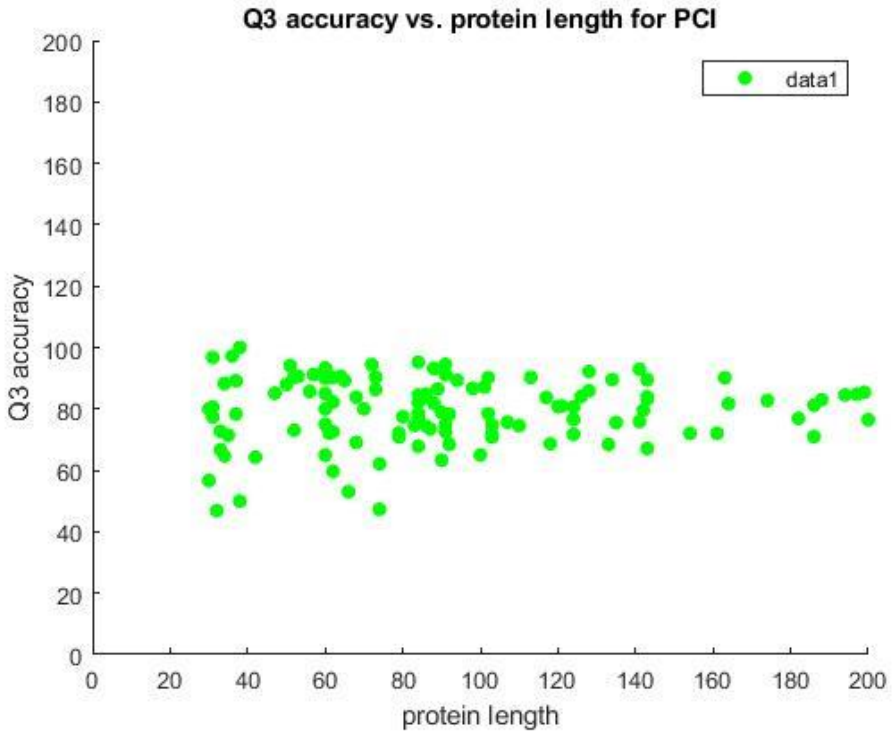
BIOM/SYSC5405 – PATTERN CLASSIFICATION AND EXPERIMENT DESIGN

Assignment 1

Solution 1: Classifier scores

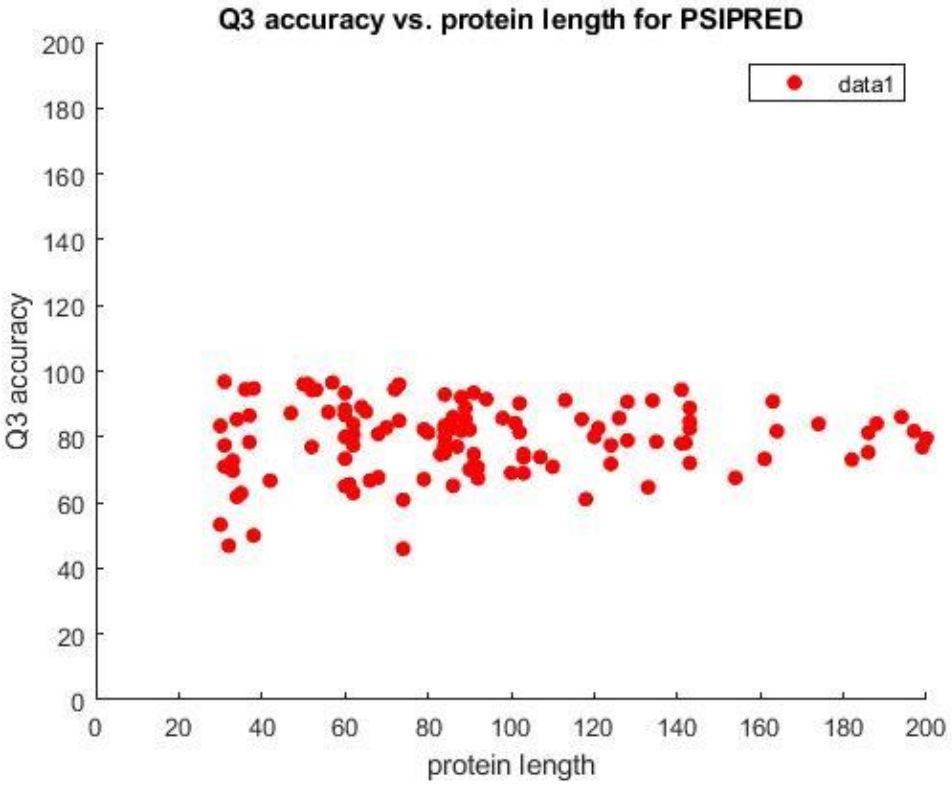
a)

```
%For PCI

%Read and define from dataset
PCI_Length = xlsread('assigData1.xls','PCI', 'B2:B126');
PCI_Acc = xlsread('assigData1.xls','PCI', 'D2:D126');

%Plot the Q3 accuracy vs. protein length
figure()
scatter(PCI_Length,PCI_Acc,'filled', 'green');
xlim([0 200]);
ylim([0 200]);
title('Q3 accuracy vs. protein length for PCI');
legend();
xlabel('protein length');
ylabel('Q3 accuracy');
```

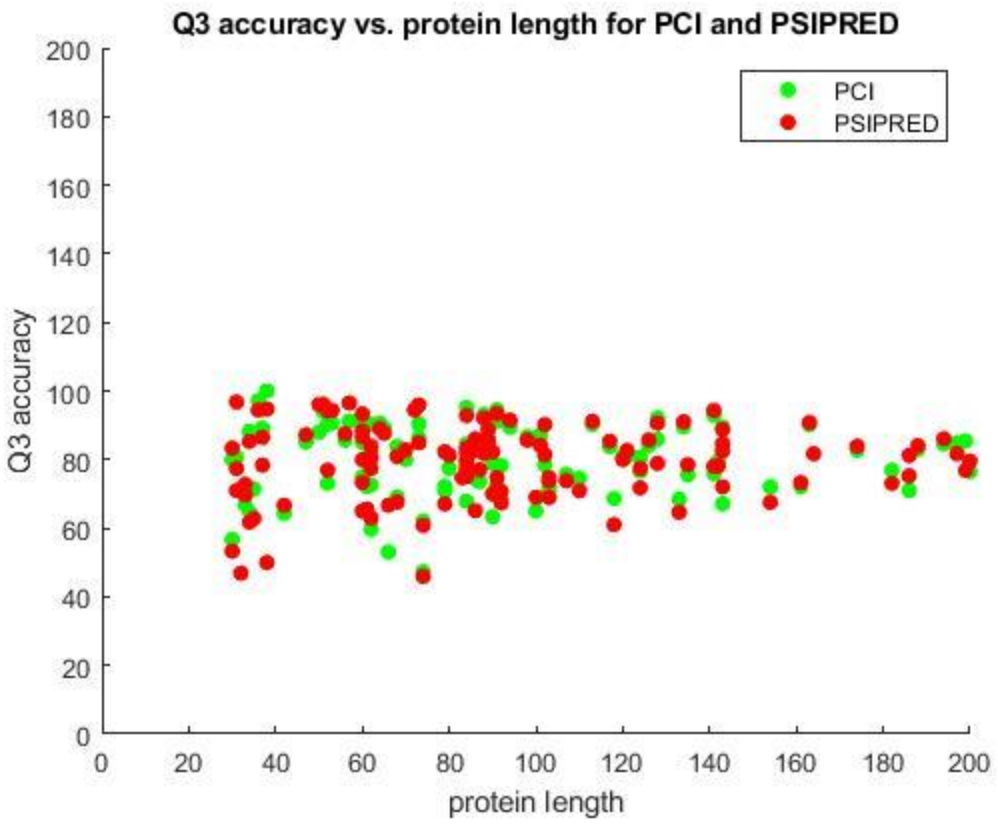



```
%For PSIPRED
```

```
%Read and define from dataset
PSIPRED_Length = xlsread('assigData1.xls','PSIPRED', 'B2:B126');
PSIPRED_Acc = xlsread('assigData1.xls','PSIPRED', 'D2:D126');

%Plot the Q3 accuracy vs. protein length
figure()
scatter(PSIPRED_Length,PSIPRED_Acc,'filled', 'red');
xlim([0 200]);
ylim([0 200]);
title('Q3 accuracy vs. protein length for PSIPRED');
legend();
xlabel('protein length');
ylabel('Q3 accuracy');
```




```
%For both PCI & PSIPRED
 figure()
 scatter(PCI_Length,PCI_Acc,'filled', 'green');
 xlim([0 200]);
 ylim([0 200]);
 hold on
 scatter(PSIPRED_Length,PSIPRED_Acc,'filled', 'red');
 title('Q3 accuracy vs. protein length for PCI and PSIPRED');
 xlabel('protein length');
 ylabel('Q3 accuracy');
 legend('PCI','PSIPRED');
 hold off
```



Based on the plot it appears that Q3 accuracy is not correlated with protein length for any of the method. I used scatterplot function to visualize the correlation of Q3 and length.

Scatterplot is used to present the relationship of two variables [1]. Two sets of data can be correlated if they have any relation between each other [1]. From the graphical representation for both classifiers, it is being observed that while the protein length is increasing, the accuracy Q3 remains almost same or close for each protein length datapoint. So, it means there is no linear relationship of the two quantitative datasets, and they are not correlated.

b)

To compute the correlation between Q3 accuracy and protein length for PCI and for PSIPRED, I applied Pearson correlation coefficient:

```
PCI_Coef = corrcoef(PCI_Length,PCI_Acc)
PSIPRED_Coef = corrcoef(PSIPRED_Length,PSIPRED_Acc)
```

It shows the results as-

```
PCI_Coef =
    1.0000    0.0281
    0.0281    1.0000
```

```
PSIPRED_Coef =
    1.0000    0.0108
    0.0108    1.0000
```

Pearson correlation coefficient measures the strength of the linear relationship between two quantitative variables. The coefficient is always a number between -1 to +1. A positive value indicates a positive association and negative value indicates negative association. If the value is zero that indicates that there is no correlation between two variables. There is one important fact about Pearson correlation coefficient is that we must assume that the relationship between two variable is linear [1].
From the computation of Pearson coefficient for both PCI and PSIPRED it is noticed that both values (PCI=0.0281, PSIPRED= 0.0108) are near to zero. So, there is no correlation between Q3 Accuracy and protein length for both classifiers.

c)

The mean, median, and standard deviation of the Matthews' correlation coefficient for PCI and PSIPRED is given below-

```
%Mean
PCI_Mean=mean(xlsread('assigData1.xls','PCI', 'C2:C126'))
PSIPRED_Mean=mean(xlsread('assigData1.xls','PSIPRED', 'C2:C126'))

PCI_Mean =
    0.6560

PSIPRED_Mean =
    0.6580

%Median
PCI_Median=median(xlsread('assigData1.xls','PCI', 'C2:C126'))
PSIPRED_Median=median(xlsread('assigData1.xls','PSIPRED', 'C2:C126'))

PCI_Median =
    0.6690

PSIPRED_Median =
    0.6730

%Standard Deviation
PCI_STD=std(xlsread('assigData1.xls','PCI', 'C2:C126'))
PSIPRED_STD=std(xlsread('assigData1.xls','PSIPRED', 'C2:C126'))

PCI_STD =
    0.1779
PSIPRED_STD =
    0.1757
```

Solution 2: Feature data

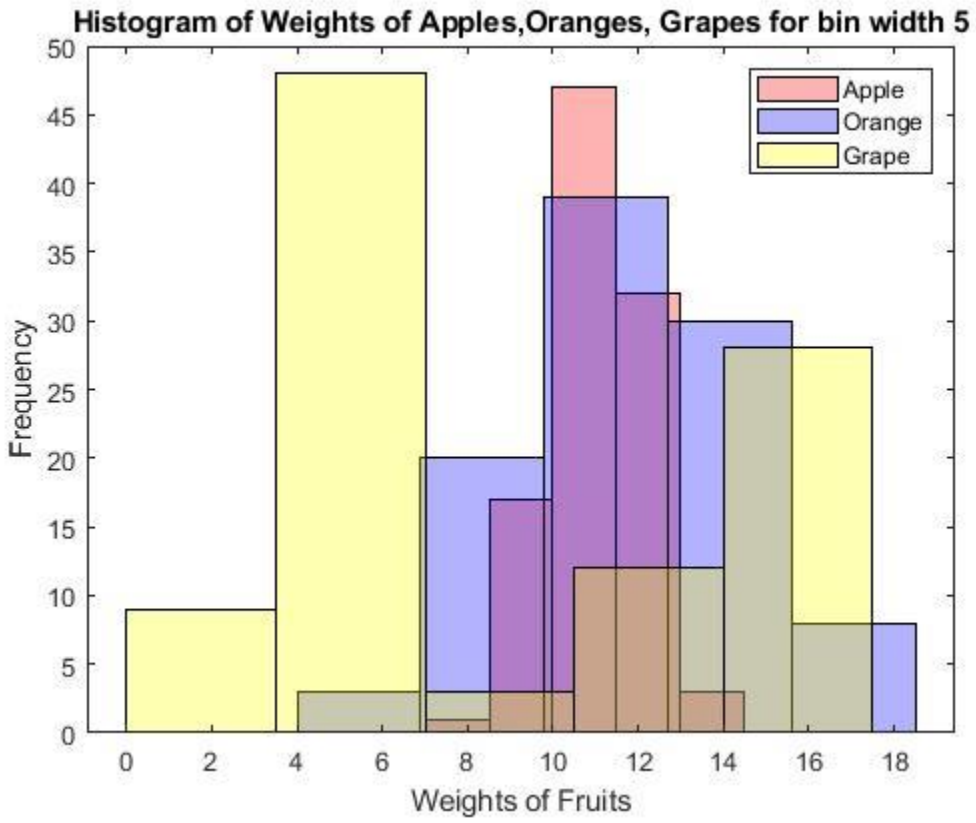
a)

```
clear
clear all
clc

fid = fopen('assigData2.tsv');
C = textscan(fid, '%f %f %f %f %f %f', 'HeaderLines', 1);
fclose(fid);

%6 estimates of mean
ma = mean(C{:,1}); % Mean of Weight of Apple
fprintf('Mean of apple weight: %0.5f\n', ma)
mb = mean(C{:,2}); % Mean of Weight of Orange
fprintf('Mean of orange weight: %0.5f\n', mb)
mc = mean(C{:,3}); % Mean of Weight of Grape
fprintf('Mean of grape weight: %0.5f\n', mc)
md = mean(C{:,4}); % Mean of Diameter of Apple
fprintf('Mean of apple diameter: %0.5f\n', md)
me = mean(C{:,5}); % Mean of Diameter of Orange
fprintf('Mean of orange diameter: %0.5f\n', me)
mf = mean(C{:,6}); % Mean of Diameter of Grape
fprintf('Mean of grape diameter: %0.5f\n\n', mf)

%6 estimates of variance
va = var(C{:,1}); % Variance of Weight of Apple
fprintf('Variance of apple weight: %0.5f\n', va)
vb = var(C{:,2}); % Mean of Weight of Orange
fprintf('Variance of orange weight: %0.5f\n', vb)
vc = var(C{:,3}); % Mean of Weight of Grape
fprintf('Variance of grape weight: %0.5f\n', vc)
vd = var(C{:,4}); % Mean of Diameter of Apple
fprintf('Variance of apple diameter: %0.5f\n', vd)
ve = var(C{:,5}); % Mean of Diameter of Orange
fprintf('Variance of orange diameter: %0.5f\n', ve)
vf = var(C{:,6}); % Mean of Diameter of Grape
fprintf('Variance of grape diameter: %0.5f\n\n', vf)

Mean of apple weight: 11.00308
Mean of orange weight: 11.94500
Mean of grape weight: 8.73336
Mean of apple diameter: 1006.70720
Mean of orange diameter: 1114.83385
Mean of grape diameter: 832.54623

Variance of apple weight: 1.40110
Variance of orange weight: 6.80662
Variance of grape weight: 24.78710
Variance of apple diameter: 1621.37446
Variance of orange diameter: 383.40582
Variance of grape diameter: 8356.38166
```

b)

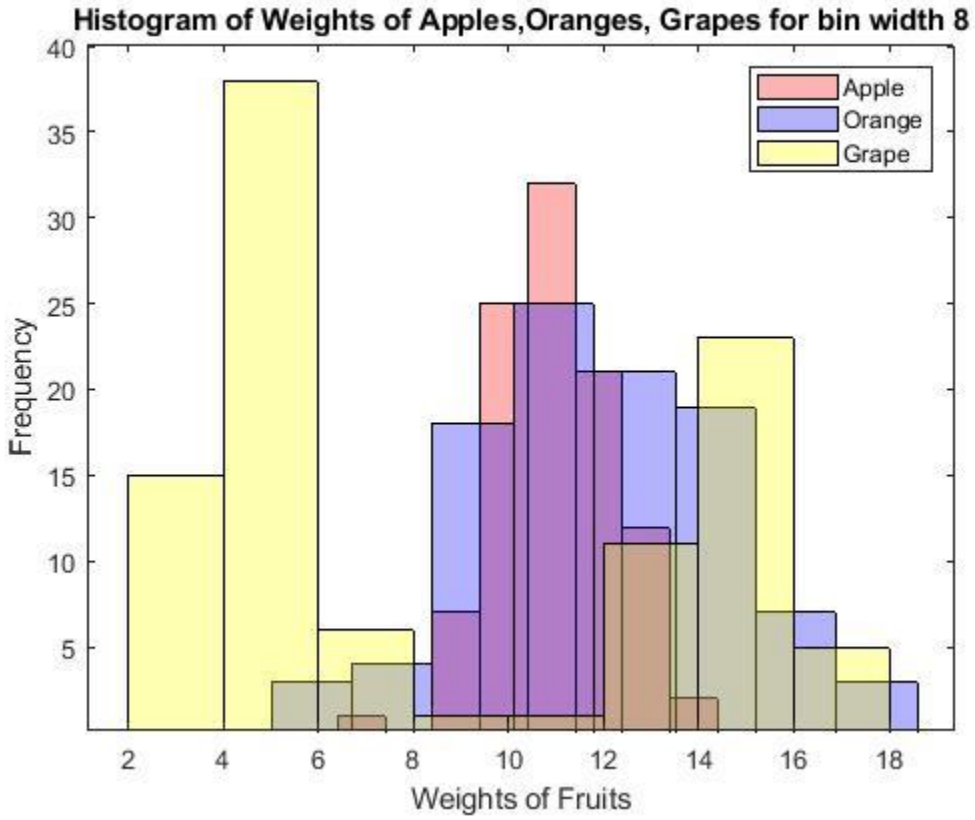
The histogram results are illustrated with three different bin width- 5, 8 and 11 for each feature.

```
%histogram for weight feature

% bin width 5
figure()
histogram(C{:,1},5, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,2},5, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,3},5, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Weights of Apples,Oranges, Grapes for bin width 5')
xlabel('Weights of Fruits')
ylabel('Frequency')
legend('Apple', 'Orange', 'Grape')
```

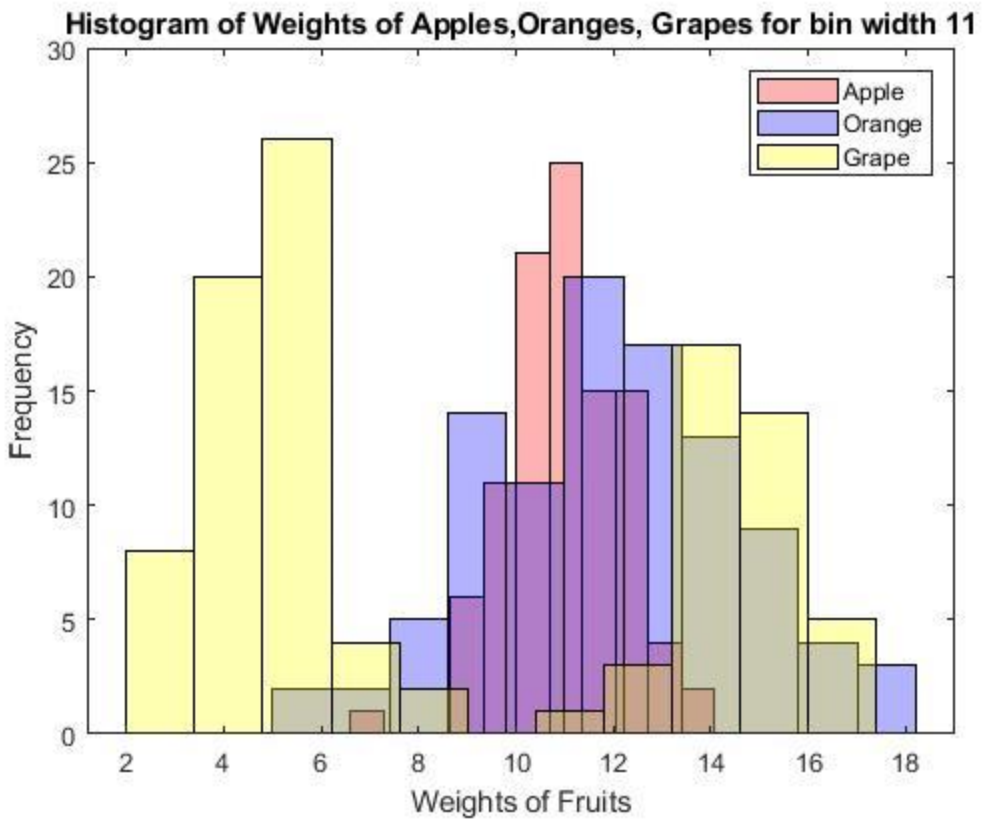



```
%bin width 8
figure()
histogram(C{:,1},8, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,2},8, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,3},8, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Weights of Apples,Oranges, Grapes for bin width 8')
xlabel('Weights of Fruits')
ylabel('Frequency')
```

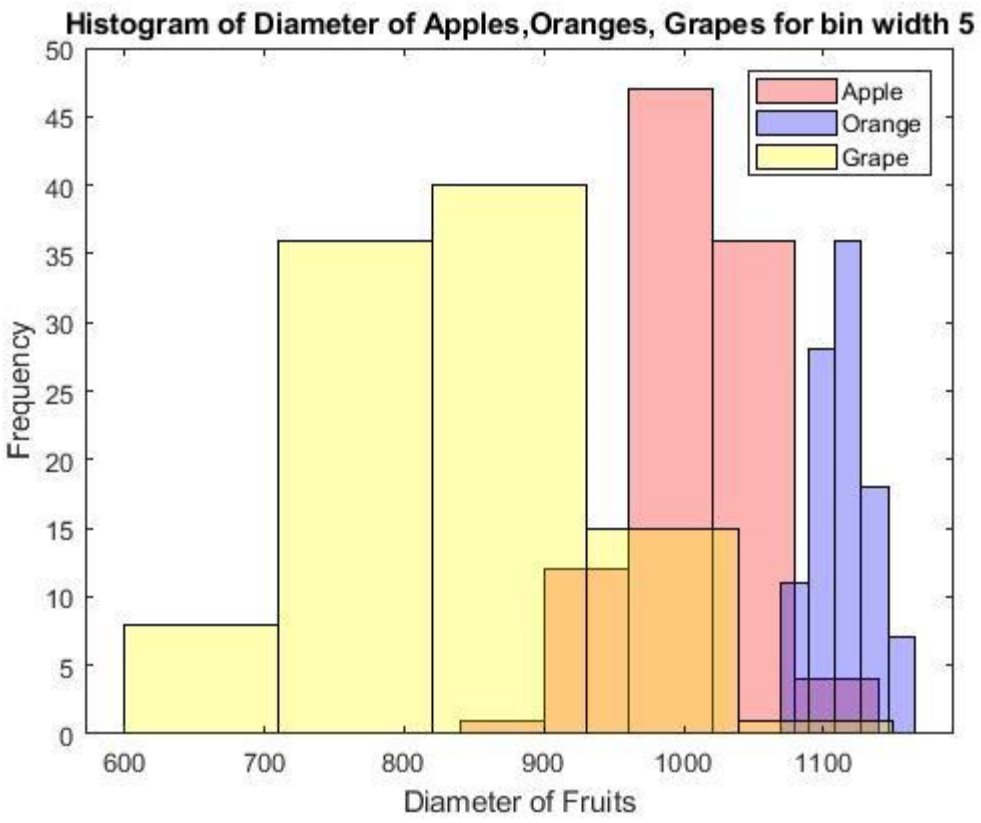
```
legend('Apple', 'Orange', 'Grape')
```



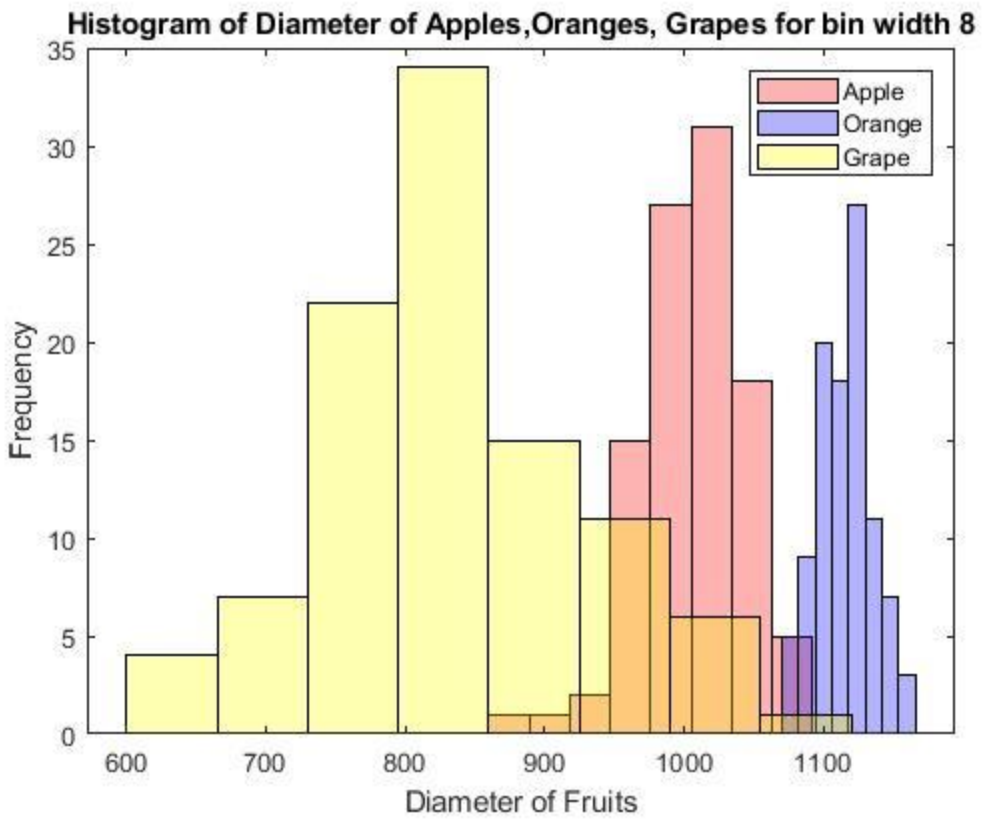

```
%bin width 11
figure()
histogram(C{:,1},11, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,2},11, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,3},11, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Weights of Apples,Oranges, Grapes for bin width 11')
xlabel('Weights of Fruits')
ylabel('Frequency')
legend('Apple', 'Orange', 'Grape')
```



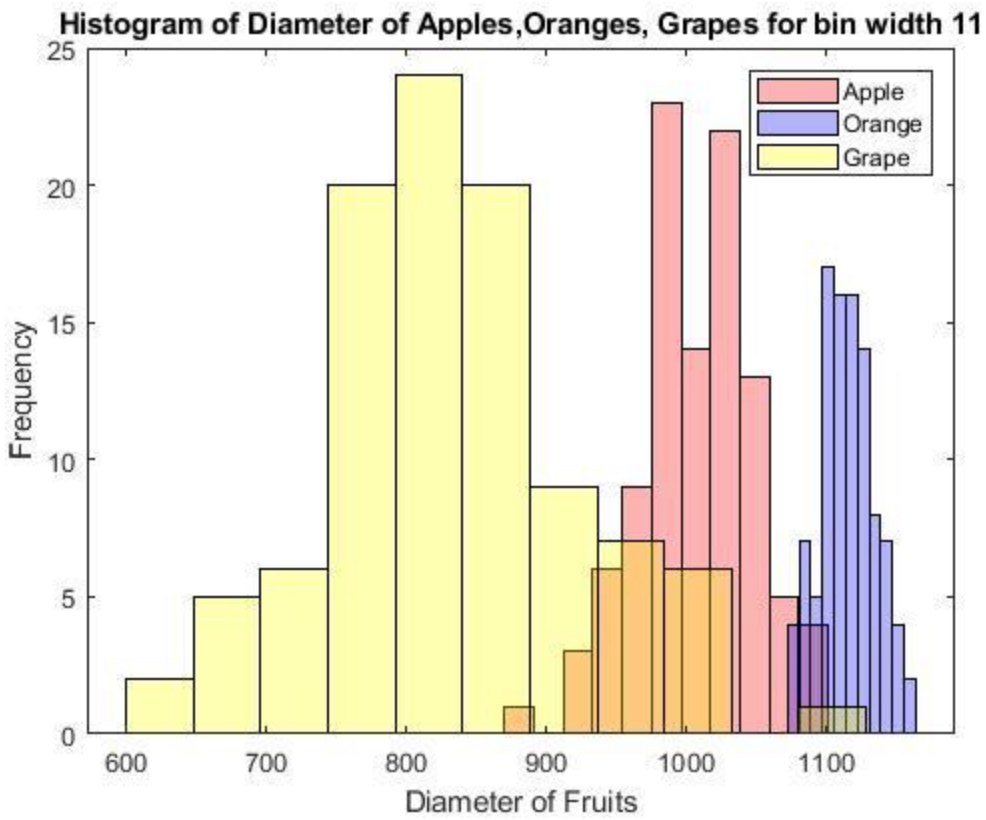

```
%histogram for diameter feature

%bin width 5

figure()
histogram(C{:,4},5, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,5},5, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,6},5, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Diameter of Apples,Oranges, Grapes for bin width 5')
xlabel('Diameter of Fruits')
ylabel('Frequency')
legend('Apple', 'Orange', 'Grape')
```




```
%bin width 8
figure()
histogram(C{:,4},8, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,5},8, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,6},8, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Diameter of Apples,Oranges, Grapes for bin width 8')
xlabel('Diameter of Fruits')
ylabel('Frequency')
legend('Apple', 'Orange', 'Grape')
```




```
%bin width 11
figure()
histogram(C{:,4},11, 'FaceColor','red', 'FaceAlpha',0.3)
hold on
histogram(C{:,5},11, 'FaceColor','blue', 'FaceAlpha',0.3)
hold on
histogram(C{:,6},11, 'FaceColor','yellow', 'FaceAlpha',0.3)
title('Histogram of Diameter of Apples,Oranges, Grapes for bin width 5')
xlabel('Diameter of Fruits')
ylabel('Frequency')
legend('Apple', 'Orange', 'Grape')
```



Preferred feature:

After analysing the histograms for weight and diameter feature for all classes it is cleared that diameter is the preferred feature to classify fruits (apple, orange, grape).

Histogram provides the scope to analyse the quantitative features analysis involving the range of values for each feature into a finite set of intervals and the generation of bins [2]. Every bin represents a range of values and the count of instances for which fall within that bin [2].

If we have a close look to the histograms of weight feature, we can see that the bars for apple and orange are overlapping highly where the bars for grapes are overlapping less with other two class. So, if we consider weight feature to classify the fruits, there will be always a high chance of not classifying apple and orange correctly as the data points are almost similar or close.

On the other hand, the histogram for diameter presents that the bars for each class are not overlapping highly like weight histograms. So, classifying fruits will be more effective by using diameter feature.

c)

```
%Normality test

Weights=[C{:,1:3}]
TestM = ismatrix(Weights)
W_Vector = reshape(Weights',[],1);
TestV = isvector(W_Vector)
Rounded=round(W_Vector)
```

```
[h,p] = lillietest(Rounded);

>> h
h =
     1
>> p
p =
   1.0000e-03
>>
```

Lilliefors test:
The Lilliefors test is an improvement on the Kolomogorov-Smirnov (K-S) test which is used for testing normality of data. It tests the null hypothesis that the data come from a normally distributed population. When the null hypothesis is not able to specify about exactly which normal distribution, it means that the test does not specify the expected mean and variance of the distribution [3].

"The Lilliefors test statistic is:

$$D^* = \max_x \downarrow \hat{F}(x) - G(x) \downarrow,$$

where $\hat{F}(x)$ is the empirical cdf of the sample data and $G(x)$ is the cdf of the hypothesized distribution with estimated parameters equal to the sample parameters.
lillietest can be used to test whether the data vector x has a lognormal or Weibull distribution by applying a transformation to the data vector and running the appropriate Lilliefors test:

- To test x for a lognormal distribution, test if log(x) has a normal distribution.
- To test x for a Weibull distribution, test if log(x) has an extreme value distribution.

The Lilliefors test cannot be used when the null hypothesis is not a location-scale family of distributions." [4]

From the code snippet we can see that the returned value of h = 1. It means that lillietest rejects the null hypothesis at the default 5% significance level.
So the distribution of vector W_Vector is not normally distributed.

Solution 3: Generating data & the normal distribution

a)

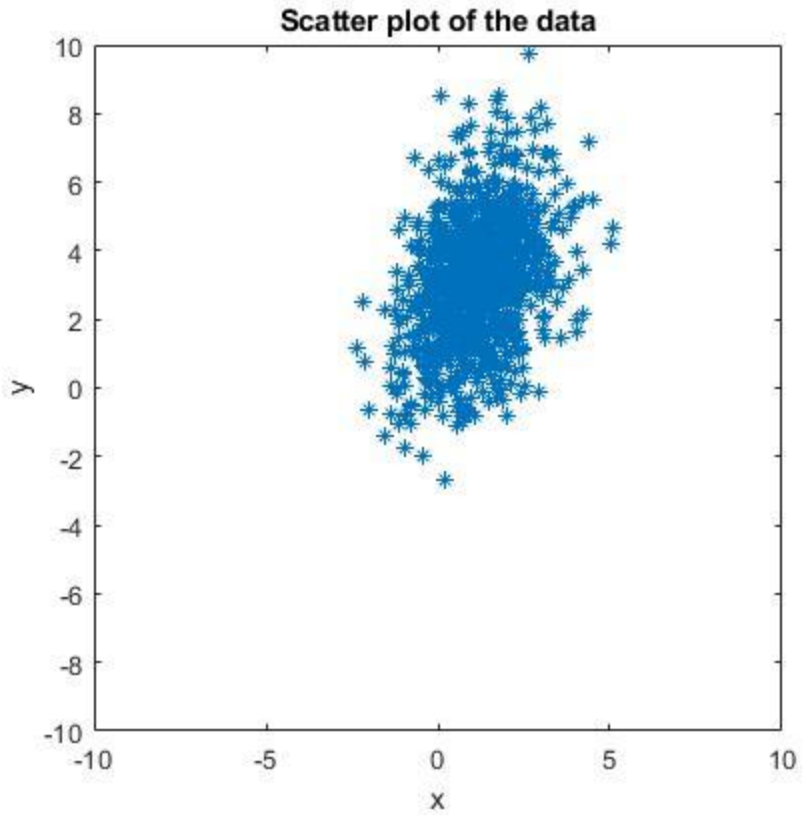
```
clear all
clear
clc

mu =[1.2 3.1];
Sigma=[1.2 0.7 ; 0.7 3.3];
rng('default') % For reproducibility

% mvnrnd returns an m-by-d matrix R of random vectors sampled
% from m separate d-dimensional multivariate normal distributions,
```

```
% with means and covariances specified by mu and Sigma, respectively.
% Each row of R is a single multivariate normal random vector.[5]
Samples = mvnrnd(mu,Sigma,1000)
```

```
Samples =

    1.7890    4.5892
    3.2089    3.1341
   -1.2744    0.9758
    2.1445    2.5085
    1.5492    4.2825
   -0.2325    0.9412
    0.7250    1.0144
    1.5753    4.2593
    5.1199    4.6666
    4.2338    5.4846
     •
     •
     •
```

b)

```
figure(1)
plot(Samples(:,1),Samples(:,2),'*')
%for equal axes
axis equal
title('Scatter plot of the data')
xlim([-10 10])
ylim([-10 10])
xlabel('x')
ylabel('y')
```



c)

```
% det(Sigma) returns the determinant of square matrix of sigma [6].
D = det(Sigma);
% trace(Sigma) calculates the sum of the diagonal elements of matrix of
Sigma[7]:
T = trace(Sigma);
fprintf(' The determinant of sigma is: %0.3f \n', D)
fprintf(' The trace of sigma is: %0.3f', T)


The determinant of sigma is: 3.470
The trace of sigma is: 4.500
```

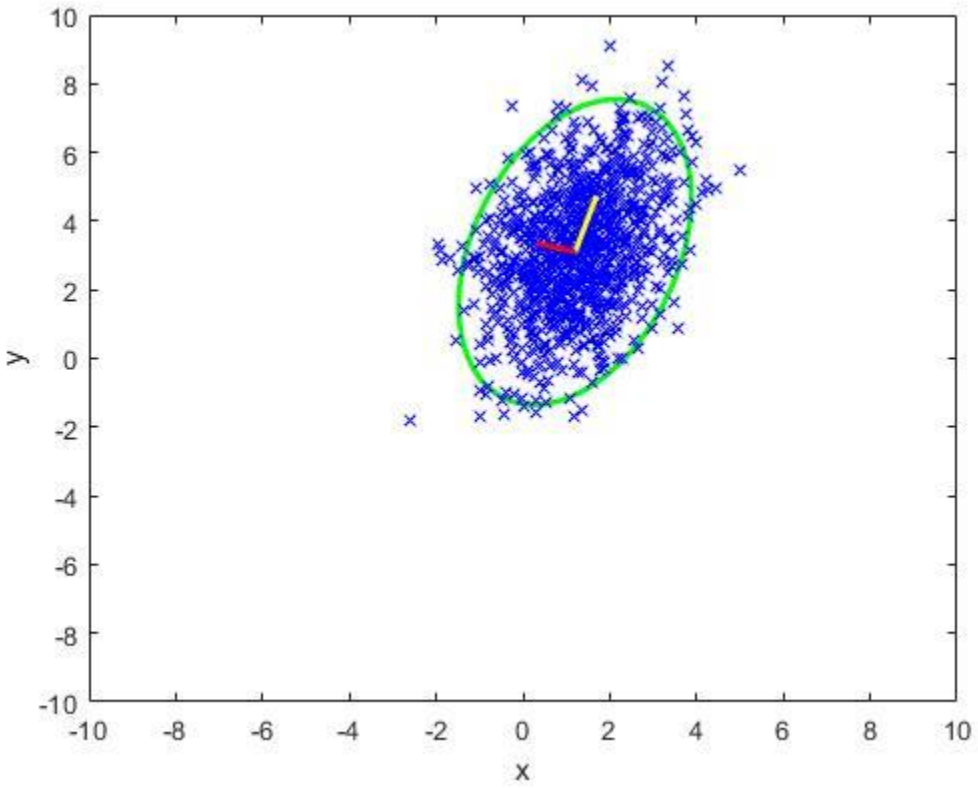
The matrix ∑ is a positive definite matrix because the determinant of the matrix det(Sigma) > 0, a positive value and elements in the matrix are positive.

d)

```
Sigma=[1.2 0.7 ; 0.7 3.3];
[eigenvec, eigenval] = eig(Sigma);

eigenvec =
-0.9571 0.2898
0.2898  0.9571
eigenval =
0.9881 0
0      3.5119
```

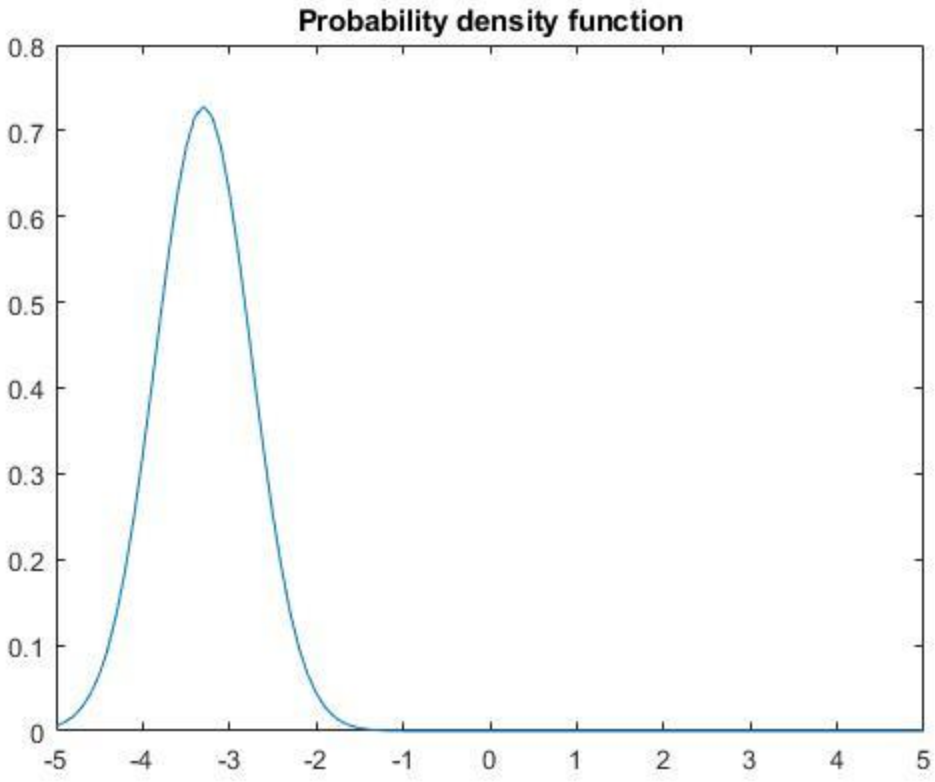
```
%ellipse
%This code is referenced from Mathworks discussion forum. The details
citation are given in reference section of the report [8][9].

[largest_eigenvec_ind_c, r] = find(eigenval == max(max(eigenval)));
largest_eigenvec = eigenvec(:, largest_eigenvec_ind_c);
% Get the largest eigenvalue
largest_eigenval = max(max(eigenval));
% Get the smallest eigenvector and eigenvalue
if(largest_eigenvec_ind_c == 1)
smallest_eigenval = max(eigenval(:,2));
smallest_eigenvec = eigenvec(:,2);
else
smallest_eigenval = max(eigenval(:,1));
smallest_eigenvec = eigenvec(1,:);
end
% Angle between the x-axis and the largest eigenvector
angle = atan2(largest_eigenvec(2), largest_eigenvec(1));
% This angle is between -pi and pi.
% Making the angle between 0 and 2pi
if(angle < 0)
angle = angle + 2*pi;
end
% Coordinates of the data mean
avg = [1.2 3.1];
chisquare_val = 2.4477;
theta_grid = linspace(0,2*pi);
phi = angle;
X0=avg(1);
Y0=avg(2);
a=chisquare_val*sqrt(largest_eigenval);
b=chisquare_val*sqrt(smallest_eigenval);
% Ellipse in x and y coordinates
ellipse_x_r = a*cos( theta_grid );
ellipse_y_r = b*sin( theta_grid );
% Define a rotation matrix
R = [ cos(phi) sin(phi); -sin(phi) cos(phi) ];
% Rotating the ellipse to some angle phi
r_ellipse = [ellipse_x_r;ellipse_y_r]' * R;
% Plotting the error ellipse
figure(2)
plot(r_ellipse(:,1) + X0,r_ellipse(:,2) + Y0,'-',"Color",'g',"LineWidth",2)
hold on;
% Plotting the scatter of the data
mu =[1.2 3.1];
Samples = mvnrnd(mu,Sigma,1000);
plot(Samples(:,1), Samples(:,2),'x',"Color",'b')
%scale of both axes are equal so that the true shape of the distribution is
visible
xlim([-10 10])
ylim([-10 10])
hold on;
% Plot the eigenvectors
quiver(X0, Y0, largest_eigenvec(1)*sqrt(largest_eigenval), ...16
largest_eigenvec(2)*sqrt(largest_eigenval), 'yellow', 'LineWidth',2);
```

```
quiver(X0, Y0, smallest_eigenvec(1)*sqrt(smallest_eigenval), ...
smallest_eigenvec(2)*sqrt(smallest_eigenval), 'r', 'LineWidth',2);
hold on;
xlabel('x');
ylabel('y');
```

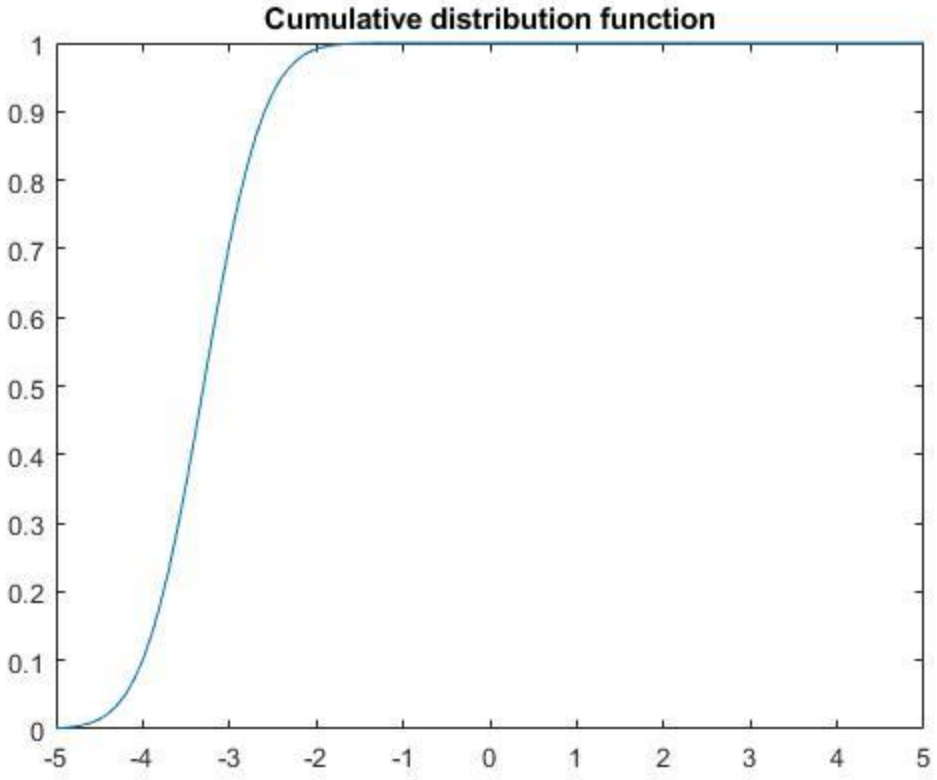


e)

```
% y = normpdf(x,mu,sigma) returns the pdf of the normal distribution
% with mean mu and standard deviation sigma, evaluated at the values in x.
mu= -3.3;
x = (-5:0.1:5);
var=0.3;
std = sqrt(var);
y = normpdf(x,mu,std);
figure(3)
plot(x,y)
title('Probability density function')
```




```
% p = normcdf(x,mu,sigma) returns the cdf of the normal distribution
% with mean mu and standard deviation sigma, evaluated at the values in x.
yy = normcdf(x,mu,std);
figure(4)
plot(x,yy)
title('Cumulative distribution function')
```



References:

[1] David Moore, W. I. (n.d.). *The Basic Practice of Statistics.*

[2] Lutu, P. E. (n.d.). The use of Histogram Analysis to Support Fast Selection of Predictive Features for Data Stream Mining. *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD).* IEEE.

[3] Wikipedia. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Lilliefors_test

[4] Mathworks.com. (n.d.). Retrieved from https://www.mathworks.com/help/stats/lillietest.html

[5] Mathworks.com. (n.d.). Retrieved from https://www.mathworks.com/help/stats/mvnrnd.html

[6] Mathworks.com. (n.d.). Retrieved from https://www.mathworks.com/help/matlab/ref/det.html

[7] Mathworks.com. (n.d.). Retrieved from https://www.mathworks.com/help/matlab/ref/double.trace.html

[8] Mathworks.com. (n.d.). Retrieved from https://ww2.mathworks.cn/matlabcentral/answers/299648-confidence-ellipsoid-from-eigenvalues

[9] Mathworks.com. (n.d.). Retrieved from https://www.mathworks.com/matlabcentral/answers/823870-check-if-point-lies-inside-error-ellipse