% Load dataset (player\_performance\_dataset\_500)

load fisheriris

X = meas;

Y = species;

% Split the dataset into training (70%) and testing (30%) sets

cv = cvpartition(size(X, 1), 'HoldOut', 0.3);

XTrain = X(training(cv), :);

YTrain = Y(training(cv));

XTest = X(test(cv), :);

YTest = Y(test(cv));

% Convert labels to numeric format for neural network

YTrainNN = dummyvar(categorical(YTrain))'; % One-hot encoding for neural network

YTestNN = dummyvar(categorical(YTest))';

%% Neural Network Training

% Create and configure the neural network

net = patternnet(10); % 10 hidden neurons

net.trainParam.epochs = 30; % Set the number of epochs to 30

net.trainParam.showWindow = false; % Disable GUI training window

% Train the neural network

[net, tr] = train(net, XTrain', YTrainNN);

% Calculate training and validation accuracy per epoch

trainAccuracyNN = 1 - tr.perf;

valAccuracyNN = 1 - tr.vperf;

% Calculate training and validation loss per epoch

trainLossNN = tr.perf;

valLossNN = tr.vperf;

% Plot accuracy vs epochs for neural network

figure;

plot(tr.epoch, trainAccuracyNN, '-o');

hold on;

plot(tr.epoch, valAccuracyNN, '-x');

hold off;

xlabel('Epochs');

ylabel('Accuracy');

legend('Training Accuracy (NN)', 'Validation Accuracy (NN)');

title('Neural Network Training and Validation Accuracy vs. Epochs');

% Plot loss vs epochs for neural network

figure;

plot(tr.epoch, trainLossNN, '-o');

hold on;

plot(tr.epoch, valLossNN, '-x');

hold off;

xlabel('Epochs');

ylabel('Loss');

legend('Training Loss (NN)', 'Validation Loss (NN)');

title('Neural Network Training and Validation Loss vs. Epochs');

% Predict on testing data using neural network

YPredNN = net(XTest');

[~, YPredNN] = max(YPredNN); % Convert to class labels

[~, YTestLabelsNN] = max(YTestNN);

% Calculate and display overall accuracy on the testing set for neural network

accuracyNN = sum(YPredNN == YTestLabelsNN) / numel(YTestLabelsNN);

fprintf('Testing Accuracy (Neural Network): %.2f%%\n', accuracyNN \* 100);

% Plot confusion matrix for Neural Network

figure;

confMatNN = confusionmat(YTestLabelsNN, YPredNN);

confusionchart(confMatNN);

title('Confusion Matrix for Testing Set (Neural Network)');

% Display classification report for Neural Network

[precisionNN, recallNN, f1scoreNN, supportNN] = precisionRecallF1(YTestLabelsNN, YPredNN);

displayClassificationReport('Neural Network', precisionNN, recallNN, f1scoreNN, supportNN);

%% SVM Training

% Train SVM on the same training data

SVMModel = fitcecoc(XTrain, YTrain);

% Initialize arrays to store metrics for each epoch

numEpochs = tr.epoch(end);

epochPrecisionSVM = zeros(numEpochs, numel(unique(YTest)));

epochRecallSVM = zeros(numEpochs, numel(unique(YTest)));

epochF1ScoreSVM = zeros(numEpochs, numel(unique(YTest)));

fprintf('\nSVM Classification Report for Each Epoch:\n');

fprintf('%-10s %-10s %-10s %-10s %-10s\n', 'Epoch', 'Class', 'Precision', 'Recall', 'F1-Score');

for e = 1:numEpochs

% Predict on the testing data using SVM

YPredSVM = predict(SVMModel, XTest);

YTestNumeric = grp2idx(YTest); % Convert cell array to numeric

YPredSVMNumeric = grp2idx(YPredSVM); % Convert cell array to numeric

% Calculate precision, recall, F1 score for the current epoch

[precisionSVM, recallSVM, f1scoreSVM, supportSVM] = precisionRecallF1(YTestNumeric, YPredSVMNumeric);

epochPrecisionSVM(e, :) = precisionSVM;

epochRecallSVM(e, :) = recallSVM;

epochF1ScoreSVM(e, :) = f1scoreSVM;

% Print epoch metrics to command window

for i = 1:numel(precisionSVM)

fprintf('%-10d %-10d %-10.2f %-10.2f %-10.2f\n', e, i, precisionSVM(i), recallSVM(i), f1scoreSVM(i));

end

end

% Calculate and display overall accuracy on the testing set for SVM

accuracySVM = sum(strcmp(YPredSVM, YTest)) / numel(YTest);

fprintf('Testing Accuracy (SVM): %.2f%%\n', accuracySVM \* 100);

% Plot confusion matrix for SVM

figure;

confMatSVM = confusionmat(YTest, YPredSVM);

confusionchart(confMatSVM);

title('Confusion Matrix for Testing Set (SVM)');

% Display final classification report for SVM

[precisionSVM, recallSVM, f1scoreSVM, supportSVM] = precisionRecallF1(YTestNumeric, YPredSVMNumeric);

displayClassificationReport('SVM', precisionSVM, recallSVM, f1scoreSVM, supportSVM);

% Helper function to calculate precision, recall, and F1 score

function [precision, recall, f1score, support] = precisionRecallF1(yTrue, yPred)

classes = unique(yTrue);

numClasses = numel(classes);

precision = zeros(1, numClasses);

recall = zeros(1, numClasses);

f1score = zeros(1, numClasses);

support = zeros(1, numClasses);

for i = 1:numClasses

class = classes(i);

tp = sum((yPred == class) & (yTrue == class));

fp = sum((yPred == class) & (yTrue ~= class));

fn = sum((yPred ~= class) & (yTrue == class));

precision(i) = tp / (tp + fp);

recall(i) = tp / (tp + fn);

f1score(i) = 2 \* (precision(i) \* recall(i)) / (precision(i) + recall(i));

support(i) = sum(yTrue == class);

end

end

% Helper function to display classification report with proper alignment

function displayClassificationReport(method, precision, recall, f1score, support)

fprintf('\n%s Classification Report:\n', method);

fprintf('%-10s %-10s %-10s %-10s %-10s\n', 'Class', 'Precision', 'Recall', 'F1-Score', 'Support');

for i = 1:numel(precision)

fprintf('%-10d %-10.2f %-10.2f %-10.2f %-10d\n', i, precision(i), recall(i), f1score(i), support(i));

end

overallAccuracy = mean(precision);

macroAverage = mean(precision);

weightedAverage = mean(f1score);

fprintf('\nOverall Accuracy: %.2f%%\n', overallAccuracy \* 100);

fprintf('Macro Average: %.2f\n', macroAverage);

fprintf('Weighted Average: %.2f\n\n', weightedAverage);

end