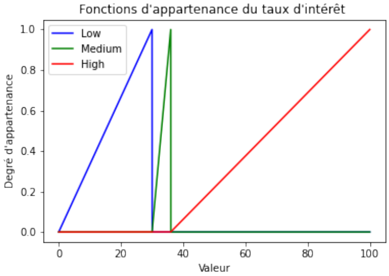

Figure – Fonctions d'appartenance du taux d'intérêt

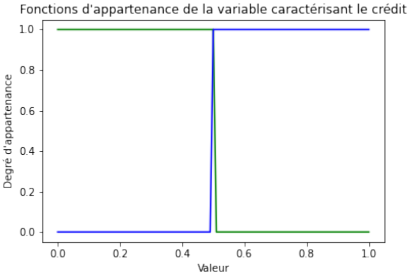

Figure – Fonctions d'appartenance de la variable caractérisant le crédit

La prise de décisions multicritères

Définition des variables floues et les fonctions d'appartenance associées

```python
# Création de la variable floue
interest_rate = ctrl.Antecedent(universe_interest_rate, 'Interest rate')

# Définition des fonctions d'appartenance
interest_rate['Low'] = fuzz.trimf(universe_interest_rate, [0, 20, 20])
interest_rate['Medium'] = fuzz.trimf(universe_interest_rate, [20, 36, 36])
interest_rate['High'] = fuzz.trimf(universe_interest_rate, [36, 100, 100])
```

Figure – Définition de la variable floue "taux d'intérêt" et des fonctions d'appartenance associées

Moteur d'inférence

- Élaboration des règles d'inférence transcrites comme suit à partir d'une étude statistique et des avis des experts :

```
rule1 = ctrl.Rule(purpose_of_loan['Creation'] & interest_rate_low['Low'], loan_variable['Low'])
rule2 = ctrl.Rule(purpose_of_loan['Creation'] & interest_rate_low['Medium'], loan_variable['Low'])
rule3 = ctrl.Rule(purpose_of_loan['Creation'] & interest_rate_low['High'], loan_variable['Medium'])
rule4 = ctrl.Rule(purpose_of_loan['Development'] & interest_rate_low['Low'], loan_variable['Low'])
rule5 = ctrl.Rule(purpose_of_loan['Development'] & interest_rate_low['Medium'], loan_variable['Medium'])
rule6 = ctrl.Rule(purpose_of_loan['Development'] & interest_rate_low['High'], loan_variable['High'])
```

Figure – Les règles d'inférence

Défuzzification

```python
# Mise en oeuvre du système de contrôle
risk_simulation.input['Purpose of loan'] = 0.9
risk_simulation.input['Interest rate Low'] = 21
risk_simulation.compute()

# Retrieve the result
result = risk_simulation.output['loan_variable']
print("Loan Variable:", result)
# Return the linguistic term with the maximum degree of membership
print("Linguistic Result:", linguistic_result)
  return go(f, seed, [])
}
```

Figure – Code de la défuzzification

Validation du modèle

Premier exemple

```
risk_simulation. input[ 'Sexe'] = 1 # masculin
risk_simulation.input[ 'Education'] = 3 # études moyennes
risk_simulation. input['Activite'] = 0.1 # commerce
risk_simulation. input['Assurance'] = 1 # non assuré
risk_simulation. input[ 'Statut marital'] = 1 # célibataire
# Évaluer le système de contrôle
risk_simulation.compute()
#Récupérer le résultat
result = risk_simulation.output['Descriptive Variable']
print("Descriptive Variable:", result)
#Renvoyer le terme Linguistique avec le degré d'appartenance maximal
print("Linguistic Result:", linguistic_result)

---------------------------------------------------------------------------------
Out[]:

Descriptive Variable: 14.999999999999996
Linguistic Result: High
```

Figure – Résultat obtenu à partir du premier cas "non assuré"

Validation du modèle

Premier exemple

```
risk_simulation. input[ 'Sexe'] = 1 # masculin
risk_simulation.input[ 'Education'] = 3 # études moyennes
risk_simulation. input['Activite'] = 0.1 # commerce
risk_simulation. input['Assurance'] = 0 # assuré
risk_simulation. input[ 'Statut marital'] = 1 # célibataire
# Évaluer le système de contrôle
risk_simulation.compute()
#Récupérer le résultat
result = risk_simulation.output['Descriptive Variable']
print("Descriptive Variable:", result)
#Renvoyer le terme Linguistique avec le degré d'appartenance maximal
print("Linguistic Result:", linguistic_result)

------------------------------------------------------------------------------
Out[]:

Descriptive Variable: 5.0
Linguistic Result: Low
```

Figure – Résultat obtenu à partir du deuxième cas "assuré"

Validation du modèle

Deuxième exemple

```
risk_simulation.input['Purpose of loan'] = 1 #destiné au développement
risk_simulation.input['Interest rate Low'] = 21 # taux d'intérêt moyen
# Évaluer le système de contrôle
risk_simulation.compute()
# Récupérer le résultat
result = risk_simulation.output ['loan_variable']
print("Loan Variable:", result)
# Renvoyer le terme Linguistique avec le degré d'appartenance maximal
print("Linguistic Result:", linguistic_result)

-------------------------------------------------------------------------------------
Out[]:

Loan Variable: 7.015542328042326
Linguistic Result: Medium
```

Figure – Résultat obtenu à partir du premier cas "taux d'intérêt faible"

Validation du modèle

Deuxième exemple

```
risk_simulation.input['Purpose of loan'] = 1 #destiné au développement
risk_simulation.input['Interest rate Low'] = 4 #taux d'intérêt faible
# Évaluer le système de contrôle
risk_simulation.compute()

# Récupérer le résultat
result = risk_simulation.output['loan_variable']
print("Loan Variable:", result)
# Renvoyer le terme Linguistique avec le degré d'appartenance maximal
print("Linguistic Result:", linguistic_result)

------------------------------------------------------------------------------------------
Out[]:

Loan Variable: 2.711111111111113
Linguistic Result: Low
```

Figure – Résultat obtenu à partir du deuxième cas "taux d'intérêt élevé"

Validation du modèle

- Dans les deux derniers exemples , le modèle agit comme prévu.
- Vu la précision des règles d'inférence et sa conformité aux résultats attendus, le modèle est à priori valide.

Validation du modèle

- Dans les deux derniers exemples , le modèle agit comme prévu.
- Vu la précision des règles d'inférence et sa conformité aux résultats attendus, le modèle est à priori valide.

Mise en marche du modèle

```
risk_simulation.input['descriptive_variable'] = Descriptive_variable(False)[0]
risk_simulation.input['behavioral_variable'] = Behavioral_variable(False)[0]
risk_simulation.input['loan_variable'] = Loan_variable(False)[0]
# Récupérer le résultat
result = risk_simulation.output['customer_variable']
print("Loan Variable:", result)

-------------------------------------------------------------------------------------

Out[]:
Customer_variable : 12.191834
Linguistic Evaluation : High Risky
```

Figure – Exécution du code

Mise en marche du modèle

Interface graphique

L'interface permet à la banque l'évaluation du risque de l'octroi du crédit à partir du score obtenu .

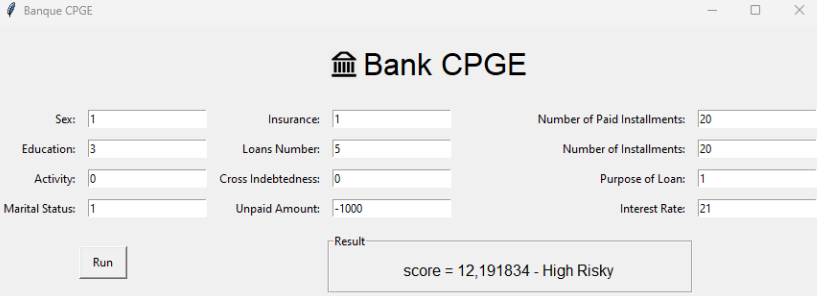

Figure – Interface graphique

Conclusion

- La logique floue permet de modéliser l'évaluation du risque basée sur des règles exprimées en termes linguistiques à partir des avis des experts.
- Les résultats obtenus à travers la logique floue et ceux découlant de l'expertise humaine sont vraisemblables si la définition des fonctions d'appartenance est correcte et les règles d'inférence adoptées sont précises.

Conclusion

- La logique floue permet de modéliser l'évaluation du risque basée sur des règles exprimées en termes linguistiques à partir des avis des experts.
- Les résultats obtenus à travers la logique floue et ceux découlant de l'expertise humaine sont vraisemblables si la définition des fonctions d'appartenance est correcte et les règles d'inférence adoptées sont précises.

Merci pour votre attention.

Annexes

Définition des fonctions d'appartenance

```
# Définition de l'univers de discours
univers = np.arange(0, 1.01 , 0.01)

# Création de la variable floue
sex = ctrl.Antecedent(univers, 'Sexe')

# Définition des fonctions d'appartenance
sex['Woman'] = fuzz.trapmf(univers, [0.5, 0.5,1,1])
sex['Man'] = fuzz.trapmf(univers, [0, 0, 0.5,0.5])

# Affichage des fonctions d'appartenance
plt.plot(univers, fuzz.trapmf(univers, [0.5, 0.5,1,1]), 'g', label='Woman')
plt.plot(univers, fuzz.trapmf(univers, [0, 0, 0.5,0.5]), 'b', label='Man')

plt.xlabel('Valeur')
plt.ylabel('Degré d\'appartenance')
plt.title('Fonctions d\'appartenance du sexe')
plt.legend()
plt.show()
```

```
# Assurance
# Définition de l'univers de discours
univers = np.arange(0, 1.01 , 0.01)

# Création de la variable floue
insurance_customer = ctrl.Antecedent(univers, 'Assurance')

# Définition des fonctions d'appartenance
insurance_customer['Uninsured'] = fuzz.trapmf(univers, [0.5, 0.5,1,1])
insurance_customer['Insured'] = fuzz.trapmf(univers, [0, 0, 0.5,0.5])

# Affichage des fonctions d'appartenance
plt.plot(univers, fuzz.trapmf(univers, [0.5, 0.5,1,1]), 'g', label='Uninsured')
plt.plot(univers, fuzz.trapmf(univers, [0, 0, 0.5,0.5]), 'b', label='Insured')

plt.xlabel('Valeur')
plt.ylabel('Degré d\'appartenance')
plt.title('Fonctions d\'appartenance de l assurance')
plt.legend()
plt.show()
```

Annexes

Définition des fonctions d'appartenance

```
#statut marital :
# Définition de l'univers de discours
univers = np.arange(0, 1.01 , 0.01)

# Création de la variable floue
marital_status = ctrl.Antecedent(univers, 'Statut marital')

# Définition des fonctions d'appartenance
marital_status['Single'] = fuzz.trapmf(univers, [0.5, 0.5,1,1])
marital_status['Married'] = fuzz.trapmf(univers, [0, 0, 0.5,0.5])

# Affichage des fonctions d'appartenance
plt.plot(univers, fuzz.trapmf(univers, [0.5, 0.5,1,1]), 'g', label='Married')
plt.plot(univers, fuzz.trapmf(univers, [0, 0, 0.5,0.5]), 'b', label='Single')

plt.xlabel('Valeur')
plt.ylabel('Degré d\'appartenance')
plt.title('Fonctions d\'appartenance du statut marital')
plt.legend()
plt.show()
```

Annexes

Définition des fonctions d'appartenance

```python
# Education :
# Définition de l'univers de discours
univers1 = np.arange(0, 5.1, 0.01)

# Création de la variable floue
education = ctrl.Antecedent(univers1, 'Education')

# Définition des fonctions d'appartenance
education['Low'] = fuzz.trapmf(univers1, [0, 0, 1.5,1.5])
education['Medium'] = fuzz.trapmf(univers1, [1.5, 2, 3,3.5])
education['High'] = fuzz.trapmf(univers1, [3.5,4, 5.1, 5.1])

# Affichage des fonctions d'appartenance
plt.plot(univers1, fuzz.trapmf(univers1, [0, 0, 1.5,1.5]), 'b',label='Low')
plt.plot(univers1, fuzz.trapmf(univers1, [1.5, 2, 3,3.5]), 'g',label='Medium')
plt.plot(univers1, fuzz.trapmf(univers1, [3.5,4, 5, 5]), 'r',label='High')

plt.xlabel('Valeur')
plt.ylabel('Degré d\'appartenance')
plt.title('Fonctions d\'appartenance de l education')
plt.legend()
plt.show()
```

Définition des fonctions d'appartenance

```python
# activite economique
# Définition de l'univers de discours
univers2 = np.arange(0, 3.01, 0.01)

# Création de la variable floue
customer_activity = ctrl.Antecedent(univers2, 'Activite')

# Définition des fonctions d'appartenance
customer_activity['Trade'] = fuzz.trapmf(univers2, [0, 0.6, 1,3])
customer_activity['AgricultureCraft'] = fuzz.trapmf(univers2, [0, 0.6, 1.2,2])
customer_activity['Others'] = fuzz.trapmf(univers2, [1.7,2.5, 5, 5])

# Affichage des fonctions d'appartenance
plt.plot(univers2, fuzz.trapmf(univers2, [0, 0, 0.75,1]), 'b', label='Trade')
plt.plot(univers2, fuzz.trapmf(univers2, [0.75, 1.25, 1.75,2]), 'g', label='AgricultureCraft')
plt.plot(univers2, fuzz.trapmf(univers2, [1.75,2.5, 5, 5]), 'r', label='Others')

plt.xlabel('Valeur')
plt.ylabel('Degré d\'appartenance')
plt.title('Fonctions d\'appartenance de l activité économique')
plt.legend()
plt.show()
```

Annexes

Définition des fonctions d'appartenance

```python
#definition de la variable de sortie
descriptive_variable = ctrl.Consequent(np.arange(0, 21, 1), 'Descriptive Variable')

# Univers de discours
universe3 = np.arange(0, 21, 1)

# Définition des fonctions d'appartenance
descriptive_variable['Low'] = fuzz.trimf(universe3, [0, 5, 10])
descriptive_variable['Medium'] = fuzz.trimf(universe3, [5, 10, 15])
descriptive_variable['High'] = fuzz.trimf(universe3, [10, 15, 20])

# Affichage des fonctions d'appartenance
plt.figure()
plt.plot(universe3, fuzz.trimf(universe3, [0, 5, 10]), 'b',  label='Low')
plt.plot(universe3, fuzz.trimf(universe3, [5, 10, 15]), 'g',  label='Medium')
plt.plot(universe3, fuzz.trimf(universe3, [10, 15, 20]), 'r',  label='High')
plt.title('Fonctions d\'appartenance de la variable descriptive')
plt.xlabel('Valeur')
plt.ylabel('degré d\'appartenance')
plt.legend()
plt.grid(True)
plt.show()
```

```
#Définition des règles
rule1 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['High'] & insurance_customer['Insured'] &
               customer_activity['Trade'], descriptive_variable['High'])

rule2 = ctrl.Rule(marital_status['Single'] & education['Medium'] & insurance_customer['Uninsured'] &
                  customer_activity['Trade'], descriptive_variable['High'])

rule3 = ctrl.Rule(marital_status['Married'] & education['Medium'] & insurance_customer['Insured'] &
                  customer_activity['Others'], descriptive_variable['Low'])

rule4 = ctrl.Rule(marital_status['Married'] & education['High'] & insurance_customer['Insured'] &
                  customer_activity['Others'], descriptive_variable['Low'])

rule5 = ctrl.Rule(education['Low'] & insurance_customer['Insured'] & customer_activity['Trade'],
                  descriptive_variable['Medium'])

rule6 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['Low'] & insurance_customer['Insured'] &
                  customer_activity['Trade'], descriptive_variable['High'])

rule7 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['Low'] & insurance_customer['Uninsured'] &
                  customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule8 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['Medium'] & insurance_customer['Insured'] &
                  customer_activity['Others'], descriptive_variable['Medium'])

rule9 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['Medium'] & insurance_customer['Insured'] &
                  customer_activity['AgricultureCraft'], descriptive_variable['Medium'])

rule10 = ctrl.Rule(sex['Man'] & marital_status['Single'] & education['High'] & insurance_customer['Uninsured'] &
                   customer_activity['Others'], descriptive_variable['High'])
```

Annexes

Définition des règles d'inférence

```
rule11 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['Medium'])

rule12 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule13 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['Medium'] & insurance_customer['Insured'] &
                   customer_activity['Others'], descriptive_variable['Low'])

rule14 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['Medium'] & insurance_customer['Insured'] &
                   customer_activity['AgricultureCraft'], descriptive_variable['Low'])

rule15 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['High'] & insurance_customer['Uninsured'] &
                   customer_activity['Others'], descriptive_variable['Medium'])

rule16 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['High'])

rule17 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule18 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Medium'] & insurance_customer['Insured']
&
                   customer_activity['Others'], descriptive_variable['Low'])

rule19 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Medium'] & insurance_customer['Insured']
&
                   customer_activity['AgricultureCraft'], descriptive_variable['Low'])

rule20 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['High'] & insurance_customer['Uninsured']
&
```

Annexes

Définition des règles d'inférence

```
rule21 = ctrl.Rule(sex['Woman'] & marital_status['Married'] & education['Low'] & insurance_customer['Uninsured']
&
               customer_activity['Trade'], descriptive_variable['High'])

rule22 = ctrl.Rule(sex['Woman'] & marital_status['Married'] & education['Low'] & insurance_customer['Uninsured']
&
               customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule23 = ctrl.Rule(sex['Woman'] & marital_status['Married'] & education['Medium'] & insurance_customer['Insured']
&
               customer_activity['Others'], descriptive_variable['Low'])

rule24 = ctrl.Rule(sex['Woman'] & marital_status['Married'] & education['Medium'] & insurance_customer['Insured']
&
               customer_activity['AgricultureCraft'], descriptive_variable['Low'])

rule25 = ctrl.Rule(sex['Woman'] & marital_status['Married'] & education['High'] & insurance_customer['Uninsured']
&
               customer_activity['Others'], descriptive_variable['Medium'])

rule26 = ctrl.Rule(marital_status['Single'] & education['Low'] & insurance_customer['Insured'] &
               customer_activity['Trade'], descriptive_variable['Medium'])

rule27 = ctrl.Rule(marital_status['Single'] & education['Low'] & insurance_customer['Insured'] &
               customer_activity['AgricultureCraft'], descriptive_variable['Medium'])

rule28 = ctrl.Rule(marital_status['Single'] & education['Medium'] & insurance_customer['Uninsured'] &
               customer_activity['Others'], descriptive_variable['High'])

rule29 = ctrl.Rule(marital_status['Single'] & education['Medium'] & insurance_customer['Uninsured'] &
               customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule30 = ctrl.Rule(marital_status['Single'] & education['High'] & insurance_customer['Insured'] &
               customer_activity['Others'], descriptive_variable['Low'])
```

Définition des règles d'inférence

```
rule30 = ctrl.Rule(marital_status['Single'] & education['High'] & insurance_customer['Insured'] &
                   customer_activity['Others'], descriptive_variable['Low'])

rule31 = ctrl.Rule(marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['High'])

rule32 = ctrl.Rule(marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['AgricultureCraft'], descriptive_variable['High'])

rule33 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['Medium'] & insurance_customer['Uninsured']
&
                   customer_activity['Trade'], descriptive_variable['Medium'])

rule34 = ctrl.Rule(sex['Man'] & marital_status['Married'] & education['High'] & insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['High'])

rule35 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Low'] & insurance_customer['Insured'] &
                   customer_activity['Others'], descriptive_variable['High'])

rule36 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Low'] & insurance_customer['Insured'] &
                   customer_activity['Trade'], descriptive_variable['Medium'])

rule37 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Medium'] &
insurance_customer['Uninsured'] &
                   customer_activity['Others'], descriptive_variable['Medium'])

rule38 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['Medium'] &
insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['High'])

rule39 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['High'] & insurance_customer['Insured'] &
                   customer_activity['Others'], descriptive_variable['Low'])

rule40 = ctrl.Rule(sex['Woman'] & marital_status['Single'] & education['High'] & insurance_customer['Insured'] &
                   customer_activity['Trade'], descriptive_variable['Medium'])
```

```
rule41 = ctrl.Rule(marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['Others'], descriptive_variable['High'])

rule42 = ctrl.Rule(marital_status['Married'] & education['Low'] & insurance_customer['Uninsured'] &
                   customer_activity['Trade'], descriptive_variable['Medium'])

rule43 = ctrl.Rule(marital_status['Married'] & education['Medium'] & insurance_customer['Insured'] &
                   customer_activity['AgricultureCraft'], descriptive_variable['Low'])

rule44 = ctrl.Rule(marital_status['Married'] & education['Medium'] & insurance_customer['Uninsured'] &
                   customer_activity['Others'], descriptive_variable['High'])

rule45 = ctrl.Rule(marital_status['Married'] & education['High'] & insurance_customer['Insured'] &
                   customer_activity['Trade'], descriptive_variable['Medium'])
```

Annexes

Définition de la variable de sortie et des fonctions d'appartenance associées

```
#definition de la variable de sortie
descriptive_variable = ctrl.Consequent(np.arange(0, 21, 1), 'Descriptive Variable')

# Univers de discours
universe3 = np.arange(0, 21, 1)

# Définition des fonctions d'appartenance
descriptive_variable['Low'] = fuzz.trimf(universe3, [0, 5, 10])
descriptive_variable['Medium'] = fuzz.trimf(universe3, [5, 10, 15])
descriptive_variable['High'] = fuzz.trimf(universe3, [10, 15, 20])

# Affichage des fonctions d'appartenance
plt.figure()
plt.plot(universe3, fuzz.trimf(universe3, [0, 5, 10]), 'b',  label='Low')
plt.plot(universe3, fuzz.trimf(universe3, [5, 10, 15]), 'g',  label='Medium')
plt.plot(universe3, fuzz.trimf(universe3, [10, 15, 20]), 'r',  label='High')
plt.title('Fonctions d\'appartenance de la variable descriptive')
plt.xlabel('Valeur')
plt.ylabel('degré d\'appartenance')
plt.legend()
plt.grid(True)
plt.show()
```

Annexes

Évaluation du système de contrôle et défuzzification

```python
# Évaluation du système de contrôle
risk_simulation.compute()

# Récupération du résultat
result = risk_simulation.output['Descriptive Variable']
print("Descriptive Variable:",result)
# Define the membership functions for DescriptiveVariable
universe_descriptive_variable = universe3
descriptive_variable_low = fuzz.trimf(universe_descriptive_variable, [0, 5, 10])
descriptive_variable_medium = fuzz.trimf(universe_descriptive_variable, [5, 10, 15])
descriptive_variable_high = fuzz.trimf(universe_descriptive_variable, [10, 15, 20])

# Evaluate the degree of membership to each linguistic term
degree_low = fuzz.interp_membership(universe_descriptive_variable, descriptive_variable_low, result)
degree_medium = fuzz.interp_membership(universe_descriptive_variable, descriptive_variable_medium, result)
degree_high = fuzz.interp_membership(universe_descriptive_variable, descriptive_variable_high, result)

# Print the degrees of membership
print("Degree of membership - Low:", degree_low)
print("Degree of membership - Medium:", degree_medium)
print("Degree of membership - High:", degree_high)

# Find the linguistic term with the maximum degree of membership
max_degree = max(degree_low, degree_medium, degree_high)

if max_degree == degree_low:
    linguistic_result = "Low"
elif max_degree == degree_medium:
    linguistic_result = "Medium"
else:
    linguistic_result = "High"

# Return the linguistic term with the maximum degree of membership
print("Linguistic Result:", linguistic_result)
```