prgamming language is julia!!

```julia
using Statistics
using Random

function e_in(data, s, theta, data_size)
    err = zeros(data_size)*1

    for i in 1:data_size
        h = s*sign(data[i][1]-theta)
        if h != data[i][2]
            err[i] = 1
        end
    end
    #@show err
    #@show mean(err)
    return mean(err)
end

function stump(tau = 0.5, data_size = 2)
    error_out_in = zeros(10000)*1
    for j in 1:10000 #10000

        Random.seed!(j+1000)
        x = rand(data_size).-0.5
        #@show x
        flip = rand(data_size)
        flip = flip .<= tau
        flip = convert(Vector{Int64}, flip)
        #@show flip
        y = sign.(x)

        #y = convert(Vector{Int64}, y)
        #@show y
        y = y .* (1 .- 2 .* flip)

        data = collect(zip(x, y))


#13
println(stump(0,2))
#14
println(stump(0, 128))
#15
println(stump(0.2,2))
#16
println(stump(0.2, 128))
```

```
result
0.2983566292451094
0.003828696497570773
0.4278488034085101
0.014607996660204566
```

```julia
function e_in(data, s, theta, data_size,d)
    err = zeros(data_size)*1

    for i in 1:data_size
        h = s*sign(data[i][d]-theta)
        if h != data[i][11]
            err[i] = 1
        end
    end
    return mean(err)
end
function muti_dstump(;mode = 1, test=0)
    # mode 1: best of best / 2: worst of best
    # test 1: test e_out

    error_out_in = zeros(10000)*1
    x=[]
    datafile = open("hw2_train.txt","r") #read the data
    lines = readlines(datafile)
    close(datafile)
    for (i, line) in enumerate(lines)
        append!(x,[parse.(Float64, split(line,"\t"))])
    end
    data_size = size(x)[1]
    #@show x
    #x:256*11


    sgn = [1.0,-1.0]
    min_dim_e=0
    if mode==1
        min_dim_e = 10000000000
    else
        min_dim_e = -1000000000
    end

    ans_big = [0.0,0.0,0.0]#dim, s, theta
    for d in 1:10

        data = sort(x, by = t -> t[d])

        min_e = 100000000000000000000
        s_dot_theta = 10000000000000
        ans_small = [0.0, 0.0] #s,theta

        for s in sgn
        #@show 7
            for i in 1:data_size-1
                theta = (data[i][d]+data[i+1][d]) / 2
                error = e_in(data, s, theta, data_size,d)
                if (error < min_e) || ((error==min_e) && (s*theta < s_dot_thet
a))

                    min_e = error
                    s_dot_theta = s*theta
                    ans_small[1] = s
                    ans_small[2] = theta
                    #@show s, theta, error, i
                end

            end
            #@show data_size
            theta = -10000000000000000000000
            error = e_in(data, s, theta, data_size, d)
            #@show s, theta, error
            if (error < min_e) || ((error==min_e) && (s*theta < s_dot_theta))
                min_e = error
                s_dot_theta = s*theta
                ans_small[1] = s
                ans_small[2] = theta
            end

        end
        if (min_e < min_dim_e) && mode==1
            min_dim_e = min_e
            ans_big[1] = d
            ans_big[2] = ans_small[1]
            ans_big[3] = ans_small[2]
        end
        if (min_e > min_dim_e) && mode==2
            min_dim_e = min_e
            ans_big[1] = d
            ans_big[2] = ans_small[1]
            ans_big[3] = ans_small[2]
        end
    end
    if test==0
        #@show 0
        #@show ans_big
        return(min_dim_e)
    else
        #@show 1
        y=[]
        datafile = open("hw2_test.txt","r") #read the data
        lines = readlines(datafile)
        close(datafile)
        for (i, line) in enumerate(lines)
            append!(y,[parse.(Float64, split(line,"\t"))])
        end
        #@show ans_big
        data_size = size(y)[1]
        err = zeros(data_size)*1
        d = trunc(Int,ans_big[1])
        s = ans_big[2]
        theta = ans_big[3]

        for i in 1:data_size
            h = s*sign(y[i][d]-theta)
            if h != y[i][11]
                err[i] = 1
            end
        end
        #@show mean(err)
        return mean(err)
    end

end


# mode 1: best of best / 2: worst of best
# test 0: train e_in / 1: test e_out
#17
println(muti_dstump(mode = 1, test = 0))
#18
println(muti_dstump(mode = 1, test = 1))
#19
println(muti_dstump(mode = 2, test = 0)-muti_dstump(mode = 1, test = 0))
#20
println(muti_dstump(mode = 2, test = 1)-muti_dstump(mode = 1, test = 1))
# 0.026041666666666668
# 0.078125
# 0.30208333333333333
# 0.34375
```