```
% Loading data
clc
clear all
close all
warning off
data=readtable('diabetes.csv');
%%
%Checking null values
a=sum(ismissing(data));
%%



Pregnancies= data.Pregnancies;
Glucose= data.Glucose;
BloodPressure = data.BloodPressure;
SkinThickness = data.SkinThickness;
Insulin = data.Insulin;
BMI = data.BMI;
DiabetesPedigreeFunction = data.DiabetesPedigreeFunction;
Age = data.Age;
Outcome = data.Outcome;

%Handling the 0 values for some features of data

k=(BloodPressure==0);
l=(SkinThickness==0);
m=(Insulin==0);
n=(BMI==0);
o=(DiabetesPedigreeFunction==0);
p=(Age==0);

%Replacing those with average of the corresponding columns of data
g=(Glucose==0);
msa=(Outcome==1);
ks=mean(Glucose(~g & msa));
Glucose(g & msa)=ks;
ks=mean(Glucose(~g & ~msa));
Glucose(g & ~msa)=ks;

g=(BloodPressure==0);
ks=mean(BloodPressure(~g & msa));
BloodPressure(g & msa)=ks;
ks=mean(BloodPressure(~g & ~msa));
BloodPressure(g & ~msa)=ks;

g=(SkinThickness==0);
ks=mean(SkinThickness(~g & msa));
SkinThickness(g & msa)=ks;
ks=mean(SkinThickness(~g & ~msa));
SkinThickness(g & ~msa)=ks;

g=(Insulin==0);
ks=mean(Insulin(~g & msa));
Insulin(g & msa)=ks;
ks=mean(Insulin(~g & ~msa));
Insulin(g & ~msa)=ks;

g=(BMI==0);
ks=mean(BMI(~g & msa));
BMI(g & msa)=ks;
ks=mean(BMI(~g & ~msa));
BMI(g & ~msa)=ks;
```

```
g=(DiabetesPedigreeFunction==0);
ks=mean(DiabetesPedigreeFunction(~g & msa));
DiabetesPedigreeFunction(g & msa)=ks;
ks=mean(DiabetesPedigreeFunction(~g & ~msa));
DiabetesPedigreeFunction(g & ~msa)=ks;

Age(p)=mean(Age(~p));
%%

% Visualization
histogram(Glucose, 'BinWidth', 10, 'EdgeColor', 'black', 'FaceColor', 'blue');
title('Glucose Levels');
xlabel('Glucose Levels');
ylabel('Frequency');

% Visualization 2
subplot(2, 2, 1);
boxplot(Pregnancies, 'Labels', {'Pregnancies'});
title(' Pregnancies');

subplot(2, 2, 2);
boxplot(BloodPressure, 'Labels', {'Blood Pressure'});
title(' Blood Pressure');

subplot(2, 2, 3);
boxplot(BMI, 'Labels', {'BMI'});
title(' BMI');

subplot(2, 2, 4);
boxplot(Age, 'Labels', {'Age'});
title('Age');

%vis 3
scatter(BloodPressure, Age, 'Marker', 'o', 'MarkerEdgeColor', 'red');
title('bloodPressure vs. Age');
xlabel('bloodPressure');
ylabel('Age');


% vis 4 reference(https://uk.mathworks.com/help/matlab/ref/hist.html)
% Count the occurrences of each class
classCounts = [sum(Outcome == 0), sum(Outcome == 1)];

% Creating bar plot
bar(classCounts);
title('Distribution of Diabetes');
xlabel('Outcome (0: No Diabetes, 1: Diabetes)');
ylabel('Count');

text(1:length(classCounts), classCounts, num2str(classCounts'), ...
    'HorizontalAlignment','center', 'VerticalAlignment','bottom');

% Showing the legend
legend('Outcome');

% Set the axis limits
ylim([0, max(classCounts) + 1]);

% Display the grid
grid on;
```

```
% Min-Max Scaling {(value-min(value from row))/(max(value)-min(value))}
% reference: (https://uk.mathworks.com/help/matlab/ref/rescale.html)
Pregnancies =(Pregnancies-min(Pregnancies))/(max(Pregnancies)-min(Pregnancies));
Glucose=(Glucose-min(Glucose))/(max(Glucose)-min(Glucose));
BloodPressure=(BloodPressure-min(data.BloodPressure))/(max(BloodPressure)-
min(BloodPressure));
SkinThickness=(SkinThickness-min(SkinThickness))/(max(SkinThickness)-
min(SkinThickness));
Insulin=(Insulin-min(Insulin))/(max(Insulin)-min(Insulin));
BMI=(BMI-min(data.BMI))/(max(BMI)-min(BMI));
DiabetesPedigreeFunction=(DiabetesPedigreeFunction-
min(DiabetesPedigreeFunction))/(max(DiabetesPedigreeFunction)-
min(DiabetesPedigreeFunction));
Age=(Age-min(data.Age))/(max(Age)-min(Age));
%%


% Spliting the data into training and testing sets
cv = cvpartition(height(data), 'HoldOut', 0.2);
idxTrain = training(cv);
idxTest = ~idxTrain;

% predictors
predictors = {'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'};

% Separate predictors and response variable
XTrain = data{idxTrain, predictors};
YTrain = data{idxTrain, 'Outcome'};
XTest = data{idxTest, predictors};
YTest = data{idxTest, 'Outcome'};

YeTest = [XTest, YTest];


% hyperparameter tuning for Decision Tree
dtHyperparams = hyperparameters('fitctree', XTrain, YTrain);

% Training Decision Tree (DT) model with hyperparameter tuning
rng(42)
dtModel = fitctree(XTrain, YTrain, 'OptimizeHyperparameters', 'all');

% predictions on the test set
dtPredictions = predict(dtModel, XTest);

% performance metrics for Decision Tree
dtAccuracy = sum(dtPredictions == YTest) / numel(YTest);
fprintf('Decision Tree Accuracy: %.4f\n', dtAccuracy);


X = data{:, predictors};
Y = data{:, 'Outcome'};

% 10-fold cross-validation for Decision Tree
%reference(https://youtu.be/OCKdOXjzzSU?si=xVBhHgFWEht_981x)
cvDt = cvpartition(Y, 'KFold', 10);
dtAccuracy = zeros(cvDt.NumTestSets, 1);

for k = 1:cvDt.NumTestSets
    idxTrain = training(cvDt, k);
    idxTest = test(cvDt, k);

    XTrain = X(idxTrain, :);
```

```
    YTrain = Y(idxTrain);
    XTest = X(idxTest, :);
    YTest = Y(idxTest);

    % Train Decision Tree model
    dtModel = fitctree(XTrain, YTrain);

    % Make predictions on the test set
    dtPredictions = predict(dtModel, XTest);

    % Calculate accuracy for this fold
    dtAccuracy(k) = sum(dtPredictions == YTest) / numel(YTest);
end

% average accuracy for Decision Tree
fprintf('Average Accuracy (Decision Tree): %.4f\n', mean(dtAccuracy));

% confusion matrix
confMat1 = confusionmat(YTest, dtPredictions);

% Displaying the confusion matrix
disp('Confusion Matrix:');
disp(confMat1);

% Calculating precision, recall, and F1 score
precision = confMat1(2,2) / sum(confMat1(:,2));
recall = confMat1(2,2) / sum(confMat1(2,:));
f1Score = 2 * (precision * recall) / (precision + recall);

% Printing precision, recall, and F1 score
fprintf('Precision: %.4f\n', precision);
fprintf('Recall: %.4f\n', recall);
fprintf('F1 Score: %.4f\n', f1Score);

% Plot the confusion matrix as a heatmap
imagesc(confMat1);

% Add labels and title
xlabel('Predicted');
ylabel('True');
title('Confusion Matrix');

% Display the color scale
colorbar;

% Show the values inside the cells
textStrings = num2str(confMat1(:), '%d');
textStrings = strtrim(cellstr(textStrings));
[x, y] = meshgrid(1:size(confMat1, 2), 1:size(confMat1, 1));
hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', 'center',
'FontSize', 12, 'FontWeight', 'bold');

% Highlight the diagonal (correctly predicted) cells
idx = sub2ind(size(confMat1), 1:size(confMat1, 1), 1:size(confMat1, 2));
set(hStrings(idx), 'Color', 'w');


% Training Naive Bayes model with hyperparameter tuning
rng(42)
nbModel = fitcnb(XTrain, YTrain, 'OptimizeHyperparameters', 'all');

% predictions on the test set
nbPredictions = predict(nbModel, XTest);
```

```
% performance metrics for Naive Bayes
nbAccuracy = sum(nbPredictions == YTest) / numel(YTest);
fprintf('Naive Bayes Accuracy: %.4f\n', nbAccuracy);

% 10-fold cross-validation for Naive Bayes
cvNb = cvpartition(Y, 'KFold', 10);
nbAccuracy = zeros(cvNb.NumTestSets, 1);

for k = 1:cvNb.NumTestSets
    idxTrain = training(cvNb, k);
    idxTest = test(cvNb, k);

    XTrain = X(idxTrain, :);
    YTrain = Y(idxTrain);
    XTest = X(idxTest, :);
    YTest = Y(idxTest);

    % Training Naive Bayes model
    nbModel = fitcnb(XTrain, YTrain);

    % predictions on the test set
    nbPredictions = predict(nbModel, XTest);

    % accuracy for this fold
    nbAccuracy(k) = sum(nbPredictions == YTest) / numel(YTest);
end

% Printing average accuracy for Naive Bayes
fprintf('Average Accuracy (Naive Bayes): %.4f\n', mean(nbAccuracy));

% Creating confusion matrix for nb
confMat = confusionmat(YTest, nbPredictions);

% Displaying the confusion matrix
disp('Confusion Matrix:');
disp(confMat);

% Calculating precision, recall, and F1 score
precision = confMat(2,2) / sum(confMat(:,2));
recall = confMat(2,2) / sum(confMat(2,:));
f1Score = 2 * (precision * recall) / (precision + recall);

% Printing precision, recall, and F1 score
fprintf('Precision: %.4f\n', precision);
fprintf('Recall: %.4f\n', recall);
fprintf('F1 Score: %.4f\n', f1Score);

% Plot the confusion matrix as a heatmap
imagesc(confMat);

% Add labels and title
xlabel('Predicted');
ylabel('True');
title('Confusion Matrix');

% Display the color scale
colorbar;

% Show the values inside the cells
textStrings = num2str(confMat(:), '%d');
textStrings = strtrim(cellstr(textStrings));
[x, y] = meshgrid(1:size(confMat, 2), 1:size(confMat, 1));
hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', 'center',
```

```
'FontSize', 12, 'FontWeight', 'bold');

% Highlight the diagonal (correctly predicted) cells
idx = sub2ind(size(confMat), 1:size(confMat, 1), 1:size(confMat, 2));
set(hStrings(idx), 'Color', 'w');


% Saving the models
save('dt_model.mat','dtModel');
save('nb_model.mat','nbModel');
% Convert the table to a dataset
YeTestDataset = dataset(YeTest);

% Saving the dataset to a CSV file
export(YeTestDataset, 'File', 'YeTest.csv', 'Delimiter', ',');



publish('ML_finalcode.m','pdf')
```