DAFTAR PUSTAKA

Abasabadi, S. et al. (2022) 'Hybrid feature selection based on SLI and genetic algorithm for microarray datasets', Journal of Supercomputing, 78(18), pp. 19725–19753. doi:10.1007/s11227-022-04650-w.

Abiodun, E.O. et al. (2021) 'A systematic review of emerging feature selection optimization methods for optimal text classification: the present state and prospective opportunities', Neural Computing and Applications, 33(22), pp. 15091–15118. doi:10.1007/s00521-021-06406-8.

Aftab, A.I.S. and Matloob, F. (2019) 'Performance Analysis of Resampling Techniques on Class Imbalance Issue in Software Defect Prediction', International Journal of Information Technology and Computer Science, 11(11), pp. 44–53. doi:10.5815/ijitcs.2019.11.05.

Agrawal, P. et al. (2021) 'Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)', IEEE Access, 9, pp. 26766–26791. doi:10.1109/ACCESS.2021.3056407.

Al-Wajih, R. et al. (2021) 'Hybrid binary grey Wolf with Harris hawks optimizer for feature selection', IEEE Access, 9, pp. 31662–31677. doi:10.1109/ACCESS.2021.3060096.

Azad, C. et al. (2022) 'Prediction model using SMOTE, genetic algorithm and decision tree (PMSGD) for classification of diabetes mellitus', Multimedia Systems, 28(4), pp. 1289–1307. doi:10.1007/s00530-021-00817-2.

Beinecke, J. and Heider, D. (2021) 'Gaussian noise up-sampling is better suited than SMOTE and ADASYN for clinical decision making', BioData Mining, 14(1), pp. 1–11. doi:10.1186/s13040-021-00283-6.

Bhattacharyya, T. et al. (2020) 'Mayfly in Harmony: A new hybrid meta-heuristic feature selection algorithm', IEEE Access, 8, pp. 195929–195945. doi:10.1109/ACCESS.2020.3031718.

Cheng, R. and Jin, Y. (2015) 'A competitive swarm optimizer for large scale optimization', IEEE Transactions on Cybernetics, 45(2), pp. 191–204. doi:10.1109/TCYB.2014.2322602.

Christo, V.R.E. et al. (2022) 'Feature Selection and Instance Selection from Clinical Datasets Using Co-operative Co-evolution and Classification Using Random Forest', IETE Journal of Research, 68(4), pp. 2508–2521. doi:10.1080/03772063.2020.1713917.

Cui, X. et al. (2020) 'A Hybrid Improved Dragonfly Algorithm for Feature Selection', IEEE Access, 8, pp. 155619–155629. doi:10.1109/ACCESS.2020.3012838.

Dawson, D.C. (2009) 02 2009 Dawson. C., Introduction to Research Methods. 4th edn.

Durugkar, S.R. et al. (2022) 'Introduction to data mining', Data Mining and Machine Learning Applications [Preprint]. doi:10.1002/9781119792529.ch1.

El-Kenawy, E.S. and Eid, M. (2020) 'Hybrid gray wolf and particle swarm optimization for feature selection', International Journal of Innovative Computing, Information and Control, 16(3), pp. 831–844. doi:10.24507/ijicic.16.03.831.

El-Shafiey, M.G. et al. (2022) 'A hybrid GA and PSO optimized approach for heart-disease prediction based on random forest', Multimedia Tools and Applications, 81(13), pp. 18155–18179. doi:10.1007/s11042-022-12425-x.

Elgamal, Z.M. et al. (2020) 'An improved harris hawks optimization algorithm with simulated annealing for feature selection in the medical field', IEEE Access, 8, pp. 186638–186652. doi:10.1109/ACCESS.2020.3029728.

Eshtay, M., Faris, H. and Obeid, N. (2018) 'Improving Extreme Learning Machine by Competitive Swarm Optimization and its application for medical diagnosis problems', Expert Systems with Applications, 104, pp. 134–152. doi:10.1016/j.eswa.2018.03.024.

Gu, S., Cheng, R. and Jin, Y. (2018) 'Feature selection for high-dimensional classification using a competitive swarm optimizer', Soft Computing, 22(3), pp. 811–822. doi:10.1007/s00500-016-2385-6.

Guo, Y. et al. (2019) 'Multi-Label Bioinformatics Data Classification with Ensemble Embedded Feature Selection', IEEE Access, 7(Mddm), pp. 103863–103875. doi:10.1109/ACCESS.2019.2931035.

Han, J., Kamber, M. and Pei, J. (2012) Data-Mining.-Concepts-and-Techniques.pdf. Third Edit. Morgan Kaufmann.

Hosseini, E. et al. (2021) 'Novel metaheuristic based on multiverse theory for optimization problems in emerging systems', Applied Intelligence, 51(6), pp. 3275–3292. doi:10.1007/s10489-020-01920-z.

Khaire, U.M. and Dhanalakshmi, R. (2022) 'Stability of feature selection algorithm: A review', Journal of King Saud University - Computer and Information Sciences, 34(4), pp. 1060–1073. doi:10.1016/j.jksuci.2019.06.012.

Khalid, A.M. et al. (2022) 'BCOVIDOA: A Novel Binary Coronavirus Disease Optimization Algorithm for Feature Selection', Knowledge-Based Systems, 248, p. 108789. doi:10.1016/j.knosys.2022.108789.

Khalid, A.M., Hosny, K.M. and Mirjalili, S. (2022) 'COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle', Neural Computing and Applications, 34(24), pp. 22465–22492. doi:10.1007/s00521-022-07639-x.

Khan, S.I. and Hoque, A.S.M.L. (2020) 'SICE: an improved missing data imputation technique', Journal of Big Data, 7(1). doi:10.1186/s40537-020-00313-w.

Khurma, R.A. et al. (2022) ‘A Review of the Modification Strategies of the Nature Inspired Algorithms for Feature Selection Problem’, Mathematics, 10(3), pp. 1–45. doi:10.3390/math10030464.

Lan, K. et al. (2018) ‘A Survey of Data Mining and Deep Learning in Bioinformatics’, Journal of Medical Systems, 42(8). doi:10.1007/s10916-018-1003-9.

Leavy, P. (2017) Research Design. 2nd edn. 2017: THE GUILFORD PRESS, New York, London.

Lockett, A.J. (2020) ‘No free lunch theorems’, Natural Computing Series, 1(1), pp. 287–322. doi:10.1007/978-3-662-62007-6_12.

Mauluddin, S., Ikbal, I. and Nursikuwagus, A. (2020) ‘Complexity and performance comparison of genetic algorithm and ant colony for best solution timetable class’, Journal of Engineering Science and Technology, 15(1), pp. 276–290.

Moradi, P. and Gholampour, M. (2016) ‘A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy’, Applied Soft Computing Journal, 43, pp. 117–130. doi:10.1016/j.asoc.2016.01.044.

Muntasir Nishat, M. et al. (2022) ‘A Comprehensive Investigation of the Performances of Different Machine Learning Classifiers with SMOTE-ENN Oversampling Technique and Hyperparameter Optimization for Imbalanced Heart Failure Dataset’, Scientific Programming, 2022(Cvd). doi:10.1155/2022/3649406.

Nopiah, Z.M. et al. (2010) ‘Time complexity estimation and optimisation of the genetic algorithm clustering method’, WSEAS Transactions on Mathematics, 9(5), pp. 334–344.

Qamar, U. and Raza, M.S. (2020) Data Science Concepts and Techniques with Applications, Data Science Concepts and Techniques with Applications. doi:10.1007/978-981-15-6133-7.

Rong, M., Gong, D. and Gao, X. (2019) ‘Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends’, IEEE Access, 7, pp. 19709–19725. doi:10.1109/ACCESS.2019.2894366.

Rostami, M., Berahmand, K. and Forouzandeh, S. (2021) ‘A novel community detection based genetic algorithm for feature selection’, Journal of Big Data, 8(1). doi:10.1186/s40537-020-00398-3.

Salecha, A. (2021) ‘Time Complexity Analysis of an Evolutionary Algorithm for approximating Nash Equilibriums’, pp. 2–4. Available at: https://arxiv.org/abs/2110.13563.

Sani, H.M., Lei, C. and Neagu, D. (2018) of Decision Tree Algorithms. Springer International Publishing. doi:10.1007/978-3-030-04191-5.

Too, J. and Abdullah, A.R. (2021) A new and fast rival genetic algorithm for feature selection, Journal of Supercomputing. Springer US.

doi:10.1007/s11227-020-03378-9.

Wah, Y.B. et al. (2018) ‘Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy’, Pertanika Journal of Science and Technology, 26(1), pp. 329–340.

Zebari, R. et al. (2020) ‘A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction’, Journal of Applied Science and Technology Trends, 1(2), pp. 56–70. doi:10.38094/jastt1224.

LAMPIRAN

Lampiran 1. Kode Algoritma Seleksi Fitur COVIDOA-C

```
function [sFeat, Sf, Nf, f1score] = HCOVIDOARGAknn(feat, label, HO, nPop,
MaxIt, minVal, maxVal, D, CostFunction, MR, CR, shifttingNo,
numOfSubprotiens)

    VarMin = zeros(1, D) .* minVal;  % O(D)
    VarMax = ones(1, D) .* maxVal;   % O(D)
    empty_individual.Position = [];  % O(1)
    empty_individual.Cost = [];      % O(1)
    pop = repmat(empty_individual, nPop, 1);  % O(nPop)
    alpha = rand(nPop, D);           % O(nPop * D)
    fitness = @FitnessFunction;      % O(1)
    D = size(feat, 2);               % O(feat)
    X = zeros(nPop, D);              % O(nPop * D)

    % Inisialisasi Populasi
    for i = 1:nPop
        pop(i).Position = minVal + alpha(i, :) .* (maxVal - minVal);  % O(D)
        for d = 1:D
            pop(i).Position = 1./(1 + exp(-pop(i).Position));  % O(D)
            if pop(i).Position > rand()  % O(1)
                X(i, d) = 1;  % O(1)
            else
                X(i, d) = 0;  % O(1)
            end
        end
    end

    bestsol.Position = zeros(1, nPop);  % O(nPop)
    bestsol.Cost = inf;  % O(1)
    for i = 1:nPop
        pop(i).Cost = CostFunction(pop(i).Position);  % O(1)
        pop(i).Cost = fitness(feat, label, X(i, :), HO);  % O(1)
        if pop(i).Cost < bestsol.Cost  % O(1)
            bestsol.Cost = pop(i).Cost;  % O(1)
            bestsol.Position = pop(i).Position;  % O(D)
            Xbestsol = X(i, :);  % O(D)
        end
    end

    CR = round(CR * nPop);  % O(1)
    X1 = zeros(CR, D);      % O(CR * D)
    X2 = zeros(CR, D);      % O(CR * D)
    curve = nan(MaxIt, 1);  % O(MaxIt)

    % Main Loop
    for it = 1:MaxIt  % O(MaxIt)
        % Virus Replication Phase
        c = [pop.Cost];  % O(nPop)
        avgc = mean(c);  % O(nPop)
        if avgc ~= 0  % O(1)
            c = c / avgc;  % O(nPop)
        end
        probs = exp(nPop * c);  % O(nPop)
        x = zeros(nPop, D);  % O(nPop * D)
```

```
        for k = 1:nPop  % O(nPop)
            parent = pop(RouletteWheelSelection(probs));  % O(nPop)
            parent.Position = max(parent.Position, VarMin);  % O(D)
            parent.Position = min(parent.Position, VarMax);  % O(D)
            x(k, :) = parent.Position;  % O(D)
            for t = 1:numOfSubprotiens  % O(numOfSubprotiens)
                for i = 1:D-shifttingNo  % O(D - shifttingNo)
                    x(k, i) = x(k, i + shifttingNo);  % O(1)
                end
                r = rand();  % O(1)
                x(k, :) = [x(k, 1:D - shifttingNo) r];  % O(D)
            end
            for i = 1:CR  % O(CR)
                k1 = RouletteWheelSelection(probs);  % O(nPop)
                k2 = RouletteWheelSelection(probs);  % O(nPop)
                P1 = X(k1, :);  % O(D)
                P2 = X(k2, :);  % O(D)
                ind = randi([1, nPop - 1]);  % O(1)
                X1(i, :) = [P1(1:ind), P2(ind + 1:D)];  % O(D)
                X2(i, :) = [P2(1:ind), P1(ind + 1:D)];  % O(D)
            end
            Xnewvirus = [X1; X2];  % O(2 * CR * D)
            for i = 1:2 * CR  % O(2 * CR)
                for d = 1:D  % O(D)
                    if rand() <= MR  % O(1)
                        Xnewvirus(i, d) = 1 - Xnewvirus(i, d);  % O(1)
                        Xbestsol = Xnewvirus(i, :);  % O(D)
                    end
                end
            end
            subprotien = ones(nPop, D);  % O(nPop * D)
            subprotien(i, :) = x(k, :);  % O(D)
            Xnewvirus(i, :) = UniformCrossover(subprotien(1, :),
subprotien(2, :));  % O(D)
            Xnewvirus(i, :) = max(Xnewvirus(i, :), VarMin);  % O(D)
            Xnewvirus(i, :) = min(Xnewvirus(i, :), VarMax);  % O(D)
        end
        for t = 1:nPop  % O(nPop)
            childcost = CostFunction(Xnewvirus(t, :));  % O(1)
            if childcost < bestsol.Cost  % O(1)
                bestsol.Position = Xnewvirus(t, :);  % O(D)
                bestsol.Cost = childcost;  % O(1)
            end
        end
        newPop = repmat(empty_individual, nPop, 1);  % O(nPop)
        % Mutation
        for l = 1:nPop  % O(nPop)
            for k = 1:D  % O(D)
                R = rand() < 0.5;  % O(1)
                if R < MR  % O(1)
                    Xnewvirus(l, k) = minVal + rand * (maxVal - minVal);  %
O(1)
                end
                newPop(l).Position = Xnewvirus(l, :);  % O(D)
            end
            newPop(l).Position = max(newPop(l).Position, VarMin);  % O(D)
            newPop(l).Position = min(newPop(l).Position, VarMax);  % O(D)
            newPop(l).Cost = CostFunction(newPop(l).Position);  % O(1)
```

```
            if newPop(l).Cost < bestsol.Cost  % O(1)
                bestsol.Position = newPop(l).Position;  % O(D)
                bestsol.Cost = newPop(l).Cost;  % O(1)
            end
        end
        pop = SortPopulation([pop; newPop]);  % O((nPop + nPop) * log(nPop))
        Pos = 1:D;  % O(D)
        Sf = Pos(Xbestsol == 1);  % O(D)
        Nf = length(Sf);  % O(1)
        sFeat = feat(:, Sf);  % O(feat * Nf)
        pop = pop(1:nPop);  % O(nPop)
        curve(it) = bestsol.Cost;  % O(1)

        % Display Iteration Information
        disp(['Iteration ' num2str(it) ': Best Solution = '
num2str(curve(it))]);  % O(1)
    end
end
```

Lampiran 2. Kode Algoritma Seleksi Fitur ACO

```
function [sFeat, Sf, Nf] = jACO(feat, label, N, max_Iter, tau, eta, alpha,
beta, rho, HO)

    % Objective function
    fun = @jFitnessFunction;  % O(1)
    % Number of dimensions
    dim = size(feat, 2);  % O(feat)
    % Initial Tau & Eta
    tau = tau * ones(dim, dim);  % O(dim^2)
    eta = eta * ones(dim, dim);  % O(dim^2)
    % Pre
    fitG = inf;  % O(1)
    fit = zeros(1, N);  % O(N)

    curve = inf;  % O(1)
    t = 1;  % O(1)
    %---Iterations start-----------------------------------------
    while t <= max_Iter  % O(max_Iter)
        % Reset ant
        X = zeros(N, dim);  % O(N * dim)
        for i = 1:N  % O(N)
            % Random number of features
            num_feat = randi([1, dim]);  % O(1)
            % Ant starts with a random position
            X(i, 1) = randi([1, dim]);  % O(1)
            k = [];
            if num_feat > 1  % O(1)
                for d = 2:num_feat  % O(num_feat)
                    % Start with the previous tour
                    k = [k(1:end), X(i, d-1)];  % O(d-1)
                    % Edge/Probability Selection
                    P = (tau(k(end), :) .^ alpha) .* (eta(k(end), :) .^
beta);  % O(dim)
                    % Set selected position = 0 probability
                    P(k) = 0;  % O(d-1)
                    % Convert probability
                    prob = P ./ sum(P(:));  % O(dim)
                    % Roulette Wheel selection
                    route = jRouletteWheelSelection(prob);  % O(dim)
                    % Store selected position to be the next tour
                    X(i, d) = route;  % O(1)
                end
            end
        end

        % Binary
        X_bin = zeros(N, dim);  % O(N * dim)
        for i = 1:N  % O(N)
            % Binary form
            ind = X(i, :);
            ind(ind == 0) = [];
            X_bin(i, ind) = 1;  % O(length(ind))
        end

        % Fitness
        for i = 1:N  % O(N)
```

```
                fit(i) = fun(feat, label, X_bin(i, :), HO);  % O(1)
                % Global update
                if fit(i) < fitG  % O(1)
                    Xgb = X(i, :);  % O(dim)
                    fitG = fit(i);  % O(1)
                end
            end

            % [Pheromone update rule on tauK]
            tauK = zeros(dim, dim);  % O(dim^2)
            for i = 1:N  % O(N)
                % Update Pheromones
                tour = X(i, :);
                tour(tour == 0) = [];
                len_x = length(tour);  % O(1)
                tour = [tour(1:end), tour(1)];  % O(len_x)
                for d = 1:len_x  % O(len_x)
                    % Feature selected on graph
                    x = tour(d);
                    y = tour(d + 1);
                    % Update delta tau k on graph
                    tauK(x, y) = tauK(x, y) + (1 / (1 + fit(i)));  % O(1)
                end
            end

            % [Pheromone update rule on tauG]
            tauG = zeros(dim, dim);  % O(dim^2)
            tour = Xgb;  % O(dim)
            tour(tour == 0) = [];
            len_g = length(tour);  % O(1)
            tour = [tour(1:end), tour(1)];  % O(len_g)
            for d = 1:len_g  % O(len_g)
                % Feature selected on graph
                x = tour(d);
                y = tour(d + 1);
                % Update delta tau G on graph
                tauG(x, y) = 1 / (1 + fitG);  % O(1)
            end

            % Evaporate pheromone
            tau = (1 - rho) * tau + tauK + tauG;  % O(dim^2)

            % Save
            curve(t) = fitG;  % O(1)
            fprintf('\nIteration %d Best (ACO)= %f', t, curve(t))  % O(1)
            t = t + 1;  % O(1)
        end
        % Select features based on the selected index
        Sf = Xgb;  % O(dim)
        Sf(Sf == 0) = [];
        sFeat = feat(:, Sf);  % O(dim * length(Sf))
        Nf = length(Sf);  % O(1)
    end
```

Lampiran 3. Kode Algoritma Seleksi Fitur PSO

```
function [sFeat,Sf,Nf,curve,pred,accuracy,precision,recall] =
jPSO(feat,label,N,max_Iter,c1,c2,w,HO)
% Parameters

lb    = 0;            % O(1)
ub    = 1;            % O(1)
thres = 0.5;          % O(1)
tic;                  % O(1)

fun = @jFitnessFunction;  % O(1)

dim = size(feat,2);  % O(feat)

X   = zeros(dim,dim);  % O(dim^2)
V   = zeros(dim,dim);  % O(dim^2)
for i = 1:N
  for d = 1:dim
    X(i,d) = lb + (ub - lb) * rand();  % O(1)
  end
end

fit  = zeros(1,N);  % O(N)
fitG = inf;           % O(1)
for i = 1:N
  fit(i) = fun(feat,label,(X(i,:) > thres),HO);  % O(1)
  if fit(i) < fitG
    Xgb  = X(i,:);  % O(dim)
    fitG = fit(i);   % O(1)
  end
end

Xpb  = X;   % O(dim * N)
fitP = fit; % O(N)

curve = inf;  % O(1)
t = 1;        % O(1)

while t <= max_Iter  % Iterasi sebanyak max_Iter kali
  for i = 1:N
    for d = 1:dim
      r1 = rand();  % O(1)
      r2 = rand();  % O(1)
      V(i,d) = w * V(i,d) + c1 * r1 * (Xpb(i,d) - X(i,d)) + c2 * r2 *
(Xgb(d) - X(i,d));  % O(1)
      X(i,d) = X(i,d) + V(i,d);  % O(1)
    end
    XB = X(i,:); XB(XB > ub) = ub; XB(XB < lb) = lb;  % O(dim)
    X(i,:) = XB;  % O(dim)
    fit(i) = fun(feat,label,(X(i,:) > thres),HO);  % O(1)
    if fit(i) < fitP(i)
      Xpb(i,:) = X(i,:);  % O(dim)
      fitP(i)  = fit(i);  % O(1)
    end
    if fitP(i) < fitG
      Xgb  = Xpb(i,:);  % O(dim)
      fitG = fitP(i);   % O(1)
```

```
    end
  end
  curve(t) = fitG;  % O(1)
  fprintf('\nIteration %d GBest (PSO)= %f',t,curve(t))  % O(1)
  t = t + 1;  % O(1)
end

% Select features based on selected index
Pos    = 1:dim;
Sf     = Pos((Xgb > thres) == 1);
sFeat = feat(:,Sf);
Nf     = length(Sf);
end
```