Membership.go  
package fuzzifikasi

func TsukamotoFuzzy(x float64, low, medium, high TriangularMF) (lowDeg, medDeg, highDeg float64) {

lowDeg = low.Degree(x)

medDeg = medium.Degree(x)

highDeg = high.Degree(x)

return

}

// Degree returns the degree of membership for input x

func (mf TriangularMF) Degree(x float64) float64 {

if x <= mf.A || x >= mf.C {

return 0.0

} else if x == mf.B {

return 1.0

} else if x > mf.A && x < mf.B {

return (x - mf.A) / (mf.B - mf.A)

} else { // x > mf.B && x < mf.C

return (mf.C - x) / (mf.C - mf.B)

}

}

// TriangularMF represents a triangular membership function

type TriangularMF struct {

A float64 // left foot

B float64 // peak

C float64 // right foot

}

academic\_advising.go  
package fuzzifikasi

// Academic Advising

var (

AcademicAdvisingLow = TriangularMF{A: 0, B: 0, C: 3}

AcademicAdvisingMedium = TriangularMF{A: 2, B: 4, C: 6}

AcademicAdvisingHigh = TriangularMF{A: 5, B: 8, C: 8}

)

func FuzzifyAcademicAdvising(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, AcademicAdvisingLow, AcademicAdvisingMedium, AcademicAdvisingHigh)

}

age.go  
package fuzzifikasi

// Age Fuzzification

var (

AgeYoung = TriangularMF{A: 17, B: 18, C: 22}

AgeMiddle = TriangularMF{A: 20, B: 23, C: 26}

AgeOld = TriangularMF{A: 24, B: 30, C: 35}

)

func FuzzifyAge(age float64) (young, middle, old float64) {

return TsukamotoFuzzy(age, AgeYoung, AgeMiddle, AgeOld)

}

attendance.go

package fuzzifikasi

var (

AttendanceLow = TriangularMF{A: 0.0, B: 0.0, C: 0.75}

AttendanceMedium = TriangularMF{A: 0.6, B: 0.8, C: 1.0}

AttendanceHigh = TriangularMF{A: 0.85, B: 1.0, C: 1.0}

)

func FuzzifyAttendance(attendance float64) (low, medium, high float64) {

return TsukamotoFuzzy(attendance, AttendanceLow, AttendanceMedium, AttendanceHigh)

}

cca.go

package fuzzifikasi

// Core Course Average

var (

CCALow = TriangularMF{A: 0, B: 0, C: 60}

CCAMedium = TriangularMF{A: 50, B: 70, C: 90}

CCAHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyCCA(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, CCALow, CCAMedium, CCAHigh)

}

dfhu.go

package fuzzifikasi

// DFHU (Distance from Home University) Fuzzification

var (

DFHUNear = TriangularMF{A: 1, B: 1, C: 33}

DFHUMedium = TriangularMF{A: 20, B: 50, C: 80}

DFHUFar = TriangularMF{A: 60, B: 100, C: 100}

)

func FuzzifyDFHU(distance float64) (near, medium, far float64) {

return TsukamotoFuzzy(distance, DFHUNear, DFHUMedium, DFHUFar)

}

ees.go

package fuzzifikasi

// EES (Educational Effectiveness Score) Fuzzification

var (

EESLow = TriangularMF{A: 0, B: 0, C: 60}

EESMedium = TriangularMF{A: 50, B: 70, C: 90}

EESHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyEES(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, EESLow, EESMedium, EESHigh)

}

family\_income.go

package fuzzifikasi

// Family Income

var (

FamilyIncomeLow = TriangularMF{A: 0, B: 0, C: 60\_000}

FamilyIncomeMedium = TriangularMF{A: 40\_000, B: 100\_000, C: 160\_000}

FamilyIncomeHigh = TriangularMF{A: 120\_000, B: 200\_000, C: 200\_000}

)

func FuzzifyFamilyIncome(income float64) (low, medium, high float64) {

return TsukamotoFuzzy(income, FamilyIncomeLow, FamilyIncomeMedium, FamilyIncomeHigh)

}

final\_exam.go

package fuzzifikasi

var (

FinalExamLow = TriangularMF{A: 0, B: 0, C: 60}

FinalExamMedium = TriangularMF{A: 50, B: 70, C: 90}

FinalExamHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyFinalExam(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, FinalExamLow, FinalExamMedium, FinalExamHigh)

}

gpa.go

package fuzzifikasi

// Example: Define GPA fuzzy sets (adjust values as needed)

var (

GPALow = TriangularMF{A: 0.0, B: 0.0, C: 2.5}

GPAMedium = TriangularMF{A: 2.0, B: 2.75, C: 3.5}

GPAHigh = TriangularMF{A: 3.0, B: 4.0, C: 4.0}

)

func FuzzifyGPA(gpa float64) (low, medium, high float64) {

return TsukamotoFuzzy(gpa, GPALow, GPAMedium, GPAHigh)

}

hsg.go

package fuzzifikasi

// HSG (High School GPA) Fuzzification

var (

HSGLow = TriangularMF{A: 0.0, B: 0.0, C: 2.5}

HSGMedium = TriangularMF{A: 2.0, B: 2.75, C: 3.5}

HSGHigh = TriangularMF{A: 3.0, B: 4.0, C: 4.0}

)

func FuzzifyHSG(hsg float64) (low, medium, high float64) {

return TsukamotoFuzzy(hsg, HSGLow, HSGMedium, HSGHigh)

}

library\_usage.go

package fuzzifikasi

// Library Usage

var (

LibraryUsageLow = TriangularMF{A: 0, B: 0, C: 10}

LibraryUsageMedium = TriangularMF{A: 5, B: 15, C: 25}

LibraryUsageHigh = TriangularMF{A: 20, B: 30, C: 30}

)

func FuzzifyLibraryUsage(usage float64) (low, medium, high float64) {

return TsukamotoFuzzy(usage, LibraryUsageLow, LibraryUsageMedium, LibraryUsageHigh)

}

mes.go

package fuzzifikasi

// Midterm exam score

var (

MESLow = TriangularMF{A: 0, B: 0, C: 60}

MESMedium = TriangularMF{A: 50, B: 70, C: 90}

MESHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyMES(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, MESLow, MESMedium, MESHigh)

}

peer\_evaluation.go

package fuzzifikasi

// Peer Evaluation

var (

PeerEvalLow = TriangularMF{A: 0, B: 0, C: 60}

PeerEvalMedium = TriangularMF{A: 50, B: 70, C: 90}

PeerEvalHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyPeerEvaluation(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, PeerEvalLow, PeerEvalMedium, PeerEvalHigh)

}

peer\_reviews.go

package fuzzifikasi

// Peer Reviews

var (

PeerReviewsLow = TriangularMF{A: 0, B: 0, C: 60}

PeerReviewsMedium = TriangularMF{A: 50, B: 70, C: 90}

PeerReviewsHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyPeerReviews(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, PeerReviewsLow, PeerReviewsMedium, PeerReviewsHigh)

}

project\_score.go

package fuzzifikasi

var (

ProjectScoreLow = TriangularMF{A: 0, B: 0, C: 60}

ProjectScoreMedium = TriangularMF{A: 50, B: 70, C: 90}

ProjectScoreHigh = TriangularMF{A: 80, B: 100, C: 100}

)

func FuzzifyProjectScore(score float64) (low, medium, high float64) {

return TsukamotoFuzzy(score, ProjectScoreLow, ProjectScoreMedium, ProjectScoreHigh)

}

studty\_hourse.go

package fuzzifikasi

var (

StudyHoursLow = TriangularMF{A: 0, B: 0, C: 10}

StudyHoursMedium = TriangularMF{A: 5, B: 17.5, C: 30}

StudyHoursHigh = TriangularMF{A: 20, B: 40, C: 40}

)

func FuzzifyStudyHours(hours float64) (low, medium, high float64) {

return TsukamotoFuzzy(hours, StudyHoursLow, StudyHoursMedium, StudyHoursHigh)

}

inferensi.go

package inferensi

import (

"math"

"tsukamoto/internal/modules/fuzzifikasi"

)

// Inference performs Tsukamoto fuzzy inference

func Inference(

attendanceLow, attendanceMedium, attendanceHigh,

finalExamLow, finalExamMedium, finalExamHigh,

gpaLow, gpaMedium, gpaHigh,

projectScoreLow, projectScoreMedium, projectScoreHigh,

studyHoursLow, studyHoursMedium, studyHoursHigh,

ageYoung, ageMiddle, ageOld,

ccaLow, ccaMedium, ccaHigh,

dfhuNear, dfhuMedium, dfhuFar,

hsgLow, hsgMedium, hsgHigh,

eesLow, eesMedium, eesHigh,

familyIncomeLow, familyIncomeMedium, familyIncomeHigh,

libraryUsageLow, libraryUsageMedium, libraryUsageHigh,

peerEvalLow, peerEvalMedium, peerEvalHigh,

peerReviewsLow, peerReviewsMedium, peerReviewsHigh float64,

) float64 {

rules := Rules()

var weightedSum, weightSum float64

for \_, rule := range rules {

alpha := 1.0

if rule.AttendanceLow > 0 {

alpha = math.Min(alpha, attendanceLow)

}

if rule.AttendanceMedium > 0 {

alpha = math.Min(alpha, attendanceMedium)

}

if rule.AttendanceHigh > 0 {

alpha = math.Min(alpha, attendanceHigh)

}

if rule.FinalExamLow > 0 {

alpha = math.Min(alpha, finalExamLow)

}

if rule.FinalExamMedium > 0 {

alpha = math.Min(alpha, finalExamMedium)

}

if rule.FinalExamHigh > 0 {

alpha = math.Min(alpha, finalExamHigh)

}

if rule.GPALow > 0 {

alpha = math.Min(alpha, gpaLow)

}

if rule.GPAMedium > 0 {

alpha = math.Min(alpha, gpaMedium)

}

if rule.GPAHigh > 0 {

alpha = math.Min(alpha, gpaHigh)

}

if rule.ProjectScoreLow > 0 {

alpha = math.Min(alpha, projectScoreLow)

}

if rule.ProjectScoreMedium > 0 {

alpha = math.Min(alpha, projectScoreMedium)

}

if rule.ProjectScoreHigh > 0 {

alpha = math.Min(alpha, projectScoreHigh)

}

if rule.StudyHoursLow > 0 {

alpha = math.Min(alpha, studyHoursLow)

}

if rule.StudyHoursMedium > 0 {

alpha = math.Min(alpha, studyHoursMedium)

}

if rule.StudyHoursHigh > 0 {

alpha = math.Min(alpha, studyHoursHigh)

}

// Tambahkan variabel fuzzy baru

if rule.AgeYoung > 0 {

alpha = math.Min(alpha, ageYoung)

}

if rule.AgeMiddle > 0 {

alpha = math.Min(alpha, ageMiddle)

}

if rule.AgeOld > 0 {

alpha = math.Min(alpha, ageOld)

}

if rule.CCALow > 0 {

alpha = math.Min(alpha, ccaLow)

}

if rule.CCAMedium > 0 {

alpha = math.Min(alpha, ccaMedium)

}

if rule.CCAHigh > 0 {

alpha = math.Min(alpha, ccaHigh)

}

if rule.DFHUNear > 0 {

alpha = math.Min(alpha, dfhuNear)

}

if rule.DFHUMedium > 0 {

alpha = math.Min(alpha, dfhuMedium)

}

if rule.DFHUFar > 0 {

alpha = math.Min(alpha, dfhuFar)

}

if rule.HSGLow > 0 {

alpha = math.Min(alpha, hsgLow)

}

if rule.HSGMedium > 0 {

alpha = math.Min(alpha, hsgMedium)

}

if rule.HSGHigh > 0 {

alpha = math.Min(alpha, hsgHigh)

}

if rule.EESLow > 0 {

alpha = math.Min(alpha, eesLow)

}

if rule.EESMedium > 0 {

alpha = math.Min(alpha, eesMedium)

}

if rule.EESHigh > 0 {

alpha = math.Min(alpha, eesHigh)

}

if rule.FamilyIncomeLow > 0 {

alpha = math.Min(alpha, familyIncomeLow)

}

if rule.FamilyIncomeMedium > 0 {

alpha = math.Min(alpha, familyIncomeMedium)

}

if rule.FamilyIncomeHigh > 0 {

alpha = math.Min(alpha, familyIncomeHigh)

}

if rule.LibraryUsageLow > 0 {

alpha = math.Min(alpha, libraryUsageLow)

}

if rule.LibraryUsageMedium > 0 {

alpha = math.Min(alpha, libraryUsageMedium)

}

if rule.LibraryUsageHigh > 0 {

alpha = math.Min(alpha, libraryUsageHigh)

}

if rule.PeerEvalLow > 0 {

alpha = math.Min(alpha, peerEvalLow)

}

if rule.PeerEvalMedium > 0 {

alpha = math.Min(alpha, peerEvalMedium)

}

if rule.PeerEvalHigh > 0 {

alpha = math.Min(alpha, peerEvalHigh)

}

if rule.PeerReviewsLow > 0 {

alpha = math.Min(alpha, peerReviewsLow)

}

if rule.PeerReviewsMedium > 0 {

alpha = math.Min(alpha, peerReviewsMedium)

}

if rule.PeerReviewsHigh > 0 {

alpha = math.Min(alpha, peerReviewsHigh)

}

// Tsukamoto: Use alpha as the weight for the rule's output value

weightedSum += alpha \* rule.OutputValue

weightSum += alpha

}

// Defuzzification: Weighted average

if weightSum == 0 {

return 50 // Avoid division by zero

}

return weightedSum / weightSum

}

func PerformInference(

attendance, finalExam, gpa, projectScore, studyHours,

age, cca, dfhu, hsg, ees, familyIncome, libraryUsage, peerEval, peerReviews float64,

) float64 {

// Fuzzify inputs

attLow, attMed, attHigh := fuzzifikasi.FuzzifyAttendance(attendance)

examLow, examMed, examHigh := fuzzifikasi.FuzzifyFinalExam(finalExam)

gpaLow, gpaMed, gpaHigh := fuzzifikasi.FuzzifyGPA(gpa)

projLow, projMed, projHigh := fuzzifikasi.FuzzifyProjectScore(projectScore)

hoursLow, hoursMed, hoursHigh := fuzzifikasi.FuzzifyStudyHours(studyHours)

ageYoung, ageMiddle, ageOld := fuzzifikasi.FuzzifyAge(age)

ccaLow, ccaMed, ccaHigh := fuzzifikasi.FuzzifyCCA(cca)

dfhuNear, dfhuMed, dfhuFar := fuzzifikasi.FuzzifyDFHU(dfhu)

hsgLow, hsgMed, hsgHigh := fuzzifikasi.FuzzifyHSG(hsg)

eesLow, eesMed, eesHigh := fuzzifikasi.FuzzifyEES(ees)

famLow, famMed, famHigh := fuzzifikasi.FuzzifyFamilyIncome(familyIncome)

libLow, libMed, libHigh := fuzzifikasi.FuzzifyLibraryUsage(libraryUsage)

peerEvalLow, peerEvalMed, peerEvalHigh := fuzzifikasi.FuzzifyPeerEvaluation(peerEval)

peerReviewsLow, peerReviewsMed, peerReviewsHigh := fuzzifikasi.FuzzifyPeerReviews(peerReviews)

// Perform inference

return Inference(

attLow, attMed, attHigh,

examLow, examMed, examHigh,

gpaLow, gpaMed, gpaHigh,

projLow, projMed, projHigh,

hoursLow, hoursMed, hoursHigh,

ageYoung, ageMiddle, ageOld,

ccaLow, ccaMed, ccaHigh,

dfhuNear, dfhuMed, dfhuFar,

hsgLow, hsgMed, hsgHigh,

eesLow, eesMed, eesHigh,

famLow, famMed, famHigh,

libLow, libMed, libHigh,

peerEvalLow, peerEvalMed, peerEvalHigh,

peerReviewsLow, peerReviewsMed, peerReviewsHigh,

)

}

rules.go

deffuzifikasi.go

package deffuzifikasi

import (

inf "tsukamoto/internal/modules/inferensi"

)

// Defuzzify maps a crisp output value to a performance category

func Defuzzify(crispOutput float64) string {

switch {

case crispOutput >= 87.5:

return "Excellent" // Close to 100 (top-performing)

case crispOutput >= 62.5:

return "Good" // Close to 75 (above-average)

case crispOutput >= 40:

return "Satisfactory" // Close to 50 (average)

case crispOutput >= 12.5:

return "Needs Improvement" // Close to 25 (below-average)

default:

return "Poor" // Close to 0 (poor)

}

}

// PerformFullInference combines fuzzification, inference, and defuzzification

func PerformFullInference(

attendance, finalExam, gpa, projectScore, studyHours,

age, cca, dfhu, hsg, ees, familyIncome, libraryUsage, peerEval, peerReviews float64,

) (string, float64) {

// Perform inference to get crisp output

crispOutput := inf.PerformInference(

attendance, finalExam, gpa, projectScore, studyHours,

age, cca, dfhu, hsg, ees, familyIncome, libraryUsage, peerEval, peerReviews,

)

// Defuzzify to get performance category

category := Defuzzify(crispOutput)

return category, crispOutput

}