Assignment 3

Q1) Ans

Given dataset contains 10 attributes (Sample code number, Clump Thickness, Uniformity of Cell, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, Mitoses) and 1 class label (either benign or malignant)

Applied **standardizeMissing** inbuilt function in matlab to identify the missing values in the given dataset.

Found missing value in Bare Nuclei attribute. Replaced all missing values in it with mean of the non-missing values.

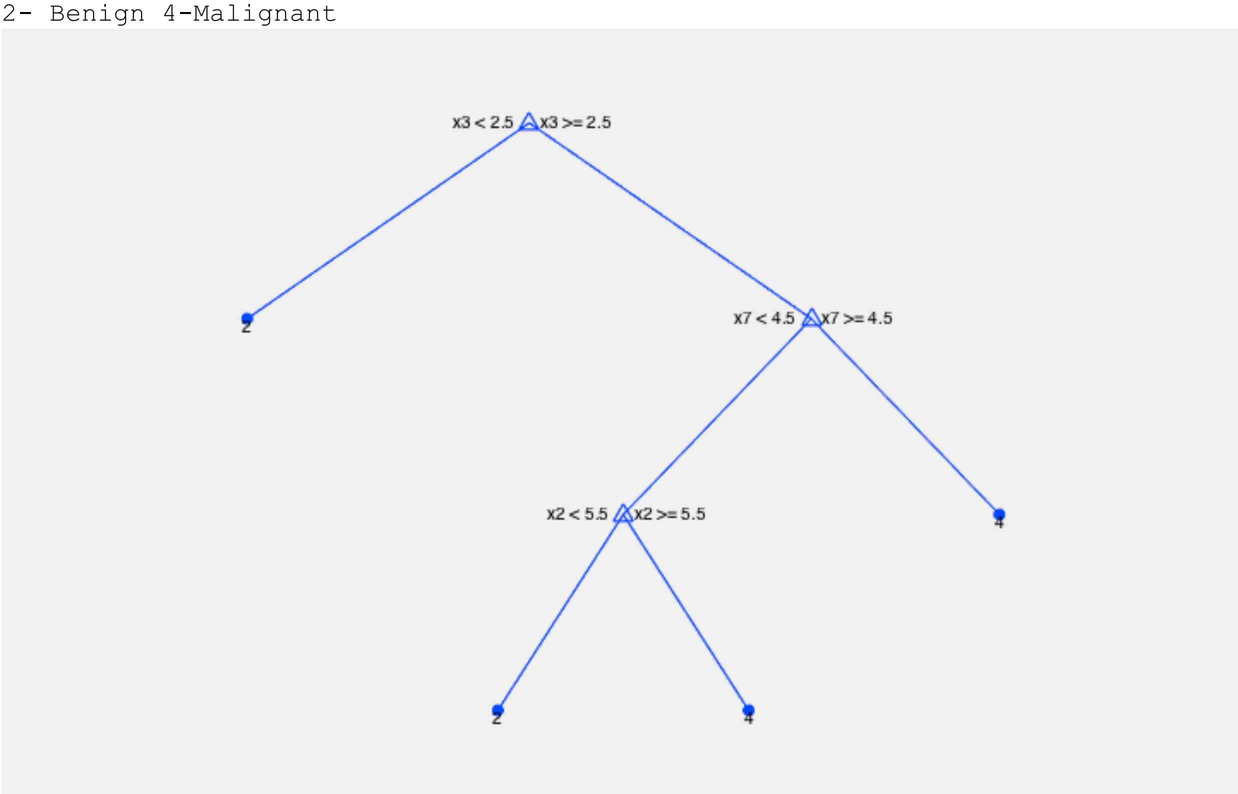
Q2) Ans

The data is cleaned and partitioned into training(500 records) and test data(199 records)

Q3) Ans

Decision tree graph

X3- Uniformity of Cell, X7- Bare Nuclei X2- Clump Thickness
2- Benign 4-Malignant



Decision tree rules whose leaf nodes have at least 75% class purity

Decision tree Rules and Purity Number report

Rule	PurityNumber
'Uni_Celsize < 3.500000'	96.615
'BrNcle < 4.500000'	97.644
'BrNcle < 4.500000'	98.176

Q4) Ans

the precision, recall and F1 metrics of this classifier based on the actual and predicted labels of the test dataset.

Precision	Recall	F1Score
0.887323943661972	0.851351351351351	0.868965517241379

Q5) Ans

Precision , recall and F1Score of SVM model

Precision	Recall	F1Score
0.972222222222222	0.945945945945946	0.958904109589041

Q6)Ans

From the above results, we observe there is more performance results from SVM Model. SVMs often do take a long time to train, this is especially true when the choice of kernel and particularly regularization parameter means that almost all the data end up as support vectors. For non linear data, Decision tree classification give impure results compared to SVM model. SVM uses RBF kernel function that classifies the non linear data with high performance.

Q7)Ans

Cost when using Decision tree - 520,
Cost when using SVM Model – 110

From the results of Confusion matrix(TP,TN,FP,FN), SVM gives proper prediction about the class labels, False positive and False Negative numbers will be less. So, the cost of the SVM model will be less than cost when using decision tree.

Q8) Ans

Misclassified Record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888523	4	4	4	2	2	3	2	1	1	2

3 nearest neighbors of give record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888820	5	10	10	3	7	3	8	10	2	4
888169	3	2	2	1	4	3	2	1	1	2
896404	2	1	1	1	2	1	3	1	1	2

Comment: 3nearest neighbors gives the majiory class lable as 2. It would classify record as 2 when it falls near the these neighnors

Misclassified Record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888523	4	4	4	2	2	3	2	1	1	2

1 nearest neighbors of give record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888820	5	10	10	3	7	3	8	10	2	4

Comment: 1nearest neighbors gives class label as 4. Its supports decision tree classification.

Misclassified Record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888523	4	4	4	2	2	3	2	1	1	2

5 nearest neighbors of give record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888820	5	10	10	3	7	3	8	10	2	4
888169	3	2	2	1	4	3	2	1	1	2
896404	2	1	1	1	2	1	3	1	1	2

897172	2	1	1	1	2	1	2	1	1	2
897471	4	8	6	4	3	4	10	6	1	4

Comment: 5nearest neighbors gives majority class label as 2. Knnsearch gives the results against decision tree.

Misclassified Record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml_Nucli	Mitoses	Class
888523	4	4	4	2	2	3	2	1	1	2

7 nearest neighbors of give record

S_CodeNo	Clmp_Thikns	Uni_Celsize	Uni_Celshp	Marg_Adhsn	SEpit_Celsize	BrNcle	Blnd_Chrmatin	Nrml _Nucli	Mitoses	Class
888820	5	10	10	3	7	3	8	10	2	4
888169	3	2	2	1	4	3	2	1	1	2
896404	2	1	1	1	2	1	3	1	1	2
897172	2	1	1	1	2	1	2	1	1	2
897471	4	8	6	4	3	4	10	6	1	4
897471	4	8	8	5	4	5	10	4	1	4
877943	3	10	3	10	6	10	5	1	4	4

Comment: 7nearest neighbors gives majority class label as 4. Knnsearch gives the results in support of decision tree classification.

Code:

Main programs:

```
% Program1
filename = '/breast-cancer-wisconsin.data';
d = fopen(filename);
while ~feof(d)
    cell=textscan(d,'%s %s %s %s %s %s %s %s %s %s %s', 'Delimiter',',');
    fgetl(d);
end
fclose(d);

% foramting data in datacell array
for idx=1:length(cell)
    datacell(:,idx)=[cellstr(cell{1,idx})];
end

% converting datacell into Table to standardize the missing values
T=cell2table(datacell);
```

```
T=standardizeMissing(T,{'?','?6',''},'DataVariables',{'datacell7'});
cell=table2cell(T);

% calculating mean for attribute values excluding missing values
TF = ismissing(T,{'' '.' '?' '?6'});
T2 = T(~any(TF,2),:);
cell2=table2cell(T2);
dble=[str2num(char(cell2(:,7)))];
meandble=ceil(mean(dble));

% replacing missing values with mean of the attribute
emptyIndex = cellfun(@isempty,cell);
cell(emptyIndex) = {num2str(meandble)};

data=cellfun(@str2num,cell);
cell=num2cell(data);

% Program2
% Randomly splitting dataset into train and test data.
[TrainData,TestData]=DataSetPartn(cell,500,199);

% Program3
objDTree=BCWDecTree(TrainData,25);
objDTree.viewDecisionTree();

ReportTable=objDTree.reportRulesofDtree();
fprintf('\nDecision tree Rules and PurityNumber report\n');
disp(ReportTable);

% Program4
[ActualClassLabels,PredictedClassLabels]=objDTree.predictClassLabels(TestData
);
[Precision,Recall,F1Score,TP_Dtr,TN_Dtr,FP_Dtr,FN_Dtr]=...

objDTree.calculateConfusionMatrix(ActualClassLabels,PredictedClassLabels);

  Performance{1,1}=Precision;
  Performance{1,2}=Recall;
 Performance{1,3}= F1Score;
  T= cell2table(Performance,'VariableNames',...
              {'Precision' 'Recall' 'F1Score'});
          disp(T);
% Program5
objSVMModel=BCWSVMModel(TrainData);

[ActualClassLabels_SVM,PredictedClassLabels_SVM]=objSVMModel.predictClassLabe
ls(TestData);
 [Precision_SVM,Recall_SVM,F1Score_SVM,TP_svm,TN_svm,FP_svm,FN_svm]=...

objSVMModel.calculateConfusionMatrix(ActualClassLabels_SVM,PredictedClassLabe
ls_SVM);

  fprintf('\n Precisio, recall and F1Score of SVM model\n')
  Performance{1,1}=Precision_SVM;
  Performance{1,2}=Recall_SVM;
 Performance{1,3}= F1Score_SVM;
```

```
  T= cell2table(Performance,'VariableNames',...
              {'Precision' 'Recall' 'F1Score'});
          disp(T);

%Program 7
DTreeCost= TP_Dtr*0+ FN_Dtr*10+FP_Dtr*30+ TN_Dtr*0;
SVMCost= TP_svm*0+ FN_svm*10+FP_svm*30+ TN_svm*0;
fprintf('\n Cost when using Decision tree - %d,\n Cost when using SVM Model - 
%d\n',...
    DTreeCost,SVMCost);

%Program 8
for idx=1:length(ActualClassLabels)
    if(~(ActualClassLabels(idx,1)==PredictedClassLabels(idx)))
        break;
    end
end
% Finding misclassified record in test dataset
misClsTuple=TestData(idx,1:11);
knnDistanceMat=[3,1,5,7];

for idx=1:length(knnDistanceMat)
nearestNeighbourReport(TrainData,misClsTuple,knnDistanceMat(idx));
end
```

DataSetPartn Function

```
function[TrainData,TestData]=DataSetPartn(dataset,traindatalength,validationd
atalength)
[rows,columns]=size(dataset);
randIdx=randperm(rows);

trainIdx=randIdx(1,1:traindatalength);

testIdx=randIdx(1,traindatalength+1:traindatalength+validationdatalength);

TrainData=dataset(trainIdx,:);
TestData=dataset(testIdx,:);

end
```

BCWDecTree Class

```
classdef BCWDecTree
    properties
        Dtr
    end

    methods
        function obj= BCWDecTree(dataSet,minLeafCondn)
```

```
            Features=dataSet(:,1:10);
            Features=cell2mat(Features);
            Class = dataSet(:,11);
            Class=cell2mat(Class);

            obj.Dtr=fitctree(Features,Class,'MinLeafSize', minLeafCondn);



        end
        function NumNodes= decTreeNumNodes(obj)
            NumNodes=obj.Dtr.NumNodes;
        end
        function viewDecisionTree(obj)
            view(obj.Dtr,'Mode','Graph');
        end

        function[T]=reportRulesofDtree(obj)
            cellindex=1;

Attrnames={'S_CodeNo','Clmp_Thikns','Uni_Celsize','Uni_Celshp','Marg_Adhsn','
SEpit_Celsize','BrNcle','Blnd_Chrmatin','Nrml_Nucli','Mitoses','Class'};

            for idx=1:length(obj.Dtr.IsBranchNode)


                if(~obj.Dtr.IsBranchNode(idx,1))
                    %                       Parent node of the child node
exists at index/2 of
                    %                       child node
                    branchnodeAtr=obj.Dtr.CutPredictor(floor(idx/2),1);
                    cutpoint=obj.Dtr.CutPoint(floor(idx/2),1);

                    if((1-obj.Dtr.NodeRisk(idx,1))>0.75)

                        NodeClass=cell2mat(obj.Dtr.NodeClass(idx,1));
                        AName=char(branchnodeAtr);

                        AttrIndex=str2num(strrep(AName,'x',''));


                        branchnodeAtr=Attrnames(1,AttrIndex);
                        if(cutpoint<NodeClass)
                            reportCell{cellindex,1}=sprintf('%s < %f',...
                                char(branchnodeAtr),cutpoint);

                            reportCell{cellindex,2}=(1-
obj.Dtr.NodeRisk(idx,1))*100;
                        elseif(cutpoint>=NodeClass)

                            reportCell{cellindex,1}=sprintf('%s >= %f ',...
                                char(branchnodeAtr),cutpoint);

                            reportCell{cellindex,2}=(1-
obj.Dtr.NodeRisk(idx,1))*100;
```

```
                        end
                        cellindex=cellindex+1;
                    end
                end

            end
%               reportCell=reportCell(~cellfun('isempty',reportCell));

            T= cell2table(reportCell,'VariableNames',{'Rule'
'PurityNumber'});

        end


function[ActualClassLabels,PredictedClassLabels]=predictClassLabels(obj,Datas
et)
            Features=Dataset(:,1:10);
            Features=cell2mat(Features);
            ActualClassLabels = Dataset(:,11);
            ActualClassLabels=cell2mat(ActualClassLabels);
            PredictedClassLabels = predict(obj.Dtr,Features);

        end



        function
[Precision,Recall,F1Score,TP,TN,FP,FN]=calculateConfusionMatrix(obj,TargetCla
ssLabels,Prediction)
            TP=0;FP=0;FN=0;TN=0;
            for idx=1:length(TargetClassLabels)
                if((TargetClassLabels(idx,1)==4) &&(Prediction(idx,1)==4))
                    TP=TP+1;
                elseif ((TargetClassLabels(idx,1)==4)
&&(Prediction(idx,1)==2))
                    FP=FP+1;           %FP(length(FP)+1,1)=Target(:,1);
                elseif ((TargetClassLabels(idx,1)==2)
&&(Prediction(idx,1)==4))
                    FN=FN+1;           %TN(length(TN)+1,1)=Target(:,1);
                elseif ((TargetClassLabels(idx,1)==2)
&&(Prediction(idx,1)==2))
                    TN=TN+1;
%FP(length(FP)+1,1)=Target(:,1);
                end
            end

            Precision=TP/(TP+FP);
            Recall=TP/(TP+FN);
            F1Score = 2*TP/(2*TP+FP+FN);

        end
    end
end
```

BCWSVMModel Class

```
classdef BCWSVMModel
    properties
        model,numNodes,infoGain,TP,TN,FP,FN
    end

    methods

        function obj= BCWSVMModel(dataSet)
            Features=dataSet(:,1:10);
            Features=cell2mat(Features);
            Class = dataSet(:,11);
            Class=cell2mat(Class);

obj.model=fitcsvm(Features,Class,'Standardize',true,'KernelFunction','RBF',..
.
                'KernelScale','auto');

        end

function[ActualClassLabels,PredictedClassLabels]=predictClassLabels(obj,Datas
et)
            Features=Dataset(:,1:10);
            Features=cell2mat(Features);
            ActualClassLabels = Dataset(:,11);
            ActualClassLabels=cell2mat(ActualClassLabels);

            PredictedClassLabels = predict(obj.model,Features);

        end
        function
[Precision,Recall,F1Score,TP,TN,FP,FN]=calculateConfusionMatrix(obj,TargetCla
ssLabels,Prediction)
            TP=0;FP=0;FN=0;TN=0;
            for idx=1:length(TargetClassLabels)
                if((TargetClassLabels(idx,1)==4) &&(Prediction(idx,1)==4))
                    TP=TP+1;
                elseif ((TargetClassLabels(idx,1)==4)
&&(Prediction(idx,1)==2))
                    FP=FP+1;           %FP(length(FP)+1,1)=Target(:,1);
                elseif ((TargetClassLabels(idx,1)==2)
&&(Prediction(idx,1)==4))
                    FN=FN+1;           %TN(length(TN)+1,1)=Target(:,1);
                elseif ((TargetClassLabels(idx,1)==2)
&&(Prediction(idx,1)==2))
                    TN=TN+1;
%FP(length(FP)+1,1)=Target(:,1);
                end
            end

            Precision=TP/(TP+FP);
            Recall=TP/(TP+FN);
            F1Score = 2*TP/(2*TP+FP+FN);

        end
```

```
    end
end
```

nearestNeighbourReport Function

```
function nearestNeighbourReport(TrainData,misClsTuple,N)
misCls=cell2mat(misClsTuple(1,1:10));
% Finding 3Nearest neighbours in Traindata
Features=TrainData(:,1:10);
Features=cell2mat(Features);

[Knnidx,dstnce]=knnsearch(Features,misCls,'K',N,'Distance','euclidean');

dataset=[];
classPredicted=cell2mat(misClsTuple(1,11));
for indx=1:length(Knnidx)
   classActuals(indx,1)=cell2mat(TrainData(Knnidx(1,indx),11));
   dataset{indx}= TrainData(Knnidx(1,indx),:);
end
for idx=1:length(dataset)
    datacell(idx,:)=[dataset{1,idx}];
end

format long;

fprintf('\n Misclassified Record \n');
 T= cell2table(misClsTuple,'VariableNames',...
              {'S_CodeNo' 'Clmp_Thikns' 'Uni_Celsize' 'Uni_Celshp'
'Marg_Adhsn' 'SEpit_Celsize'...
              'BrNcle' 'Blnd_Chrmatin' 'Nrml_Nucli' 'Mitoses' 'Class'});
 disp(T);

fprintf('\n %d nearest neighnours of give record\n',N);
  T= cell2table(datacell,'VariableNames',...
                        {'S_CodeNo' 'Clmp_Thikns' 'Uni_Celsize' 'Uni_Celshp'
'Marg_Adhsn' 'SEpit_Celsize'...
              'BrNcle' 'Blnd_Chrmatin' 'Nrml_Nucli' 'Mitoses' 'Class'});
 disp(T);




end
```