Code 1:

clc; %clear screen

clear all;

close all;

baseDir = 'C:\Users\Dimple\Downloads\Problem\_Statement\Problem Statement\Dataset';

outputDir = fullfile(baseDir, 'Output'); % Directory to save output images

if ~exist(outputDir, 'dir')

mkdir(outputDir); % make output directory

end

fileName = 'dec\_2023.jpg'; %write all files names individually.

fullFilePath = fullfile(baseDir, fileName);

% get the year and month for each image

[~, name, ~] = fileparts(fileName);

dateParts = regexp(name, '(\w+)\_(\d{4})', 'tokens');

monthStr = lower(dateParts{1}{1});

yearStr = dateParts{1}{2};

% assign values to each month

monthMap = containers.Map({'jan', 'feb', 'march', 'oct', 'nov', 'dec'}, {'01', '02', '03', '04', '05', '06'});

monthNum = monthMap(monthStr);

% combine year and month

fileSuffix = strcat(yearStr, monthNum);

% reading and resizing the image

img = imread(fullFilePath);

img = imresize(img, [256 NaN]);

% generating panchromatic images

input = img(:,:,1:3);

try

input1 = img(:,:,4);

catch ex

input1 = rgb2gray(input);

end

% fuse both the images

MS = double(input);

PAN = double(input1);

% use a scale up factor to enhance the brightness of the images

k = 4;

% Fused image generating by enhancing green band

FusedImage = zeros(size(input));

for r = 1:size(input, 1)

for c = 1:size(input, 2)

red = MS(r, c, 1);

green = MS(r, c, 2);

blue = MS(r, c, 3);

shape = PAN(r, c);

% apply brovey transform formula

fused\_r = k \* shape \* red / (red + green + blue);

fused\_g = k \* shape \* green / (red + green + blue);

fused\_b = k \* shape \* blue / (red + green + blue);

FusedImage(r, c, 1) = fused\_r;

FusedImage(r, c, 2) = fused\_g;

FusedImage(r, c, 3) = fused\_b;

end

end

% convert the images to unit8 datatype

img\_out = uint8(FusedImage);

% imge plotting for Original image, Pan image, fused image

subplot(2, 2, 1);

imshow(uint8(MS));

title('Multispectral Color Information');

subplot(2, 2, 2);

imshow(uint8(PAN));

title('Panchromatic Shape Information');

subplot(2, 2, [3 4]);

imshow(img\_out);

title('Brovey Fusion');

% saving images

imwrite(uint8(MS), fullfile(outputDir, strcat('MS\_Image\_', fileSuffix, '.jpg')), 'jpg');

imwrite(uint8(PAN), fullfile(outputDir, strcat('PAN\_Image\_', fileSuffix, '.jpg')), 'jpg');

imwrite(img\_out, fullfile(outputDir, strcat('Fused\_Image\_', fileSuffix, '.jpg')), 'jpg');

% segmentation using ratio based thresholding

fused\_img = uint8(FusedImage);

threshold\_ratio\_gr = 1.2; % Ratio of green to red

threshold\_ratio\_gb = 1.2; % Ratio of green to blue

seg\_img = zeros(size(fused\_img), 'like', fused\_img);

veg\_area = 0;

for row = 1:size(fused\_img, 1)

for col = 1:size(fused\_img, 2)

red = double(fused\_img(row, col, 1));

green = double(fused\_img(row, col, 2));

blue = double(fused\_img(row, col, 3));

gr = green / red;

gb = green / blue;

if (gr > threshold\_ratio\_gr && gb > threshold\_ratio\_gb)

seg\_img(row, col, :) = fused\_img(row, col, :);

veg\_area = veg\_area + 1;

end

end

end

veg\_area = veg\_area / (size(fused\_img, 1) \* size(fused\_img, 2)) \* 100;

imwrite(uint8(seg\_img), fullfile(outputDir, strcat('Segmented\_Image\_', fileSuffix, '.jpg')), 'jpg'); % Save the segmented image

% plot segmented image

figure;

subplot(2, 1, 1);

imshow(fused\_img);

title('Fused Image');

subplot(2, 1, 2);

imshow(uint8(seg\_img));

title('Segmented Image');

% saving to databse

try

load('database', 'forest\_covers', 'time\_tags');

catch ex

forest\_covers = [];

time\_tags = [];

save('database', 'forest\_covers', 'time\_tags');

end

str = sprintf('Total forest area %0.04f %%\nEnter time tag', veg\_area);

time\_tag = str2num(cell2mat(inputdlg(str)));

db\_count = length(time\_tags) + 1;

forest\_covers(db\_count) = veg\_area;

time\_tags(db\_count) = time\_tag;

save('database', 'forest\_covers', 'time\_tags');

ch = menu(sprintf('Database saved with %d entries\nSelect an option', db\_count), 'Add more', 'Clear DB', 'Generate CSV File', 'Exit');

if ch == 1

Module3\_DataCreation;

elseif ch == 2

ch = menu('Are you sure', 'Yes', 'No');

if ch == 1

forest\_covers = [];

time\_tags = [];

save('database', 'forest\_covers', 'time\_tags');

menu('Database cleared', 'Ok');

end

elseif ch == 3

file\_name = 'final.csv';

csvwrite(file\_name, [forest\_covers', time\_tags']);

menu(sprintf('File saved at %s', file\_name), 'Ok');

end

**Code 2: Pearson correlation**

% Define paths for the original and fused images

baseDir = 'C:\Users\Dimple\Downloads\Problem\_Statement\Problem Statement\Dataset'; % Update this path as needed

outputDir = fullfile(baseDir, 'Output'); % Directory where fused images are saved

% Define file names

originalImageFileName = 'dec\_2023.jpg'; % original image

fusedImageFileName = 'Fused\_Image\_202306.jpg'; % fused image of the original image

% loading these images

originalImage = imread(fullfile(baseDir, originalImageFileName));

fusedImage = imread(fullfile(outputDir, fusedImageFileName));

% generating pan images

if size(originalImage, 3) == 3

originalImageGray = rgb2gray(originalImage);

else

originalImageGray = originalImage;

end

if size(fusedImage, 3) == 3

fusedImageGray = rgb2gray(fusedImage);

else

fusedImageGray = fusedImage;

end

% resizing images

if size(originalImageGray) ~= size(fusedImageGray)

fusedImageGray = imresize(fusedImageGray, size(originalImageGray));

end

% convert to double datatype

originalImageVec = double(originalImageGray(:));

fusedImageVec = double(fusedImageGray(:));

% Pearson correlation calculation

correlationCoefficient = corr(originalImageVec, fusedImageVec);

% print the results

fprintf('Pearson correlation coefficient between %s and %s: %0.4f\n', ...

originalImageFileName, fusedImageFileName, correlationCoefficient);

**Code 3: Optimal Threshold**

% copy same code from above

clc;

clear all;

close all;

baseDir = 'C:\Users\Dimple\Downloads\Problem\_Statement\Problem Statement\Dataset';

outputDir = fullfile(baseDir, 'Output');

if ~exist(outputDir, 'dir')

mkdir(outputDir);

end

fileName = 'dec\_2023.jpg';

fullFilePath = fullfile(baseDir, fileName);

[~, name, ~] = fileparts(fileName);

dateParts = regexp(name, '(\w+)\_(\d{4})', 'tokens');

monthStr = lower(dateParts{1}{1});

yearStr = dateParts{1}{2};

monthMap = containers.Map({'jan', 'feb', 'march', 'oct', 'nov', 'dec'}, {'01', '02', '03', '04', '05', '06'});

monthNum = monthMap(monthStr);

fileSuffix = strcat(yearStr, monthNum);

img = imread(fullFilePath);

img = imresize(img, [256 NaN]);

input = img(:,:,1:3);

try

input1 = img(:,:,4);

catch ex

input1 = rgb2gray(input);

end

MS = double(input);

PAN = double(input1);

k = 4;

FusedImage = zeros(size(input));

for r = 1:size(input, 1)

for c = 1:size(input, 2)

red = MS(r, c, 1);

green = MS(r, c, 2);

blue = MS(r, c, 3);

shape = PAN(r, c);

fused\_r = k \* shape \* red / (red + green + blue);

fused\_g = k \* shape \* green / (red + green + blue);

fused\_b = k \* shape \* blue / (red + green + blue);

FusedImage(r, c, 1) = fused\_r;

FusedImage(r, c, 2) = fused\_g;

FusedImage(r, c, 3) = fused\_b;

end

end

img\_out = uint8(FusedImage);

figure;

subplot(2, 2, 1);

imshow(uint8(MS));

title('Multispectral Color Information');

subplot(2, 2, 2);

imshow(uint8(PAN));

title('Panchromatic Shape Information');

subplot(2, 2, [3 4]);

imshow(img\_out);

title('Brovey Fusion');

imwrite(uint8(MS), fullfile(outputDir, strcat('MS\_Image\_', fileSuffix, '.jpg')), 'jpg');

imwrite(uint8(PAN), fullfile(outputDir, strcat('PAN\_Image\_', fileSuffix, '.jpg')), 'jpg');

imwrite(img\_out, fullfile(outputDir, strcat('Fused\_Image\_', fileSuffix, '.jpg')), 'jpg');

R = FusedImage(:,:,1) ./ 255;

G = FusedImage(:,:,2) ./ 255;

B = FusedImage(:,:,3) ./ 255;

r = R ./ (R + G + B);

g = G ./ (R + G + B);

b = B ./ (R + G + B);

% Now, start by initialising base values as 0 for d

best\_d = 0;

max\_separation = 0;

%printing figure for manual inspection

figure;

for d = 0.01:0.01:2

% Calculate ECI

eci = (r - 1).^2 + (g.^2 / d^2);

%apply threshold to separate vegetation and non-vegetation areas

threshold = 0.5;

seg\_img = eci < threshold;

% display the segmented image for manual inspection

imshow(seg\_img);

title(['Segmented Image with d = ', num2str(d)]);

drawnow;

pause(0.5); % Pause for 0.5 seconds for each iteration

veg\_area = sum(seg\_img(:)) / numel(seg\_img) \* 100;

if veg\_area > max\_separation

max\_separation = veg\_area;

best\_d = d;

end

end

% print the optimal value

disp(['Optimal d value: ', num2str(best\_d)]);

% segment using the threshold

eci = (r - 1).^2 + (g.^2 / best\_d^2);

seg\_img = eci < threshold;

%calculate vegetation area percentage

veg\_area = sum(seg\_img(:)) / numel(seg\_img) \* 100;

% save image after segmentation

imwrite(uint8(seg\_img \* 255), fullfile(outputDir, strcat('Segmented\_Image\_', fileSuffix, '.jpg')), 'jpg');

% plot the image

figure;

subplot(2, 1, 1);

imshow(img\_out);

title('Fused Image');

subplot(2, 1, 2);

imshow(uint8(seg\_img \* 255));

title('Segmented Image');

% save the values

try

load('database', 'forest\_covers', 'time\_tags');

catch

forest\_covers = [];

time\_tags = [];

save('database', 'forest\_covers', 'time\_tags');

end

%print vegetation area and time tag

str = sprintf('Total vegetation area: %0.04f %%\nEnter time tag:', veg\_area);

time\_tag = str2num(cell2mat(inputdlg(str)));

% update the values and save them

db\_count = length(time\_tags) + 1;

forest\_covers(db\_count) = veg\_area;

time\_tags(db\_count) = time\_tag;

save('database', 'forest\_covers', 'time\_tags');

% menu for different actions

ch = menu(sprintf('Database saved with %d entries\nSelect an option', db\_count), 'Add more', 'Clear DB', 'Generate CSV File', 'Exit');

if ch == 1

Module3\_DataCreation;

elseif ch == 2

ch = menu('Are you sure?', 'Yes', 'No');

if ch == 1

forest\_covers = [];

time\_tags = [];

save('database', 'forest\_covers', 'time\_tags');

menu('Database cleared', 'Ok');

end

elseif ch == 3

file\_name = 'final.csv';

csvwrite(file\_name, [forest\_covers', time\_tags']);

menu(sprintf('File saved at %s', file\_name), 'Ok');

end

**Code 4: Graphs**

clc;

clear;

close all;

% load the csv file

data = readtable('final.csv', 'VariableNamingRule', 'preserve');

% get the columns forest cover and timestamp

forest\_cover = data{:, 'Forest Cover'};

timestamps = data{:, 'Timestamp'};

% get year and month separately

years = floor(timestamps / 100);

months = mod(timestamps, 100);

% get data for each year

year\_2010\_idx = (years == 2010);

forest\_cover\_2010 = forest\_cover(year\_2010\_idx);

months\_2010 = months(year\_2010\_idx);

% map the month numbers to the names

monthNames = {'Jan', 'Feb', 'Mar', 'Oct', 'Nov', 'Dec'};

monthNumbers = [1, 2, 3, 4, 5, 6];

% plot figure

figure;

plot(monthNumbers, forest\_cover\_2010, '-o', 'LineWidth', 2);

set(gca, 'XTick', monthNumbers, 'XTickLabel', monthNames);

xlabel('Month');

ylabel('Forest Cover');

title('Forest Cover for Year 2010');

grid on;

%save file

saveas(gcf, 'Forest\_Cover\_2010.png');

clc;

clear;

close all;

% load the csv file

data = readtable('final.csv', 'VariableNamingRule', 'preserve');

%get columns forest cover and timestamp

forest\_cover = data{:, 'Forest Cover'};

timestamps = data{:, 'Timestamp'};

% get year and month separately

years = floor(timestamps / 100);

months = mod(timestamps, 100);

% Unique years

unique\_years = unique(years);

% average forest cover calculation for every year

avg\_forest\_cover\_per\_year = arrayfun(@(y) mean(forest\_cover(years == y)), unique\_years);

% plot for year vs FC

figure;

plot(unique\_years, avg\_forest\_cover\_per\_year, '-o', 'LineWidth', 2);

xlabel('Year');

ylabel('Average Forest Cover');

title('Year vs Average Forest Cover');

grid on;

% Save plot

saveas(gcf, 'Year\_vs\_Average\_Forest\_Cover.png');

monthNames = {'Jan', 'Feb', 'Mar', 'Oct', 'Nov', 'Dec'};

valid\_months = [1, 2, 3, 4, 5, 6];

% get filtered months

valid\_idx = ismember(months, valid\_months);

filtered\_months = months(valid\_idx);

filtered\_forest\_cover = forest\_cover(valid\_idx);

% plot month vs FC

figure;

boxplot(filtered\_forest\_cover, filtered\_months, 'Labels', monthNames);

xlabel('Month');

ylabel('Forest Cover');

title('Box Plot of Forest Cover by Month');

grid on;

% Save plot

saveas(gcf, 'Month\_vs\_Forest\_Cover\_BoxPlot.png');

**Code 5: Correlation Matrix**

% load climatedata.csv file and read it

file\_path = 'C:\Users\Dimple\OneDrive\Desktop\Climatedata.csv';

climateData = readtable(file\_path);

% get all columns

temperature = climateData.Temperature;

dewPoint = climateData.Dewpoint;

wind = climateData.Wind;

seaLevelPressure = climateData.SeaLevelPressure;

forestCover = climateData.Forestcover;

% put all the columns together

climatic\_data = [temperature, dewPoint, wind, seaLevelPressure, forestCover];

% correlation matrix calculation

correlation\_matrix = corr(climatic\_data);

% print the matrix

disp('Correlation Matrix:');

disp(correlation\_matrix);

% plot

figure;

heatmap({'Temperature', 'Dew Point', 'Wind', 'Sea Level Pressure', 'Forest Cover'}, ...

{'Temperature', 'Dew Point', 'Wind', 'Sea Level Pressure', 'Forest Cover'}, ...

correlation\_matrix, 'Colormap', jet, 'ColorbarVisible', 'on');

title('Correlation Matrix: Climatic Conditions and Forest Cover');

% scatter plot

figure;

% temp vs FC

subplot(2,2,1);

scatter(temperature, forestCover, 50, 'x', 'MarkerEdgeColor', 'blue');

title('Temperature vs Forest Cover');

xlabel('Temperature');

ylabel('Forest Cover');

grid on;

% Dp vs FC

subplot(2,2,2);

scatter(dewPoint, forestCover, 50, 'x', 'MarkerEdgeColor', 'green');

title('Dew Point vs Forest Cover');

xlabel('Dew Point');

ylabel('Forest Cover');

grid on;

% Wind vs FC

subplot(2,2,3);

scatter(wind, forestCover, 50, 'x', 'MarkerEdgeColor', 'red');

title('Wind vs Forest Cover');

xlabel('Wind');

ylabel('Forest Cover');

grid on;

% SLP vs Fc

subplot(2,2,4);

scatter(seaLevelPressure, forestCover, 50, 'x', 'MarkerEdgeColor', 'magenta');

title('Sea Level Pressure vs Forest Cover');

xlabel('Sea Level Pressure');

ylabel('Forest Cover');

grid on;

% Adjust figure layout

sgtitle('Scatter Plots: Climatic Conditions vs Forest Cover');

**Code 6 : Coefficient values For LR**

% load climate data

climate\_data = readtable('C:\Users\Dimple\OneDrive\Desktop\Climatedata.csv');

% get input and target columns

dew\_point = climate\_data.Dewpoint;

wind = climate\_data.Wind;

forest\_cover = climate\_data.Forestcover;

% normalize input values

dew\_point\_normalized = (dew\_point - mean(dew\_point)) / std(dew\_point);

wind\_normalized = (wind - mean(wind)) / std(wind);

% design matrix (independent variables)

% Adding column of 1 to X for the intercept

X = [ones(length(dew\_point), 1), dew\_point\_normalized, wind\_normalized];

% Multiple linear reg model

coefficients = (X' \* X) \ (X' \* forest\_cover); % coeff calculation

% get coefficients

intercept = coefficients(1); % Intercept (beta\_0)

dew\_point\_coefficient = coefficients(2); % Dew Point coefficient (beta\_1)

wind\_coefficient = coefficients(3); % Wind coefficient (beta\_2)

% print coefficients

fprintf('Intercept (beta\_0): %.4f\n', intercept);

fprintf('Dew Point Coefficient (beta\_1): %.4f\n', dew\_point\_coefficient);

fprintf('Wind Coefficient (beta\_2): %.4f\n', wind\_coefficient);

% predict forest cover

forest\_cover\_pred = X \* coefficients;

% calculate evaluation metric

rmse = sqrt(mean((forest\_cover - forest\_cover\_pred).^2));

mse = mean((forest\_cover - forest\_cover\_pred).^2);

mae = mean(abs(forest\_cover - forest\_cover\_pred));

SStot = sum((forest\_cover - mean(forest\_cover)).^2);

SSres = sum((forest\_cover - forest\_cover\_pred).^2);

rsquared = 1 - SSres / SStot;

% print evaluation metrics

fprintf('Root Mean Squared Error (RMSE): %.4f\n', rmse);

fprintf('Mean Squared Error (MSE): %.4f\n', mse);

fprintf('Mean Absolute Error (MAE): %.4f\n', mae);

fprintf('R-squared: %.4f\n', rsquared);

% plot actual vs predicted FC

figure;

plot(1:length(forest\_cover), forest\_cover, 'bo-', 'LineWidth', 2); hold on;

plot(1:length(forest\_cover\_pred), forest\_cover\_pred, 'go-', 'LineWidth', 2);

title('Actual vs. Predicted Forest Cover (LR Model)', 'FontSize', 14);

xlabel('Observation Index', 'FontSize', 12);

ylabel('Forest Cover (%)', 'FontSize', 12);

legend({'Actual Forest Cover', 'Predicted Forest Cover'}, 'Location', 'Best');

grid on;

**Code 7 – SVR**

% load climate data

climate\_data = readtable('C:\Users\Dimple\OneDrive\Desktop\Climatedata.csv');

% get input and target features

dew\_point = climate\_data.Dewpoint;

wind = climate\_data.Wind;

forest\_cover = climate\_data.Forestcover;

% normalize input values

dew\_point\_normalized = (dew\_point - mean(dew\_point)) / std(dew\_point);

wind\_normalized = (wind - mean(wind)) / std(wind);

% design matrix (independent variables)

X = [dew\_point\_normalized, wind\_normalized];

% cross-validation and hyperparameter tuning for SVR

C\_values = [0.1, 1, 10, 100];

epsilon\_values = [0.01, 0.1, 1];

% initialize variable to store best results

best\_model = [];

best\_rmse = Inf;

% code to get the best parameter

for C = C\_values

for epsilon = epsilon\_values

% SVR model training using hypeparameter values

MdlSVR = fitrsvm(X, forest\_cover, 'KernelFunction', 'gaussian', ...

'BoxConstraint', C, 'Epsilon', epsilon, 'Standardize', true);

% k-fold cross-validation k = 5

CVModel = crossval(MdlSVR, 'KFold', 5);

% RMSE calculation for cross validation

kfold\_loss = kfoldLoss(CVModel);

rmse = sqrt(kfold\_loss);

% best model check

if rmse < best\_rmse

best\_rmse = rmse;

best\_model = MdlSVR;

end

fprintf('C: %.2f, epsilon: %.2f, Cross-validated RMSE: %.4f\n', C, epsilon, rmse);

end

end

% prediction using best parameters

forest\_cover\_pred = predict(best\_model, X);

% eval metrics

rmse\_svr = sqrt(mean((forest\_cover - forest\_cover\_pred).^2));

mse\_svr = mean((forest\_cover - forest\_cover\_pred).^2);

mae\_svr = mean(abs(forest\_cover - forest\_cover\_pred));

SStot = sum((forest\_cover - mean(forest\_cover)).^2);

SSres = sum((forest\_cover - forest\_cover\_pred).^2);

rsquared\_svr = 1 - SSres / SStot;

% print eval metrics

fprintf('Best SVR Model - RMSE: %.4f\n', rmse\_svr);

fprintf('Best SVR Model - MSE: %.4f\n', mse\_svr);

fprintf('Best SVR Model - MAE: %.4f\n', mae\_svr);

fprintf('Best SVR Model - R-squared: %.4f\n', rsquared\_svr);

% plot actual vs pred

figure;

plot(1:length(forest\_cover), forest\_cover, 'bo-', 'LineWidth', 2); hold on;

plot(1:length(forest\_cover\_pred), forest\_cover\_pred, 'go-', 'LineWidth', 2);

title('Actual vs. Predicted Forest Cover (SVR Model)', 'FontSize', 14);

xlabel('Observation Index', 'FontSize', 12);

ylabel('Forest Cover (%)', 'FontSize', 12);

legend({'Actual Forest Cover', 'Predicted Forest Cover'}, 'Location', 'Best');

grid on;