1. Download this dataset. It doesn't need to be normalized because each attribute is valued between 1 and 10.

```
%Ques1
%Reading data from csv file
Data = xlsread('\\clusterfsnew.ceas1.uc.edu\students\gangamrh\desktop\breast-cancer-wisconsin.data.csv');
%Filling the missing data by using KNN method
input = knnimpute(Data)
```

2. Select 500 records randomly from these dataset for training and keep the remaining 199 for the test set

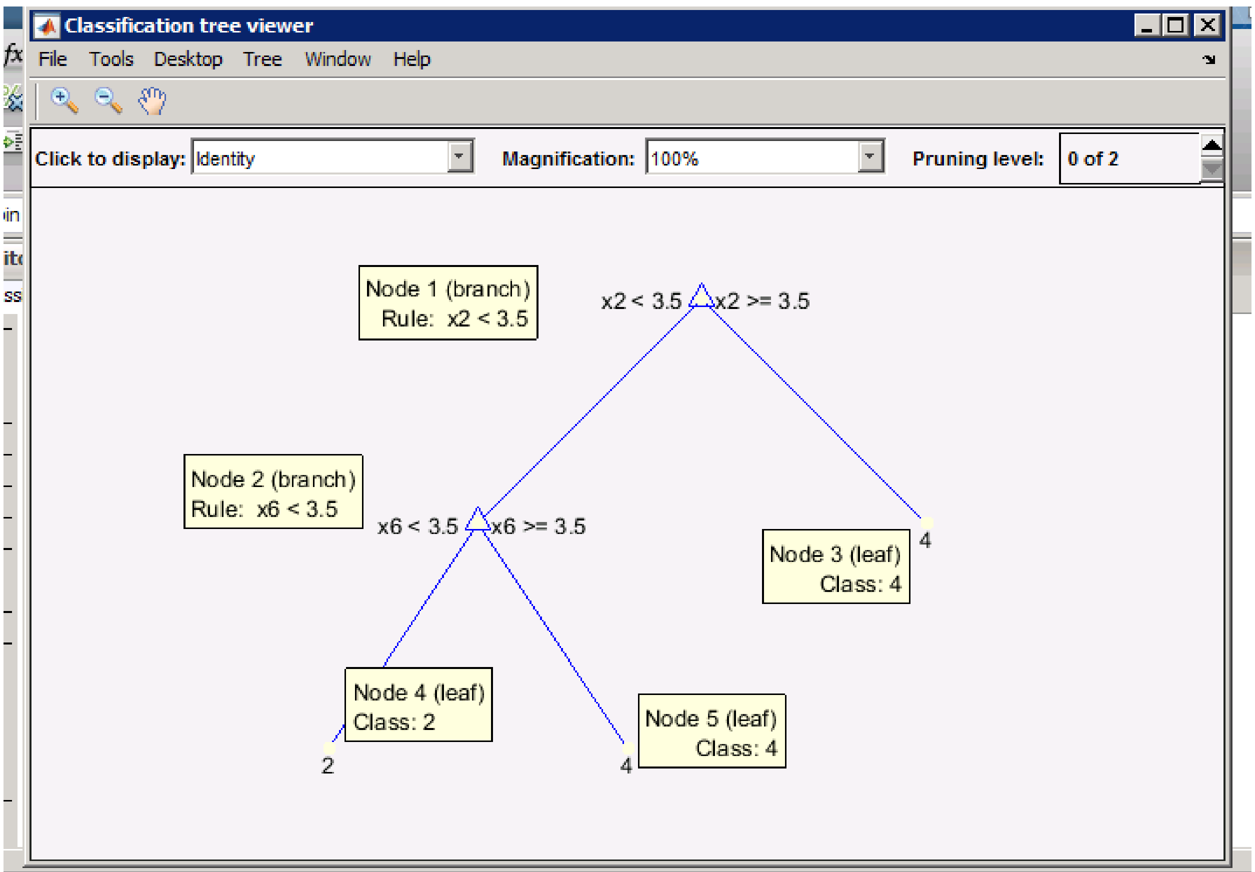
```
%Ques2
%sorts randomly
% Splitting data into 2 sections randomly. Training and Testing
random_in = randperm(699);
training = input(random_in(1:500),1:11);
Testing = input(random_in(501:699),1:11);

training_features = training(:,2:10);
training_ClassLabels= training(:,11);
```

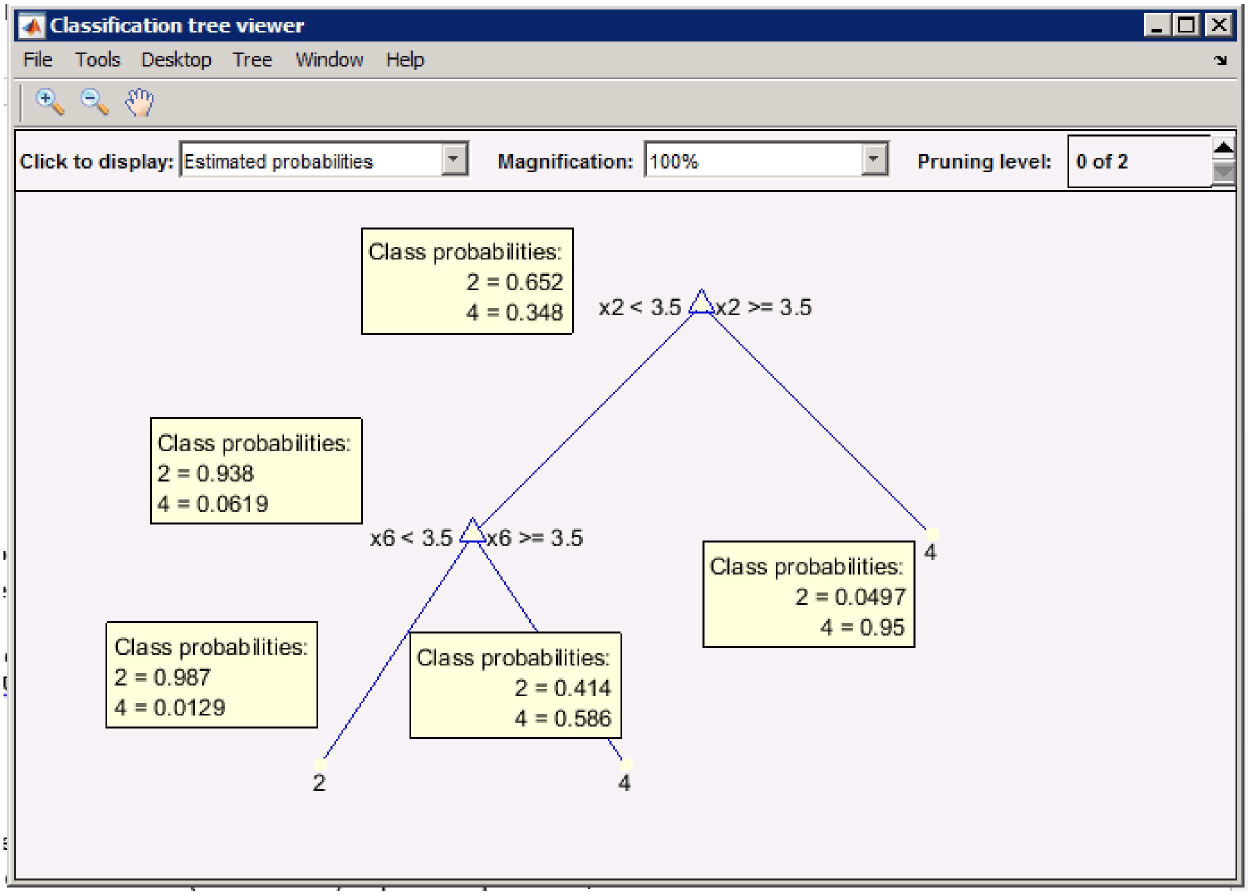
3. Create a decision tree with this dataset with the constraint that every leaf node has at least 25 data records. (**Submit This**) From this decision tree report those rules whose leaf nodes have at least 75% class purity. Include rule purity number with each reported rule.

```
%Ques3
%Creates decision tree
dtr = fitctree(training_features, training_ClassLabels, 'MinLeafSize', 25)
 %Displays decision tree
 view(dtr,'Mode','graph')
%Calculation of purity
%Stores branch node or leaf node status
 Branch_Node = dtr.IsBranchNode;
%impurity at each node is stored here
Impurity = dtr.NodeError;
No_oF_Nodes = dtr.NumNodes;
k = 1;

%purity  calculation at leaf node level
for i = 1:dtr.NumNodes
if ((Branch_Node(i) ~= 1) && (Impurity(i)<=0.25))
Purity(k) = (1 - Impurity(i))*100;
LeafNodes(k) = i;
k = k + 1;
```

```
end
end

%Displaying purity and Leaf node number


    tab = table(LeafNodes,Purity)
```

X2 Uniformity of Cell Size
X6 - Bare Nuclei






```
Output.



tab =

     LeafNodes          Purity
     _________     ____________________

     3     4         95.031       98.70
```

To come to Node2 x2>=3.5
Node 5 x2<3.5 and x6>=3.5

Where
X2 Uniformity of Cell Size
X6 - Bare Nuclei

4. Use the test dataset to predict the class labels of its records using the decision tree model created in #3 above. (**Submit This**) Report the precision, recall and F1 metrics of this classifier based on the actual and predicted labels of the test dataset.

```
Testing_features = Testing(:,2:10);
Testing_ClassLabels= Testing(:,11);

 Test_labels = predict(dtr, Testing_features)
%Calculate_Quality calculates TP,TN,FP,FN, Accuracy,Precision,Recall.
%Calculate_Quality function code is attached end
[F1,TP,TN,FP,FN,Precision,Recall] = Calculate_Quality(
Test_labels,Testing_ClassLabels)

%[F1,Precision,Recall] = Calculate_Accuracy1(
Test_labels,Testing_ClassLabels);

%creating table and writing F Measure Accuracy precision and recall to
%table
 tab = table(F1,Precision,Recall)
 display(tab)


tab =

      F1        Precision     Recall
    _______     _________     ______

    0.94942     0.93467       0.976
```

5. Use the fitcsvm function of Matlab to train a SVM model for the 500 records of the training set. Use ‘KernelFunction’ parameter with value ‘RBF’ to tell the trainer to use the Radial Basis Function as the non-linear transformation of the data space. Use this model to test the 199 records of the test dataset. (**Submit This**) Report the precision, recall, and F1 metrics of this classifier

```
%ques5
%Creates SVM
 SVMModel =
fitcsvm(training_features,training_ClassLabels,'Standardize',true,'KernelFunc
tion','RBF','KernelScale','auto');
%Predicts
 Test_labels_SVM = predict(SVMModel, Testing_features)
[F1_SVM,Precision_SVM,Recall_SVM] = Calculate_Accuracy1(
Test_labels_SVM,Testing_ClassLabels);

[F1_SVM, TP_SVM,TN_SVM,FP_SVM,FN_SVM,Precision_SVM,Recall_SVM ] =
Calculate_Quality( Test_labels_SVM,Testing_ClassLabels)

%creating table and writing F Measure Accuracy precision and recall to
%table
```

```
tab = table(F1_SVM,Precision_SVM,Recall_SVM)
display(tab)
```

```
tab =

    F1_SVM     Precision_SVM    Recall_SVM
    _______    _____________    __________

    0.96552    0.95477          0.97674
```

6.(**Submit This**) Compare and contrast the performance metrics obtained in #4 and #5 above. Give reasons for the differences that you observe.

Metrics for decision tree are.

```
      F1        Precision    Recall
    _______    _________    ______

    0.94942    0.93467      0.976
```

Metrics for SVM are

```
    F1_SVM     Precision_SVM    Recall_SVM
    _______    _____________    __________

    0.96552    0.95477          0.97674
```

From above metrics F1, precision and recall are better for SVM. From this we can say that SVM is more accurate then decision tree.

7.(**Submit This**) Assume the cost of making a correct diagnosis is 0; the cost of predicting an actual benign case as malignant is 10 units, and the cost of predicting an actual malignant case as benign is 30 units. Use the results obtained in #4 and #5 above and determine the cost of misclassification for each type of classifier. Show your work for arriving at your answers.

```
%Ques7
%cost of false negative prediction and cost of false positive prediction
 FN_Cost = 10;
 FP_Cost = 30;
 %calaculate cost function calcualtes cost of wrong prediction
 cost_SVM =
Calculate_Cost(F1_SVM,TP_SVM,TN_SVM,FP_SVM,FN_SVM,FN_Cost,FP_Cost)
```

```
 cost_Tree = Calculate_Cost(F1,TP,TN,FP,FN,FN_Cost,FP_Cost)
 %storing cost in table and displaying them
 tab = table(cost_SVM, cost_Tree)
 display(tab)
```

Cost is calculated by formula cost = FP*FP_Cost+FN*FN_Cost

```
function cost = Calculate_Cost( F1_SVM, TP_SVM,TN_SVM,FP_SVM,FN_SVM,
FN_Cost,FP_Cost )
%calculating cost
%   Detailed explanation goes here
%FP multiplied by FP_Cost+ FN negative multiplied by FN cost
cost = (FP_SVM * FP_Cost) + (FN_SVM * FN_Cost)
end
```

```
Output:
tab =

    cost_SVM    cost_Tree
    ________    _________

    150       190
```

8. (**Submit This**) Pick one record from the test set that is misclassified by the decision tree model. Find its 3 nearest neighbors, using Euclidean distance, in the training dataset. Show the query instance and the nearest neighbors retrieved. Find the class label that this instance should get using the KNN method? Repeat the above steps for 1, 5, and 7 nearest neighbors. Comment on the class labels that you get.

```
%Missclassifier index calculation. Misclassifier index returns it
 Misclassifer = misclasifier_index(Test_labels,Testing_ClassLabels)
 display(Testing(Misclassifer));


%generating fitcknn model
KNNModel3 =
fitcknn(training_features,training_ClassLabels,'NumNeighbors',3,'Standardize'
,1, 'Distance', 'euclidean');
%Predicting with fitcknn
With_KNN_Labels3 = predict(KNNModel3, Testing_features(Misclassifer,:));
%calaculating neighbours and distance
[Neighbours3,EuclideanDistances3]=knnsearch(training_features,Testing_feature
s(Misclassifer,:),'k',3,'distance','euclidean');
```

```
K3TrainingData = training(Neighbours3,:);
display('misclassified record with id')
%printing in output
display(Testing(Misclassifer,:));
display(K3TrainingData);
display(EuclideanDistances3);
```

```
misclassifier

ans =

     673637          3          1          1          1          2          5          5          1          1          2
```

```
K3TrainingData =

      128059          1          1          1          1          2          5          5          1          1          2
     1133136          3          1          1          1          2          3          3          1          1          2
     1173235          3          3          2          1          2          3          3          1          1          2
```

```
EuclideanDistances3 =

    2.0000    2.8284    3.6056
```

```
%generating fitcknn model
KNNModel1 =
fitcknn(training_features,training_ClassLabels,'NumNeighbors',1,'Standardize'
,1, 'Distance', 'euclidean');
%Predicting with fitcknn
With_KNN_Labels1 = predict(KNNModel1, Testing_features(Misclassifer,:));
%calaculating neighbours and distance
[Neighbours1,EuclideanDistances1]=knnsearch(training_features,Testing_feature
s(Misclassifer,:),'k',1,'distance','euclidean');
K1Neighbours = training((Neighbours1),:);
display(Testing(Misclassifer,:));
display(K1Neighbours);
display(EuclideanDistances1);
```

```
ans =

     673637           3           1           1           1           2           5           5           1           1           2


K1Neighbours =

     128059           1           1           1           1           2           5           5           1           1           2


EuclideanDistances1 =

     2
```

```
%generating fitcknn model for 5
KNNModel5 =
fitcknn(training_features,training_ClassLabels,'NumNeighbors',5,'Standardize'
,1, 'Distance', 'euclidean');
With_KNN_Labels5 = predict(KNNModel5, Testing_features(Misclassifer,:));
%calaculating neighbours and distance

[Neighbours5,EuclideanDistances5]=knnsearch(training_features,Testing_feature
s(Misclassifer,:),'k',5,'distance','euclidean');
K5Neighbours = training((Neighbours5,:));
display(Testing(Misclassifer));
display(K5Neighbours);
display(EuclideanDistances5);
```

```
ans =

     673637           3           1           1           1           2           5           5           1           1           2


K5Neighbours =

     128059           1           1           1           1           2           5           5           1           1           2
    1133136           3           1           1           1           2           3           3           1           1           2
    1173235           3           3           2           1           2           3           3           1           1           2
    1015425           3           1           1           1           2           2           3           1           1           2
    1241232           3           1           4           1           2           4           3           1           1           2


EuclideanDistances5 =

    2.0000    2.8284    3.6056    3.6056    3.7417
```

```
%generating fitcknn model
KNNModel7 =
fitcknn(training_features,training_ClassLabels,'NumNeighbors',7,'Standardize'
,1, 'Distance', 'euclidean');
With_KNN_Labels7 = predict(KNNModel7, Testing_features(Misclassifer,:));
%calaculating neighbours and distance
```

```
[Neighbours7,EuclideanDistances7]=knnsearch(training_features,Testing_feature
s(Misclassifer,:),'k',7,'distance','euclidean');
K7Neighbours = training((Neighbours7),:);
display(Testing(Misclassifer,:));
display(K7Neighbours);
display(EuclideanDistances7);
```

```
     673637          3          1          1          1          2          5          5          1          1          2


K7Neighbours =

     128059          1          1          1          1          2          5          5          1          1          2
    1133136          3          1          1          1          2          3          3          1          1          2
    1173235          3          3          2          1          2          3          3          1          1          2
    1015425          3          1          1          1          2          2          3          1          1          2
    1241232          3          1          4          1          2          4          3          1          1          2
    1255384          3          2          2          3          2          3          3          1          1          2
    1173681          3          2          1          1          2          2          3          1          1          2


EuclideanDistances7 =

    2.0000    2.8284    3.6056    3.6056    3.7417    3.7417    3.7417
```

```
%generating fitcknn model
KNN_Prediction = table(Testing_ClassLabels(Misclassifer),
Test_labels(Misclassifer),
With_KNN_Labels3,With_KNN_Labels1,With_KNN_Labels5, With_KNN_Labels7);
display(KNN_Prediction);
```

KNN_Prediction =

Var1	Var2	With_KNN_Labels3	With_KNN_Labels1	With_KNN_Labels5	With_KNN_Labels7
2	4	2	2	2	2

Decision tree wrongly predicted for 673637 as class 2. KNNmodel correctly predicted it as class 4.

Userdefined Functions are:

```
function [Misclassifer ] = misclasifier_index( PredictLabels,TestLabels)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
```

```
TP = 0
TN = 0
FP = 0
FN = 0

for i=1:length(PredictLabels)
     if PredictLabels(i)==2 && TestLabels(i)==2
         TP = TP + 1
     elseif PredictLabels(i)==2 && TestLabels(i)==4
          FP = FP + 1
          Misclassifer =i;
       break;
     elseif PredictLabels(i)==4 && TestLabels(i)==2
          FN = FN + 1
          Misclassifer =i;
          break;
     else
         TN = TN + 1
     end
end


function cost = Calculate_Cost( F1_SVM, TP_SVM,TN_SVM,FP_SVM,FN_SVM,
FN_Cost,FP_Cost )
%calculating cost
%   Detailed explanation goes here
%FP multiplied by FP_Cost+ FN negative multiplied by FN cost
cost = (FP_SVM * FP_Cost) + (FN_SVM * FN_Cost)
end
```