```
%%first of all lets reset our environment%%
%commands to clear the command window
%reference:https://uk.mathworks.com/help/matlab/ref/clc.html
clc;
%command to clear all the variables
%reference:https://uk.mathworks.com/help/matlab/ref/clear.html
clear;
%command to close any open figures
%referencehttps://uk.mathworks.com/help/matlab/ref/close.html
close all;

%%datasetlink from kaggle%%
%https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

%%now we have to read our dataset%%
%reference:https://uk.mathworks.com/help/matlab/ref/readtable.html
diabetesdataset=readtable('diabetes.csv');
% we used read table function and not read csv function mainly because our
% dataset also has column names and not ony numerical data

%%the next step is to display the first few columns of the dataset  just
%%like we learnt in the principles of data science module%%
%reference:https://uk.mathworks.com/help/matlab/ref/double.head.html
head(diabetesdataset)
%%lets now check for missing values like we know from principles of data
%%science module%%
%reference: https://uk.mathworks.com/help/matlab/ref/ismissing.html
ismissing(diabetesdataset)
%this function shows all 0s showing our data has no missing values however
%this does not make sense since when visualizing the dataset we saw that
%the 0s we have in each column act as placeholders.It is not realistic to
%have 0 BMI, insulin,skin thickness,blood pressure and glucose. We also
%know that the ismissing function does not treat 0 values as missing values

%%we will replace the 0 values with NaN not a number values in order for
%%the ismissing function to work%%.
%reference:https://uk.mathworks.com/matlabcentral/answers/6038-convert-zeros-to-nan
diabetesdataset.Pregnancies(diabetesdataset.Pregnancies==0) =NaN;
%we replaced the pregnancies column
diabetesdataset.Glucose(diabetesdataset.Glucose==0) =NaN;
%we replaced the glucose column
diabetesdataset.BloodPressure(diabetesdataset.BloodPressure==0) =NaN;
%we replaced the blood pressure column
diabetesdataset.SkinThickness(diabetesdataset.SkinThickness==0) =NaN;
%we replaced the skin thickness column
diabetesdataset.Insulin(diabetesdataset.Insulin==0) =NaN;
%we replaced the insulin column
diabetesdataset.BMI(diabetesdataset.BMI==0) =NaN;
%replacing the BMI column

%%now we can check if we replaced the 0 values with NaN%%
%reference:https://uk.mathworks.com/help/matlab/ref/ismissing.html
ismissing(diabetesdataset)
%the output now is 0 and 1 so we have successfully replaced the 0 values
%with NaN not a number
```

```
%%lets create a new dataframe that stores the NaN values%%
diabetesdataset2=diabetesdataset;

%%lets fill the NaN values with the median for each variable%%
%we will use the fillmissing function and we will specify that the method
%will be 'constant' so that we replace the missing values by a constant.
%The fill value will be the median.The option will be 'omitnan' so that we
%can ignore the missing NaN values when doing the median calculation
%reference:https://uk.mathworks.com/help/matlab/ref/fillmissing.html
diabetesdataset2.Pregnancies=fillmissing(diabetesdataset2.Pregnancies,'constant',medi
an(diabetesdataset2.Pregnancies,'omitnan'));
%filled the pregnancies column with the median
diabetesdataset2.Glucose=fillmissing(diabetesdataset2.Glucose,'constant',median(diabe
tesdataset2.Glucose,'omitnan'));
%filled the glucose column with the median
diabetesdataset2.BloodPressure=fillmissing(diabetesdataset2.BloodPressure,'constant',
median(diabetesdataset2.BloodPressure,'omitnan'));
%filled the blood pressure column with the median
diabetesdataset2.SkinThickness=fillmissing(diabetesdataset2.SkinThickness,'constant',
median(diabetesdataset2.SkinThickness,'omitnan'));
%filling the skin thickness column with the median
diabetesdataset2.Insulin=fillmissing(diabetesdataset2.Insulin,'constant',median(diabe
tesdataset2.Insulin,'omitnan'));
%filled the insulin column with the median
diabetesdataset2.BMI=fillmissing(diabetesdataset2.BMI,'constant',median(diabetesdatas
et2.BMI,'omitnan'));
%filled the BMI with the median

%%now we can check if we replaced the NaN values with the median%%
%reference:https://uk.mathworks.com/help/matlab/ref/ismissing.html
ismissing(diabetesdataset2)
% we can see that all the rows and columns consist of 0s so we succesfully
% replaced the NaN values with the median

%%now we have to check if our dataset has any outliers. We will do this
%%using a boxplot visualization for each variable%%
%reference:https://uk.mathworks.com/help/matlab/ref/boxchart.html
boxchart(diabetesdataset2.Pregnancies)
%the pregnancies column has 14 outliers
boxchart(diabetesdataset2.Glucose)
%the glucose column does not have any outliers
boxchart(diabetesdataset2.BloodPressure)
%the blood pressure column has 14 outliers
boxchart(diabetesdataset2.SkinThickness)
%the skin thickness column has 87 outliers
boxchart(diabetesdataset2.Insulin)
%the insulin column has 345 outliers
boxchart(diabetesdataset2.BMI)
%the BMI column has 8 outliers
boxchart(diabetesdataset2.DiabetesPedigreeFunction)
%the diabetes pedigree function column has 29 outliers
boxchart(diabetesdataset2.Age)
%the age column has 9 outliers
```

```
%%now we will remove the outliers using the rmoutliers function and save as
%%a new dataset
%reference:https://uk.mathworks.com/help/matlab/ref/rmoutliers.html
diabetesdataset3=rmoutliers(diabetesdataset2,"mean");
%in the above code we used the rmoutliers function and specified the method
%to be "mean" so that an outlier is detected only when it is 3 standard
%deviations away from the mean

%%now we have to normalize/standardise the columns. since we will start
%%with naive bayes algorithm we will do normalization since naive bayes
%%assumes normality%%. We wont normalize the pregnancies and outcome column
%%because it does not make sense since. Pregnancies is a count variable and
%%outcome is binary%%
%we can find the formula for normalization  in our week 2 lecture notes on
%page 8
%reference: https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
%lets nornalized the glucose column
diabetesdataset3.Glucose=(diabetesdataset3.Glucose-
min(diabetesdataset3.Glucose))/(max(diabetesdataset3.Glucose)-
min(diabetesdataset3.Glucose));
%displaying the normalized glucose column
diabetesdataset3.Glucose
%lets normalize the blood pressure column
diabetesdataset3.BloodPressure=(diabetesdataset3.BloodPressure-
min(diabetesdataset3.BloodPressure))/(max(diabetesdataset3.BloodPressure)-
min(diabetesdataset3.BloodPressure));
%displaying the normalized blood pressure column
diabetesdataset3.BloodPressure
%lets normalize skin thickness column
diabetesdataset3.SkinThickness=(diabetesdataset3.SkinThickness-
min(diabetesdataset3.SkinThickness))/(max(diabetesdataset3.SkinThickness)-
min(diabetesdataset3.SkinThickness));
%displaying the normalized skin thickness column
diabetesdataset3.SkinThickness
%lets normalize the insulin column
diabetesdataset3.Insulin=(diabetesdataset3.Insulin-
min(diabetesdataset3.Insulin))/(max(diabetesdataset3.Insulin)-
min(diabetesdataset3.Insulin));
%displaying the normalized insulin column
diabetesdataset3.Insulin
%lets normalize the BMI column
diabetesdataset3.BMI=(diabetesdataset3.BMI-
min(diabetesdataset3.BMI))/(max(diabetesdataset3.BMI)-min(diabetesdataset3.BMI));
%displaying the normalized BMI column
diabetesdataset3.BMI
%normalizing the diabetes pedigree function column
diabetesdataset3.DiabetesPedigreeFunction=(diabetesdataset3.DiabetesPedigreeFunction-
min(diabetesdataset3.DiabetesPedigreeFunction))/(max(diabetesdataset3.DiabetesPedigre
eFunction)-min(diabetesdataset3.DiabetesPedigreeFunction));
%displaying the normalized diabetes pedigree function
diabetesdataset3.DiabetesPedigreeFunction
%lets normalize the age column
diabetesdataset3.Age=(diabetesdataset3.Age-
min(diabetesdataset3.Age))/(max(diabetesdataset3.Age)-min(diabetesdataset3.Age));
%displaying the normalized age column
```

```
diabetesdataset3.Age

%%now lets save the normalized column in a new dataset%%
diabetesdataset4=diabetesdataset3;

%%lets check the summary statistics for our fully pre processed dataset%%
%reference:https://uk.mathworks.com/help/matlab/ref/table.summary.html
summary(diabetesdataset4)
%the summary function shows us the minimum, maximum and median values of
%each column. We can see that the minimum and maximum values for all the
%normalized columns as 0 and 1 as we expected

%lets calculate the mean of our dataset
%reference:https://uk.mathworks.com/help/matlab/ref/mean.html
meannormalized=mean(diabetesdataset4);

%lets calculate the median of our dataset
%reference:https://uk.mathworks.com/help/matlab/ref/median.html
mediannormalized=median(diabetesdataset4);

%lets calculate the standard deviation of our dataset
%reference: https://uk.mathworks.com/help/matlab/ref/std.html
standarddeviationnormalized=std(diabetesdataset4);

%%lets visualize each of our variables using a histogram with a title and
%labelled axis%%
%reference:
https://uk.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.ht
ml
%reference:https://uk.mathworks.com/help/matlab/titles-and-labels.html
%histogram of pregnancies column with title and labelled axis
histogram(diabetesdataset4.Pregnancies)
title('Distribution of Pregnancies');
xlabel('Normalized Pregnancies');
ylabel('Frequency');
%histogram of glucose column with title and labelled axis
histogram(diabetesdataset4.Glucose)
title('Distribution of Glucose');
xlabel('Normalized Glucose');
ylabel('Frequency');
%histogram of blood pressure column with title and labelled axis
histogram(diabetesdataset4.BloodPressure)
title('Distribution of Blood Pressure');
xlabel('Normalized Blood Pressure');
ylabel('Frequency');
%histogram of skin thickness column with title and labelled axis
histogram(diabetesdataset4.SkinThickness)
title('Distribution of Skin Thickness');
xlabel('Normalized Skin Thickness');
ylabel('Frequency');

%histogram of insulin column with title and labelled axis
histogram(diabetesdataset4.Insulin)
title('Distribution of Insulin');
xlabel('Normalized Insulin');
```

```
ylabel('Frequency');

%histogram of BMI column with title and labelled axes
histogram(diabetesdataset4.BMI)
title('Distribution of BMI');
xlabel('BMI Normalized')
ylabel('Frequency')

%histogram of diabetes pedigree function column with title and labelled
%axes
histogram(diabetesdataset4.DiabetesPedigreeFunction)
title('Distribution of Diabetes Pedigree Function');
xlabel('Normalized Diabetes Pedigree Function');
ylabel('Frequency');

%histogram of Age column with title and labelled axes
histogram(diabetesdataset4.Age)
title('Distribution of Age');
xlabel('Normalized Age');
ylabel('Frequency');

%histogram of Outcome column with title and labelled axes
histogram(diabetesdataset4.Outcome)
title('Distribution of diabetes');
xlabel('Outcome');
ylabel('Frequency');

%%finally lets plot a correlation matrix and heatmap of our dataset%%
%reference:https://uk.mathworks.com/help/stats/corr.html
normalizedcorrelation=corr(diabetesdataset4{:,:}, 'Rows', 'complete');
%in the above code line we used the corr function to calculate the
%correlation between the columns of our normalized dataset. We have used
%the {:,:} symbols to convert the dataset into a numeric matrix. The curly
%brackets {} gives us the array of the cell and the symbol ;,; means that
%we we choose all the rows and all the columns of the dataset. We have used
%'Rows' to specify the way we want to deal with NaN values. We used the
%'complete' function so that we only work with rows that dont have any
%missing values. Finally we stored the correlation matrix into a new
%variable called normalizedcorrelation

%%lets plot our heatmap%%
%reference:https://uk.mathworks.com/help/matlab/ref/heatmap.html
%lets add the variables for our x axis
xaxisvariables={'Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BM
I','DiabetesPedigreeFunction','Age','Outcome'};
%lets add the variables for our y axis
yaxisvariables={'Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BM
I','DiabetesPedigreeFunction','Age','Outcome'};
%lets use the reference to plot the finalized heatmap
normalizedheatmap=heatmap(xaxisvariables,yaxisvariables,normalizedcorrelation);
%lets add a title
title('Correlation Matrix For Normalized Diabetes Dataset')
%changing the colour of the heatmap
normalizedheatmap.Colormap=hot;
```

```
%%the above code concludes the first step of my project which is pre
%%processing and visualizing the data%%
%%now we have to start impamenting our two machine learning algoriths. We
%%will start with the Naive Bayes algorithm%%
%the first thing we have to do is assign a control random number generator
%for reproducibility and consistency
%reference:https://uk.mathworks.com/help/matlab/ref/rng.html
rng(1)

%the next thing we have to do is to partition the dataset
%reference:https://uk.mathworks.com/help/stats/cvpartition.html
partitiondata=cvpartition(size(diabetesdataset4,1),'HoldOut',0.2);
% in the above code we used the cv partition function to partition our
% data where 80% of the data is for training and 20% of the data is for
% testing. We use size to calculate the size of our dataset and 1 to
% use the number of rows of the dataset. We used the holdout cross
%validation  which is a popular partition to do. Alternatively we could
%have used the kfold validation partition. The p value represents the 20%
%of the dataset being hold out for testing which is a very common
%percentage to use

%%the next thing we have to do is split our dataset into a training set and
%%a testing set%%
%reference:https://uk.mathworks.com/matlabcentral/answers/1745585-how-to-split-test-
and-training-set
trainingdataset=diabetesdataset4(partitiondata.training,:);
%we first got the training data using the partition that we already did in
%the above code. We include the : in our code so that we select all
%columns
testingdataset=diabetesdataset4(partitiondata.test,:);
%we then got the test data using the partition that we already did in the
%above code. Again we have to include the : in order to select all of the
%columns

%%the next thing we have to do is to separate the feature variables and the
%%target variable%%
%reference:https://uk.mathworks.com/matlabcentral/answers/1745585-how-to-split-test-
and-training-set
xtrainingdata=trainingdataset(:, 1:end-1);
% again we use the : in our code to select all of the columns but we are
% also using 1:end-1 to exclude the last column from the trainingdataset
% since its the target variable outcome
ytrainingdata=trainingdataset.Outcome;
% here we extract the target variable outcome from the trainingdataset
xtestingdata=testingdataset(:, 1:end-1);
% here again we use the : in our code to select all of the columns but we
% are also using 1:end-1 to exclude the last column from the
% trainingdataset since its the target variable outcome
ytestingdata=testingdataset.Outcome;
%here again we exctract the target variable column outcome from the
%trainingdataset

%lets exctract the test data for naive bayes model as a csv file%
%reference:https://uk.mathworks.com/help/matlab/ref/writetable.html
writetable(xtestingdata, 'test set naive bayes.csv');
```

```
%now lets also save it as a matlab file%
%%reference:https://uk.mathworks.com/help/matlab/ref/save.html
save('test set naive bayes.mat',"xtestingdata")
%lets load the saved test set for the naive bayes model
load('test set naive bayes.mat')

%lets exctract the target data required for the confusion matrix of the
%naive bayes model as a csv file%
%reference:https://uk.mathworks.com/help/matlab/ref/writematrix.html
writematrix(ytestingdata,'target variable naive bayes.csv');

%now lets also save it as a matlab file%
%%reference:https://uk.mathworks.com/help/matlab/ref/save.html
save('target variable naive bayes.mat',"ytestingdata")
%lets load the saved test set for the naive bayes model
load('target variable naive bayes.mat')

%%the next thing we have to do is to train our naive bayes model%%
%reference:https://uk.mathworks.com/help/stats/classificationnaivebayes.html
naivebayesmodel=fitcnb(xtrainingdata,ytrainingdata);
%in the above code we use the fitcnb function which is the function to
%train the naive bayes algorithm

%%lets now make the prediction based on our trained model%%
%reference:https://uk.mathworks.com/help/stats/compactclassificationnaivebayes.predic
t.html
naivebayesprediction=predict(naivebayesmodel,xtestingdata);
%we used the predict function to make a prediction based on the naive bayes
%model that we have trained

%%now we can evaluate our model using different methods%%
%lets start with the confusion matrix which suits well classifications like
%naive bayes
%reference:https://uk.mathworks.com/help/stats/confusionmat.html#mw_cc607721-b874-
447d-96cd-b5b65e141c57
[confusionmatrixnaivebayes,order]=confusionmat(ytestingdata, naivebayesprediction);
%the above code gives us a 2x2 confusion matrix as we expected and we also
%included the order in our code so we dont get confused when we will
%comment on it

%the next thing we can do is calculate true positives true negatives false
%positives and false negatives off of our confusion matrix%%
truenegativesnaivebayes=confusionmatrixnaivebayes(1,1);
%we use top left entry (1,1) to get the true negatives since the order of
%our confusion matrix is 0 1
truepositivesnaivebayes=confusionmatrixnaivebayes(2,2);
% we use bottom right entry (2,2) to get the true positives since the order
% of our confusion matrix is 0 1
falsepositivesnaivebayes=confusionmatrixnaivebayes(1,2);
%we use the top right entry (1,2) to get the false positives since the
%order of our confusion matrix is 0 1
falsenegativesnaivebayes=confusionmatrixnaivebayes(2,1);
%we use the bottom left entry (2,1) to get the false negatives since the
%order of our confusion matrix is 0 1
```

```
%%the next thing we can do is calculate the classification accuracy which
%%is also good for classifications like naive bayes. We can find the
%%formula for it in our week 2 lecture notes page 20. Total number of
%correctly classified inputs/Total number of inputs means
%true positives + true negatives/ture positives + truenegatives+ false
%positives + false negatives
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
classificationaccuracynaivebayes=((truepositivesnaivebayes+truenegativesnaivebayes)/(
truepositivesnaivebayes+truenegativesnaivebayes+falsepositivesnaivebayes+falsenegativ
esnaivebayes));

%%now we can also calculate precision using the formula in our week 2
%lecture notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
precisionnaivebayes=truepositivesnaivebayes/(truepositivesnaivebayes+falsepositivesna
ivebayes);

%%now we can calculate recall using again the formula in our week 2 lecture
%notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
recallnaivebayes=truepositivesnaivebayes/(truepositivesnaivebayes+falsenegativesnaive
bayes);

%%we can also calculate the F1 score using again the formula in our week 2
%%lecture notes on page 21
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
f1naivebayes=2*truepositivesnaivebayes/(2*truepositivesnaivebayes+falsepositivesnaive
bayes+falsenegativesnaivebayes);

%%now we can finally calculate the ROC curve which is also a good
%%classification metric like naive bayes%%
%the first thing we need is to calculate the posterior probabilities using
%the predict function
%reference:https://uk.mathworks.com/help/stats/classificationgam.predict.html#mw_e1a8
8a8e-6b89-438c-8e32-b12bf99d413b
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[~,posteriorprobabilitiesnaivebayes]=predict(naivebayesmodel,xtestingdata);
%in the above code we use the ~ to ignore the predicted class labels and
%focus on the predicted probabilities instead.
%the next thing we have to do is to get the positive probabilities from the
%positive class
positiveposteriorprobabilitiesnaivebayes=posteriorprobabilitiesnaivebayes(:,2);
% in the above code we used (:,2) since the prediction outputs are labelled
%as 0 and 1 so 1 is the positive probabilities
%now we have to calculate the ROC curve
%reference:https://uk.mathworks.com/help/stats/perfcurve.html#d126e817516[falseposition
vesnaivebayes,truepositivesnaivebayes]=perfcurve(ytestingdata,positiveposteriorprobab
ilitiesnaivebayes,1);
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[falsepositivesnaivebayes,truepositivesnaivebayes,~,aucvaluenaivebayes]=perfcurve(yte
stingdata,positiveposteriorprobabilitiesnaivebayes,1);
%in the above code we calculate the ROC graph and AUC using the reference
%and the perf function. We include the false positive rate and the true
% positive rate as our X and Y.The labels are the true labels which is the
```

```
%ytesting data. The scores are the positive posterior predictions and the oom
%possclass is the positive class so we can replace it with 1. We are using
%the ~ to ignore the threshold values
%now we can plot the x and y axis
%reference:https://uk.mathworks.com/help/matlab/ref/plot.html
plot(falsepositivesnaivebayes, truepositivesnaivebayes);
%labeling the x axis
xlabel('False Positive Rate')
%labeling the y axis
ylabel('True Positive Rate')
%adding a title
title('ROC Curve For Naive Bayes Using Holdout Method')

%%finally lets see what the AUC value is%%
%reference:https://uk.mathworks.com/help/matlab/ref/disp.html
disp(aucvaluenaivebayes)
%in the above code we used the disp function to show the AUC value

%%lets move on to decision tree model using holdout method%%

%%the first thing we have to do is remove the outliers from our dataset
%%once again and save it as a new dataset%%
%reference:https://uk.mathworks.com/help/matlab/ref/rmoutliers.html
diabetesdataset5=rmoutliers(diabetesdataset2,'mean');
%%in the above code we used the rmoutliers function and specified the method
%to be "mean" so that an outlier is detected only when it is 3 standard
%deviations away from the mean

%%now we will standardize rather than normalize our variables in order to
%%have a different approach to that of the naive bayes model%%.We wont
%%standardize the pregnancies and outcome column because it does not make
%%sense since. Pregnancies is a count variable and outcome is binary%%The
%%formula can be found in week 2 lecture notes on page 8%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
%as we can see we need to calculate the mean and standard deviation for
%each column
%reference:https://uk.mathworks.com/help/matlab/ref/mean.html
%reference:https://uk.mathworks.com/help/matlab/ref/std.html
diabetesdataset5.Glucose=(diabetesdataset5.Glucose-
mean(diabetesdataset5.Glucose))/std(diabetesdataset5.Glucose);
%the above code standardizes the glucose column using the formula from the
%notes by first calculating the mean and standard deviation
diabetesdataset5.Glucose
%the above code shows us the standardized glucose column
diabetesdataset5.BloodPressure=(diabetesdataset5.BloodPressure-
mean(diabetesdataset5.BloodPressure))/std(diabetesdataset5.BloodPressure);
%the above code standardizes the blood pressure column using the formula
%from the notes by first calculating the mean and standard deviation
diabetesdataset5.BloodPressure
%the above code shows us the standardized blood pressure column
diabetesdataset5.SkinThickness=(diabetesdataset5.SkinThickness-
mean(diabetesdataset5.SkinThickness))/std(diabetesdataset5.SkinThickness);
%the above code standardizes the skin thickness column using the formula
%from notes by first calculating the mean and standard deviation
diabetesdataset5.SkinThickness
```

```
%the above code shows the standardized skin thickness column
diabetesdataset5.Insulin=(diabetesdataset5.Insulin-
mean(diabetesdataset5.Insulin))/std(diabetesdataset5.Insulin);
%the above code standardizes the insulin column using the formula from the
%notes by first calculating the mean and standard deviation
diabetesdataset5.Insulin
%the above code shows the standardized insulin column
diabetesdataset5.BMI=(diabetesdataset5.BMI-
mean(diabetesdataset5.BMI))/std(diabetesdataset5.BMI);
%the above code standardizes the BMI column using the formula from the
%notes by first calcul.ating the mean and standard deviation
diabetesdataset5.BMI
%the above code shows the standardized BMI column
diabetesdataset5.DiabetesPedigreeFunction=(diabetesdataset5.DiabetesPedigreeFunction-
mean(diabetesdataset5.DiabetesPedigreeFunction))/std(diabetesdataset5.DiabetesPedigre
eFunction);
%the above code standardizes the diabetespedigreefunction column using the
%formula from the notes by first calculating the mean and standard
%deviation
diabetesdataset5.DiabetesPedigreeFunction
%the above code shows the standardized diabetes pedigree function
diabetesdataset5.Age=(diabetesdataset5.Age-
mean(diabetesdataset5.Age))/std(diabetesdataset5.Age);
%the above code standardizes the age column using the formula from the
%notesby first calculating the mean and standard deviation
diabetesdataset5.Age
%the above code shows the standardized age column

%%now lets save the normalized column in a new dataset%%
diabetesdataset6=diabetesdataset5;

%%lets check the summary statistics for our standardized dataset%%
%reference:https://uk.mathworks.com/help/matlab/ref/table.summary.html
summary(diabetesdataset6)
%the summary function shows us the minimum, maximum and median values of
%each column. We can see that all the values for each column that we
%normalized has been normalized as expected

%lets calculate the mean of our dataset
%reference:https://uk.mathworks.com/help/matlab/ref/mean.html
meanstandardized=mean(diabetesdataset6);

%lets calculate the median of our dataset
%reference:https://uk.mathworks.com/help/matlab/ref/median.html
medianstandardized=median(diabetesdataset6);

%lets calculate the standard deviation of our dataset
%reference: https://uk.mathworks.com/help/matlab/ref/std.html
standarddeviationstandardized=std(diabetesdataset6);

%%lets visualize each of our variables using a histogram with a title and
%labelled axis%%
%reference:
https://uk.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.ht
ml
```

```
%reference:https://uk.mathworks.com/help/matlab/titles-and-labels.html
%histogram of pregnancies column with title and labelled axis
histogram(diabetesdataset6.Pregnancies)
title('Distribution of Pregnancies');
xlabel('Standardized Pregnancies');
ylabel('Frequency');

%histogram of glucose column with title and labelled axis
histogram(diabetesdataset6.Glucose)
title('Distribution of Glucose');
xlabel('Standardized Glucose');
ylabel('Frequency');

%histogram of blood pressure column with title and labelled axis
histogram(diabetesdataset6.BloodPressure)
title('Distribution of Blood Pressure');
xlabel('Standardized Blood Pressure');
ylabel('Frequency');
%histogram of skin thickness column with title and labelled axis
histogram(diabetesdataset6.SkinThickness)
title('Distribution of Skin Thickness');
xlabel('Standardized Skin Thickness');
ylabel('Frequency');

%histogram of insulin column with title and labelled axis
histogram(diabetesdataset6.Insulin)
title('Distribution of Insulin');
xlabel('Standardized Insulin');
ylabel('Frequency');

%histogram of BMI column with title and labelled axis
histogram(diabetesdataset6.BMI)
title('Distribution of BMI');
xlabel('BMI Standardized');
ylabel('Frequency');

%histogram of Diabetes Pedigree Function column with title and labelled
%axis
histogram(diabetesdataset6.DiabetesPedigreeFunction)
title('Distribution of Diabetes Pedigree Function');
xlabel('Standardized Diabetes Pedigree Function');
ylabel('Frequency');

%histogram of Age column with title and labelled axis
histogram(diabetesdataset6.Age)
title('Distribution of Age');
xlabel('Standardized Age');
ylabel('Frequency');

%%finally lets plot a correlation matrix and heatmap of our dataset%%
%reference:https://uk.mathworks.com/help/stats/corr.html
normalizedcorrelation2=corr(diabetesdataset6{:,:},'Rows','complete');
%in the above code line we used the corr function to calculate the
%correlation between the columns of our standardized dataset. We have used
%the {:,:} symbols to convert the dataset into a numeric matrix. The curly
```

```
%brackets {} gives us the array of the cell and the symbol ;,; means that
%we we choose all the rows and all the columns of the dataset. We have used
%'Rows' to specify the way we want to deal with NaN values. We used the
%'complete' function so that we only work with rows that dont have any
%missing values. Finally we stored the correlation matrix into a new
%variable called normalizedcorrelation2

%%lets plot our heatmap%%
%reference:https://uk.mathworks.com/help/matlab/ref/heatmap.html
%lets add the variables for our x axis
xaxisvariables2={'Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','B
MI','DiabetesPedigreeFunction','Age','Outcome'};
%lets add the variables for our y axis
yaxisvariables2={'Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','B
MI','DiabetesPedigreeFunction','Age','Outcome'};
%lets use the reference to plot the finalized heatmap
standardizedheatmap=heatmap(xaxisvariables2,yaxisvariables2,normalizedcorrelation2);
%lets add a title
title('Correlation Matrix For Standardized Diabetes Dataset')
%lets add colour to the heatmap
standardizedheatmap.Colormap=jet;

%%Now we will have to implament the decision tree model%%
%we have to again assign a control random number generator
%for reproducibility and consistency
%reference:https://uk.mathworks.com/help/matlab/ref/rng.html
rng(1)

%the next thing we have to do is to partition the dataset%%
%reference:https://uk.mathworks.com/help/stats/cvpartition.html
partitiondata2=cvpartition(size(diabetesdataset6,1),'HoldOut',0.2);
% in the above code we used the cv partition function to partition our
% data where 80% of the data is for training and 20% of the data is for
% testing. We use size to calculate the size of our dataset and 1 to
% use the number of rows of the dataset. We used the holdout cross
%validation  which is a popular partition to do. Alternatively we could
%have used the kfold validation partition. The p value represents the 20%
%of the dataset being hold out for testing which is a very common
%percentage to use

%%the next thing we have to do is split our dataset into a training set and
%%a testing set%%
%reference:https://uk.mathworks.com/matlabcentral/answers/1745585-how-to-split-test-
and-training-set
trainingdataset2=diabetesdataset6(partitiondata2.training,:);
%we first got the training data using the partition that we already did in
%the above code. We include the : in our code so that we select all
%columns
testingdataset2=diabetesdataset6(partitiondata2.test,:);
%we then got the test data using the partition we already did in the above
%code. Again we have to include the : in order to select all of the columns

%%the next thing we have to do is to separate the feature variables and the
%%target variable%%
```

```
%reference:https://uk.mathworks.com/matlabcentral/answers/1745585-how-to-split-test-
and-training-set
xtrainingdata2=trainingdataset2(:, 1:end-1);
% again we use the : in our code to select all of the columns but we are
% also using 1:end-1 to exclude the last column from the trainingdataset2
% since its the target variable outcome
ytrainingdata2=trainingdataset2.Outcome;
% here we extract the target variable outcome from the trainingdataset2
xtestingdata2=testingdataset2(:, 1:end-1);
% here again we use the : in our code to select all of the columns but we
% are also using 1:end-1 to exclude the last column from the
% trainingdataset2 since its the target variable outcome
ytestingdata2=testingdataset2.Outcome;
%here again we exctract the target variable column outcome from the
%trainingdataset2

%lets exctract the test data for decision tree model as a csv file%
%reference:https://uk.mathworks.com/help/matlab/ref/writetable.html
writetable(xtestingdata2, 'test set decision tree.csv');

%now lets also save it as a matlab file%
%%reference:https://uk.mathworks.com/help/matlab/ref/save.html
save('test set decision tree.mat',"xtestingdata2")
%lets load the saved test data for the decision tree model
load('test set decision tree.mat')

%lets exctract the target data required for the confusion matrix of the
%decision tree model as a csv file%
%reference:https://uk.mathworks.com/help/matlab/ref/writematrix.html
writematrix(ytestingdata2, 'target variable decision tree.csv');

%now lets also save it as a matlab file%
%%reference:https://uk.mathworks.com/help/matlab/ref/save.html
save('target variable decision tree.mat',"ytestingdata2")
%lets load the saved test set for the naive bayes model
load('target variable decision tree.mat')

%%now we have to train our decision tree model%%
%reference:https://uk.mathworks.com/help/stats/decision-trees.html
decisiontreemodel=fitctree(xtrainingdata2,ytrainingdata2);
%in the above code we used the fitctree function in order to train the
%decision tree algorithm

%%lets now make the prediction based on our trained model%%
%reference:https://uk.mathworks.com/help/stats/compactclassificationnaivebayes.predic
t.html
[decisiontreeprediction]=predict(decisiontreemodel,xtestingdata2);
%in the above code we used the predict function in order to make a
%prediction and get the scores using the decision tree model we have
%trained

%%lets evaluate our decision tree model starting with the confusion
%%matrix%%
%reference:https://uk.mathworks.com/help/stats/confusionmat.html#mw_cc607721-b874-
447d-96cd-b5b65e141c57
```

```
[confusionmatrixdecisiontree,order]=confusionmat(ytestingdata2,decisiontreeprediction
);
%the above code gives us a 2x2 confusion matrix as we expected and we also
%included the order in our code so we dont get confused when we will
%comment on it

%the next thing we can do is calculate true positives true negatives false
%positives and false negatives off of our confusion matrix%%
truenegativesdecisiontree=confusionmatrixdecisiontree(1,1);
%we use top left entry (1,1) to get the true negatives since the order of
%our confusion matrix is 0 1
truepositivesdecisiontree=confusionmatrixdecisiontree(2,2);
% we use bottom right entry (2,2) to get the true positives since the order
% of our confusion matrix is 0 1
falsepositivesdecisiontree=confusionmatrixdecisiontree(1,2);
%we use the top right entry (1,2) to get the false positives since the
%order of our confusion matrix is 0 1
falsenegativesdecisiontree=confusionmatrixdecisiontree(2,1);
%we use the bottom left entry (2,1) to get the false negatives since the
%order of our confusion matrix is 0 1

%%the next thing we can do is calculate the classification accuracy which
%%is also good for classifications like decision trees. We can find the
%%formula for it in our week 2 lecture notes page 20. Total number of
%correctly classified inputs/Total number of inputs means
%true positives + true negatives/ture positives + truenegatives+ false
%positives + false negatives
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
classificationaccuracydecisiontree=((truepositivesdecisiontree+truenegativesdecisiont
ree)/(truepositivesdecisiontree+truenegativesdecisiontree+falsepositivesdecisiontree+
falsenegativesdecisiontree));

%%now we can firstly calculate precision using the formula in our week 2
%lecture notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
precisiondecisiontree=truepositivesdecisiontree/(truepositivesdecisiontree+falseposit
ivesdecisiontree);

%%now we can calculate recall using again the formula in our week 2 lecture
%notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
recalldecisiontree=truepositivesdecisiontree/(truepositivesdecisiontree+falsenegative
sdecisiontree);

%%we can also calculate the F1 score using again the formula in our week 2
%%lecture notes on page 21
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
f1decisiontree=2*truepositivesdecisiontree/(2*truepositivesdecisiontree+falsepositive
sdecisiontree+falsenegativesdecisiontree);

%%now we can finally calculate the ROC curve which is also a good
%%classification metric like naive bayes%%
%the first thing we need is to calculate the posterior probabilities using
%the predict function
```

```
%reference:https://uk.mathworks.com/help/stats/classificationgam.predict.html#mw_e1a8
8a8e-6b89-438c-8e32-b12bf99d413b
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[~,posteriorprobabilitiesdecisiontree]=predict(decisiontreemodel,xtestingdata2);
%in the above code we use the ~ to ignore the predicted class labels and
%focus on the predicted probabilities instead.

%the next thing we have to do is to get the positive probabilities from the
%positive class
positiveposteriorprobabilitiesdecisiontree=posteriorprobabilitiesdecisiontree(:,2);
% in the above code we used (:,2) since the prediction outputs are labelled
%as 0 and 1 so 1 is the positive probabilities
%now we have to calculate the ROC curve
%reference:https://uk.mathworks.com/help/stats/perfcurve.html#d126e817516[falseposititi
vesnaivebayes,truepositivesnaivebayes]=perfcurve(ytestingdata,positiveposteriorprobab
ilitiesnaivebayes,1);
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[falsepositivesdecisiontree,truepositivesdecisiontree,~,aucvaluedecisiontree]=perfcur
ve(ytestingdata2,positiveposteriorprobabilitiesdecisiontree,1);
%in the above code we calculate the ROC graph and AUC using the reference
%and the perf function. We include the false positive rate and the true
% positive rate as our X and Y.The labels are the true labels which is the
%ytesting data2. The scores are the positive posterior predictions and the
%possclass is the positive class so we can replace it with 1. We are using
%the ~ to ignore the threshold values
%now we can plot the x and y axis
%reference:https://uk.mathworks.com/help/matlab/ref/plot.html
plot(falsepositivesdecisiontree,truepositivesdecisiontree);
%labeling the x axis
xlabel('False Positive Rate')
%labeling the y axis
ylabel('True Positive Rate')
%adding a title
title('ROC Curve For Decision Tree Using Holdout Method')

%%finally lets see what the AUC value is%%
%reference:https://uk.mathworks.com/help/matlab/ref/disp.html
disp(aucvaluedecisiontree)
%in the above code we used the disp function to show the AUC value


%%now lets perform some feature selection on our decision tree model we
%%will not do this for the naive bayes model because it treats each
%%variable independently given the class label%%

%lets calculate the important features of our decision tree model%
%it will be difficult to do this for the naive bayes model since it treats
%each feature independently
%reference:https://uk.mathworks.com/help/stats/compactclassificationtree.predictorimp
ortance.html
importantfeaturesdecisiontree=predictorImportance(decisiontreemodel);
%lets display the important features of our model
%reference:https://uk.mathworks.com/help/matlab/ref/disp.html
disp(importantfeaturesdecisiontree)
```

```
%finally lets plot a bar graph of our important features
%reference:https://uk.mathworks.com/help/matlab/ref/bar.html
%reference:https://uk.mathworks.com/help/matlab/ref/gca.html
%reference:https://uk.mathworks.com/help/matlab/ref/xticklabels.html
bar(importantfeaturesdecisiontree);
title('Feature Selection For Decision Tree Model');
xlabel('Variables');
ylabel('Importance');
set(gca,'XTickLabel',xaxisvariables2);
%in the above code we use set gca and XTickLabel in order to have our x
%axis as our xaxisvariables2 variables so that we dont have to specify the
%variables %again

%%Finally we can perform hyperparameter optimization on our models%%

%lets start with the naive bayes model using the holdout dataset
%%the first thing we have to do is assign a control random number generator
%for reproducibility and consistency
%reference:https://uk.mathworks.com/help/matlab/ref/rng.html
rng(1)

%lets perform hyperparameter optimization to get the best naive bayes
%model%
%reference:https://uk.mathworks.com/help/stats/hyperparameter-optimization-in-
classification-learner-app.html#mw_5d2b39ea-dff8-4926-aca0-96d972deb768
%reference:https://uk.mathworks.com/help/stats/fitcnb.html
bestmodelnaivebayes=fitcnb(xtrainingdata,ytrainingdata,"OptimizeHyperparameters","aut
o");
%in the above code we make a hyperparameter optimization using our trained
%naive bayes model. We use the auto choice to let matlab choose how many
%hyperparameters to optimize in order to give us a fast and stable solution

%lets save the best naive bayes model as  a matlab file
%reference:https://uk.mathworks.com/help/matlab/ref/save.html
save('bestmodelnaivebayes.mat',"bestmodelnaivebayes")
%lets load the saved best naive bayes model
load('bestmodelnaivebayes.mat')

%%now we can make predictions using our hyperparameter naive bayes model%%
%reference:https://uk.mathworks.com/help/stats/compactclassificationnaivebayes.predic
t.html
bestmodelpredictionnaivebayes=predict(bestmodelnaivebayes,xtestingdata);
%we used the predict function to make a prediction based on the naive bayes
%model that we have trained with adjusted hyperparameters

%%now we can evaluate our model using different methods%%
%lets start with the confusion matrix
%reference:https://uk.mathworks.com/help/stats/confusionmat.html#mw_cc607721-b874-
447d-96cd-b5b65e141c57
[bestmodelconfusionmatrixnaivebayes,order]=confusionmat(ytestingdata,bestmodelpredict
ionnaivebayes);
%the above code gives us a 2x2 confusion matrix as we expected and we also
%included the order in our code so we dont get confused when we will
%comment on it
```

```
%the first thing we can do is calculate true positives true negatives false
%positives and false negatives off of our confusion matrix%%
truenegativesbestmodelnaivebayes=bestmodelconfusionmatrixnaivebayes(1,1);
%we use top left entry (1,1) to get the true negatives since the order of
%our confusion matrix is 0 1
truepositivesbestmodelnaivebayes=bestmodelconfusionmatrixnaivebayes(2,2);
% we use bottom right entry (2,2) to get the true positives since the order
% of our confusion matrix is 0 1
falsepositivesbestmodelnaivebayes=bestmodelconfusionmatrixnaivebayes(1,2);
%we use the top right entry (1,2) to get the false positives since the
%order of our confusion matrix is 0 1
falsenegativesbestmodelnaivebayes=bestmodelconfusionmatrixnaivebayes(2,1);
%we use the bottom left entry (2,1) to get the false negatives since the
%order of our confusion matrix is 0 1

%%the next thing we can do is calculate the classification accuracy which
%%We can find the formula for it in our week 2 lecture notes page 20. Total
%number of correctly classified inputs/Total number of inputs means true
%positives + true negatives/ture positives + true negatives+
%positives + false negatives
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
classificationaccuracybestmodelnaivebayes=((truepositivesbestmodelnaivebayes+truenega
tivesbestmodelnaivebayes)/(truepositivesbestmodelnaivebayes+truenegativesbestmodelnai
vebayes+falsepositivesbestmodelnaivebayes+falsenegativesbestmodelnaivebayes));

%%now we can firstly calculate precision using the formula in our week 2
%lecture notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
percisionbestmodelnaivebayes=truepositivesbestmodelnaivebayes/(truepositivesbestmodel
naivebayes+falsepositivesbestmodelnaivebayes);

%%now we can calculate recall using again the formula in our week 2 lecture
%notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
recallbestmodelnaivebayes=truepositivesbestmodelnaivebayes/(truepositivesbestmodelnai
vebayes+falsenegativesbestmodelnaivebayes);

%%we can also calculate the F1 score using again the formula in our week 2
%%lecture notes on page 21
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
f1bestmodelnaivebayes=2*truepositivesbestmodelnaivebayes/(2*truepositivesbestmodelnai
vebayes+falsepositivesbestmodelnaivebayes+falsenegativesbestmodelnaivebayes);

%we can also calculate the resubloss of our naive bayes model since its a
%good evaluation metric when we do hyperparameter optimization
%reference:https://uk.mathworks.com/help/stats/classificationnaivebayes.html
%reference:https://uk.mathworks.com/help/stats/classificationsvm.resubloss.html
resublossbestmodelnaivebayes=resubLoss(bestmodelnaivebayes);

%%now we can finally calculate the ROC curve which is also a good
%%classification metric like naive bayes%%
%the first thing we need is to calculate the posterior probabilities using
%the predict function
%reference:https://uk.mathworks.com/help/stats/classificationgam.predict.html#mw_e1a8
8a8e-6b89-438c-8e32-b12bf99d413b
```

```
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[~,posteriorbestmodelprobabilitiesnaivebayes]=predict(bestmodelnaivebayes,xtestingdat
a);
%in the above code we use the ~ to ignore the predicted class labels and
%focus on the predicted probabilities instead.
%the next thing we have to do is to get the positive probabilities from the
%positive class
positiveposteriorbestmodelprobabilitiesnaivebayes=posteriorbestmodelprobabilitiesnaiv
ebayes(:,2);
% in the above code we used (:,2) since the prediction outputs are labelled
%as 0 and 1 so 1 is the positive probabilities
%now we have to calculate the ROC curve
%reference:https://uk.mathworks.com/help/stats/perfcurve.html#d126e817516
[falsepositivesbestmodelnaivebayes,truepositivesbestmodelnaivebayes,~,aucvaluebestmod
elnaivebayes]=perfcurve(ytestingdata,positiveposteriorbestmodelprobabilitiesnaivebaye
s,1);
%in the above code we calculate the ROC graph and AUC using the reference
%and the perf function. We include the false positive rate and the true
% positive rate as our X and Y.The labels are the true labels which is the
%ytesting data. The scores are the positive posterior predictions and the
%possclass is the positive class so we can replace it with 1. We are using
%the ~ to ignore the threshold values
%now we can plot the x and y axis
%reference:https://uk.mathworks.com/help/matlab/ref/plot.html
plot(falsepositivesbestmodelnaivebayes,truepositivesbestmodelnaivebayes)
%labeling the x axis
xlabel('False Positive Rate')
%labeling the y axis
ylabel('True Positive Rate')
%adding a title
title('ROC Curve For Naive Bayes Best Model')
%%finally lets see what the AUC value is%%
%reference:https://uk.mathworks.com/help/matlab/ref/disp.html
disp(aucvaluebestmodelnaivebayes)
%in the above code we used the disp function to show the AUC value

%%now lets do hyperparameter optimization for the decision tree model using
%%the holdout dataset%%

%%the first thing we have to do is assign a control random number generator
%for reproducibility and consistency
%reference:https://uk.mathworks.com/help/matlab/ref/rng.html
rng(1)
%now lets perform hyperparameter optimization
%reference:https://uk.mathworks.com/help/stats/hyperparameter-optimization-in-
classification-learner-app.html#mw_5d2b39ea-dff8-4926-aca0-96d972deb768
%reference:https://uk.mathworks.com/help/stats/fitctree.html
bestmodeldecisiontree=fitctree(xtrainingdata2,ytrainingdata2,"OptimizeHyperparameters
","auto");
%in the above code we make a hyperparameter optimization using our trained
%decision tree model. We use the auto choice to let matlab choose how many
%hyperparameters to optimize in order to give us a fast and stable solution

%lets save the best decision tree model as  a matlab file
%reference:https://uk.mathworks.com/help/matlab/ref/save.html
```

```
save('bestmodeldecisiontree.mat',"bestmodeldecisiontree")
%lets load the saved best decision tree model
load('bestmodeldecisiontree.mat')

%%now we can make predictions using our hyperparameter decision tree
%%model%%
%reference:https://uk.mathworks.com/help/stats/compactclassificationnaivebayes.predic
t.html
bestmodelpredictiondecisiontree=predict(bestmodeldecisiontree,xtestingdata2);
%we used the predict function to make a prediction based on the naive bayes
%model that we have trained with adjusted hyperparameters

%%now we can evaluate our model using different methods%%
%lets start with the confusion matrix
%reference:https://uk.mathworks.com/help/stats/confusionmat.html#mw_cc607721-b874-
447d-96cd-b5b65e141c57
[bestmodelconfusionmatrixdecisiontree,order]=confusionmat(ytestingdata2,bestmodelpred
ictiondecisiontree);
%the above code gives us a 2x2 confusion matrix as we expected and we also
%included the order in our code so we dont get confused when we will
%comment on it

%the first thing we can do is calculate true positives true negatives false
%positives and false negatives off of our confusion matrix%%
truenegativesbestmodeldecisiontree=bestmodelconfusionmatrixdecisiontree(1,1);
%we use top left entry (1,1) to get the true negatives since the order of
%our confusion matrix is 0 1
truepositivesbestmodeldecisiontree=bestmodelconfusionmatrixdecisiontree(2,2);
% we use bottom right entry (2,2) to get the true positives since the order
% of our confusion matrix is 0 1
falsepositivesbestmodeldecisiontree=bestmodelconfusionmatrixdecisiontree(1,2);
%we use the top right entry (1,2) to get the false positives since the
%order of our confusion matrix is 0 1
falsenegativesbestmodeldecisiontree=bestmodelconfusionmatrixdecisiontree(2,1);
%we use the bottom left entry (2,1) to get the false negatives since the
%order of our confusion matrix is 0 1

%%the next thing we can do is calculate the classification accuracy which
%%We can find the formula for it in our week 2 lecture notes page 20. Total
%number of correctly classified inputs/Total number of inputs means true
%positives + true negatives/ture positives + true negatives+
%positives + false negatives
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
classificationaccuracybestmodeldecisiontree=((truepositivesbestmodeldecisiontree+true
negativesbestmodeldecisiontree)/(truepositivesbestmodeldecisiontree+truenegativesbest
modeldecisiontree+falsepositivesbestmodeldecisiontree+falsenegativesbestmodeldecision
tree));

%%now we can firstly calculate precision using the formula in our week 2
%lecture notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
percisionbestmodeldecisiontree=truepositivesbestmodeldecisiontree/(truepositivesbestm
odeldecisiontree+falsepositivesbestmodeldecisiontree);

%%now we can calculate recall using again the formula in our week 2 lecture
```

```
%notes on page 21%%
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
recallbestmodeldecisiontree=truepositivesbestmodeldecisiontree/(truepositivesbestmode
ldecisiontree+falsenegativesbestmodeldecisiontree);

%%we can also calculate the F1 score using again the formula in our week 2
%%lecture notes on page 21
%reference:https://moodle4.city.ac.uk/mod/folder/view.php?id=382059
f1bestmodeldecisiontree=2*truepositivesbestmodeldecisiontree/(2*truepositivesbestmode
ldecisiontree+falsepositivesbestmodeldecisiontree+falsenegativesbestmodeldecisiontree
);

%we can also calculate the resubloss of our decision tree model since its a
%good evaluation metric when we do hyperparameter optimization
%reference:https://uk.mathworks.com/help/stats/classificationnaivebayes.html
%reference:https://uk.mathworks.com/help/stats/classificationsvm.resubloss.html
resublossbestmodeldecisiontree=resubLoss(bestmodeldecisiontree);

%%now we can finally calculate the ROC curve which is also a good
%%classification metric like naive bayes%%
%the first thing we need is to calculate the posterior probabilities using
%the predict function
%reference:https://uk.mathworks.com/help/stats/classificationgam.predict.html#mw_e1a8
8a8e-6b89-438c-8e32-b12bf99d413b
%reference:https://uk.mathworks.com/help/matlab/ref/not.html
[~,posteriorbestmodelprobabilitiesdecisiontree]=predict(bestmodeldecisiontree,xtestin
gdata2);
%in the above code we use the ~ to ignore the predicted class labels and
%focus on the predicted probabilities instead.
%the next thing we have to do is to get the positive probabilities from the
%positive class
positiveposteriorbestmodelprobabilitiesdecisiontree=posteriorbestmodelprobabilitiesde
cisiontree(:,2);
% in the above code we used (:,2) since the prediction outputs are labelled
%as 0 and 1 so 1 is the positive probabilities
%now we have to calculate the ROC curve
%reference:https://uk.mathworks.com/help/stats/perfcurve.html#d126e817516
[falsepositivesbestmodeldecisiontree,truepositivesbestmodeldecisiontree,~,aucvaluebes
tmodeldecisiontree]=perfcurve(ytestingdata2,positiveposteriorbestmodelprobabilitiesde
cisiontree,1);
%in the above code we calculate the ROC graph and AUC using the reference
%and the perf function. We include the false positive rate and the true
% positive rate as our X and Y.The labels are the true labels which is the
%ytesting data. The scores are the positive posterior predictions and the
%possclass is the positive class so we can replace it with 1. We are using
%the ~ to ignore the threshold values
%now we can plot the x and y axis
%reference:https://uk.mathworks.com/help/matlab/ref/plot.html
plot(falsepositivesbestmodeldecisiontree,truepositivesbestmodeldecisiontree)
%labeling the x axis
xlabel('False Positive Rate')
%labeling the y axis
ylabel('True Positive Rate')
%adding a title
title('ROC Curve For Decision Tree Best Model')
```

```
%%finally lets see what the AUC value is%%
%reference:https://uk.mathworks.com/help/matlab/ref/disp.html
disp(aucvaluebestmodeldecisiontree)
%in the above code we used the disp function to show the AUC value
```