%% Loading Data
clear all
close all
c1c

data = xlsread('MLstudentboth.xlsx');

%% INITIAL STATISTICS ON KEY VARIABLES
Age = data(:,3);
G1 = data(:, 31);
G2 = data(:, 32);

agemean = mean(Age(:))
agestd = std(Age(:))

G1mean = mean(G1(:))
G1std = std(G1(:))

G2mean = mean(G2(:))
G2std = std(G2(:))

block_colour = [121 217 249] ./ 255;
line_colour = [252 12 255] ./ 255;
edge_colour = [45 253 234] ./ 255;
close all
figure;
G1x = histfit(G1,20);
G1x(1).FaceColor = block_colour;
G1x(2).Color = line_colour;
G1x(1).EdgeColor = edge_colour;
title('G1');
ylabel('');
set(gca,'YTick',[]);
xlim([0 25])
figure;
G2x = histfit(G2,15);
G2x(1).FaceColor = block_colour;
G2x(2).Color = line_colour;
G2x(1).EdgeColor = edge_colour;
title('G2');
ylabel('');
set(gca,'YTick',[]);
xlim([0 25])
figure;
Agex = histfit(Age,8);
% Comparing the method for partitioning training and test data:
% 1) Independent test set, 2) k fold cross validation and 3) out of bag methods

X = data(:, 1:31);
Y = data(:, 32);
%Train data
cvpart = cvpartition(Y,'holdout',0.2);
Xtrain = X(training(cvpart),:);
Ytrain = Y(training(cvpart),:);
Xtest = X(test(cvpart),:);
Ytest = Y(test(cvpart),:);
% Requires matlab version b
%1)Independent test data
bag = fitensemble(Xtrain,Ytrain,'Bag',1000,'Tree',...
    'Type','Classification')
%2)5 fold cross validation
cv = fitensemble(X,Y,'Bag',1000,'Tree',...
    'type','classification','kfold',5)

%Conclusion: K-fold reduces amount of variation greatly and also produces
%the lowest error. Error is lowest when the number of trees = 1000
% Establishing optimal k
for a = 5:1:10
    cv = fitensemble(X,Y,'Bag',1000,'Tree',...
        'type','classification','kfold',a)
    figure;
    plot(kfoldLoss(cv,'mode','cumulative'),'r.');
    xlabel('Number of trees');
    ylabel('Classification error');
    legend('Test','Cross-validation','Out of bag','Location','NE');
end

Page 2
% Conclusion: 5-fold has the lowest error and reduces time taken

%% Feature analysis to discover important variables in data for predicting G2
% Requires matlab version b
Mdl = fitctree(Xtrain,Ytrain,'PredictorSelection','curvature',...
    'Surrogate','on');
%Mdl
imp = predictorImportance(Mdl);

% Our factors we decide are important
a = [3 7 8 9 11 13 14 15 24 25 26 27 28 29 30 31]
figure;
impfit = bar(imp);
hold on
markers = scatter(a, imp(a), '*');
hold off
title('Predictor Importance Estimates for G2');
ylabel('Estimates');
xlabel('Predictors');
h = gca;
%PredictorNames = ['school' 'sex' 'age' 'address' 'famsize'...'Pstatus'
  'Medu' 'Fedu' 'Mjob' 'Fjob' 'reason' 'guardian' 'traveltime' 'studytime'
  'failures' 'schoolsup' 'famsup' 'paid' 'activities' 'nursery' 'higher'
  'internet' 'romantic' 'famrel' 'freetime' 'goout' 'Dalc' 'Walc'
  'health' 'absences' 'G1'] corresponding to x1-x31
block_colour = [121 217 249] ./ 255;
edge_colour = [45 253 234] ./ 255;
markers_colour = [252 12 255] ./ 255;
markers.MarkerEdgeColor = markers_colour;
impfit.FaceColor = block_colour;
impfit.EdgeColor = edge_colour;
h.XTickLabel = Mdl.PredictorNames;
h.XTickLabelRotation = 90;
h.TickLabelInterpreter = 'none';
set(h,'Xtick',[1:31])
xlim([0 32]);

%% RANDOM FORESTS: Establishing best parameters for prediction

% Reloading data for clarity
clear all
close all
c1c
data = xlsread('MLstudentboth.xlsx');
X2 = data(:, [3 7 8 9 11 13 14 15 24:31]);
Y2 = data(:, 32);
%Train data
%Creating Training and Test set using K-fold method, k=5 (i.e. 80% and 20%)
cvpart2 = cvpartition(Y2,'KFold',5);

indices = crossvalind('Kfold',Y2,5);
cp = classperf(Y2);
for i = 1:5
    test2 = (indices == i); train2 = ~test2;
    Xtrain2 = X2(train2,:);
    Xtest2 = X2(test2,:);
    Ytrain2 = Y2(train2,:);
    Ytest2 = Y2(test2,:);
end
% Number of Trees = 1000, Min leaf size increasing from 1 to 100
b=1
for b = 1:1:100

% tic/toc used to measure time taken
% Training the model
rng(1); % For reproducibility
tic
Mdl = TreeBagger(1000,Xtrain2,Ytrain2,'MinLeafSize',b,'OOBPrediction','On',...'
'Method','classification')
t3(b)= toc

%Prediction
prediction3 = Mdl.predict(Xtest2);

% Show difference between prediction and actual results of each student
prediction3 = str2double(prediction3);
check3 = (Ytest2 - prediction3);
disp(check3);
% Difference of prediction and actual results of each student for each iteration
e3(b) = sum(abs(check3));
end

% Plot to discover best choice of minimum leaf size by analysing
% 1)time and 2)error
StudentsMatlabCode.txt

% Defining b1 as a matrix for analysis
b3=[1:1:100];
% plot
figure;
plot(b3,t3)

figure;
plot(b3,e3)

% Results:
% 1) Time decreases exponentially as minimum leaf size increases, leveling out
% at minleafsize = 20 to a third of the time at minleafsize = 1
% 2) error is at its lowest when minimum leaf size is between 10 and 30

% Conclusion:
% Optimal minimum leaf size is between 20 to 30, therefore we have chosen:
% Minimum leaf size = 25

% Minimum leaf size = 25, number of trees increasing from 1 to 1000 in intervals of 20

for nTrees = 1:20:1000
    b4 = 25;
    % tic/toc used to measure time taken
    % Training the model
    rng(1); % For reproducibility
    tic
    Mdl = TreeBagger(nTrees,Xtrain2,Ytrain2,'MinLeafSize',b4,'OOBPrediction','On',...
        'Method','classification')
    t4(nTrees) = toc;
end

% Show difference between prediction and actual results of each student
prediction4 = Mdl.predict(Xtest2);

% Comparison to real results obtained by each student in test set
prediction4 = str2double(prediction4);
check4 = (Ytest2 - prediction4);
disp(check4);
% Difference of prediction and actual results of each student for each iteration
e4(nTrees) = sum(abs(check4));

% plot to confirm best choice of number of trees by analysing
% 1)time and 2)error
% Defining Number of Trees as a matrix for analysis
nTrees4 = [1:20:1000];
% t2 and e2 need to been 1x50 so zeros are removed

% plot
figure;
plot(nTrees4,t4)
figure;
plot(nTrees4,e4)

% Results:
% 1) Time increases linearly as the number of trees increase
% 2) error decreases exponentially and levels out after 50 but at its
% lowest and more stable after 400

% Conclusion:
% Optimal number of trees is at least 50, therefore we have chosen:
% Number of trees = 1000

% This seems too large from analysis but this corresponds to our intial
% analysis for the partitioning methods and time taken is not too large due
% to us having so few observations

% Now using our parameters: Number of Trees = 1000, Min leaf size 25
% Restarting main data loading and partitioning process for clarity
% WARNING: NEEDS TO BE RUN IN ORDER TO WORK CORRECTLY

clear all
close all
clc
data = xlsread('MLstudentboth.xlsx');
b=25;
nTrees=1000;

% K NEAREST NEIGHBOUR PARTITIONS

% Prediction for G2 from G1

% X contains columns 31 which are marks for G1
% Y contains column 32, which are marks for Grade 2

X3 = data(:, 31);
Y3 = data(:, 32);

% Creating Training and Test set using K-fold method, k=5 (i.e. 80% and 20%)
cvpart3 = cvpartition(Y3,'KFold',5);
indices3 = crossvalind('Kfold', Y3, 5);
cp = classperf(Y3);
for i = 1:5
    test3 = (indices3 == i); train3 = ~test3;
    Xtrain3 = X3(train3,:);
    Xtest3 = X3(test3,:);
    Ytrain3 = Y3(train3,:);
    Ytest3 = Y3(test3,:);
end

% Prediction for G2 from additional variables

%X contains columns 3, 7, 8, 9, 11, 13, 14, 15 and 24-31
%Y contains column 32, which are marks for Grade 2
X4 = data(:, [3 7 8 9 11 13 14 15 24:31]);
Y4 = data(:, 32);
%Train data
%Creating Training and Test set using K-fold method, k=5 (i.e. 80% and 20%)
cvpart4 = cvpartition(Y4,'KFold',5);

indices4 = crossvalind('Kfold', Y4, 5);
cp = classperf(Y4);
for i = 1:5
    test4 = (indices4 == i); train4 = ~test4;
    Xtrain4 = X4(train4,:);
    Xtest4 = X4(test3,:);
    Ytrain4 = Y4(train3,:);
    Ytest4 = Y4(test3,:);
end

% RANDOM FOREST PARTITIONS
% Predicting G2 from G1
X = data(:, 31);
Y = data(:, 32);
%Train data
%Creating Training and Test set using K-fold method, k=5 (i.e. 80% and 20%)
cvpart1 = cvpartition(Y,'KFold',5);

indices1 = crossvalind('Kfold', Y, 5);
cp = classperf(Y);
for i = 1:5
    test = (indices1 == i); train = ~test;
    Xtrain = X(train,:);
StudentsMatlabCode.txt

Xtest = X(test,:);
Ytrain = Y(train,:);
Ytest = Y(test,:);

end

% Predicting G2 from more variables
X2 = data(:, [3 7 8 9 11 13 14 15 24:31]);
Y2 = data(:, 32);
%Train data
%Creating Training and Test set using K-fold method, k=5 (i.e. 80% and 20%)
cvpart2 = cvpartition(Y2,'KFold',5);

indices2 = crossvalind('Kfold',Y2,5);
cp = classperf(Y2);
for i = 1:5
    test2 = (indices2 == i); train2 = ~test2;
    Xtrain2 = X2(train2,:);
    Xtest2 = X2(test2,:);
    Ytrain2 = Y2(train2,:);
    Ytest2 = Y2(test2,:);
end

%% RANDOM FORESTS MODEL

% Predicting G2 using G1
% Treebagger method for 1000 trees and Minimum leaf size 25
rng(1); % For reproducibility
% tic/toc used to measure time taken
% Training the model

tic
Mdl = TreeBagger(nTrees,Xtrain,Ytrain,'MinLeafSize',b,'OOBPrediction','On','
'Method','classification')
t1 = toc
%Prediction
prediction = Mdl.predict(Xtest);
% Show difference between prediction and actual results of each student
prediction = str2double(prediction);
check = (Ytest - prediction);
disp(check);
% Table to show classification error
G1table = tabulate(abs(check))
% Plot for prediction of G2 using G1 only and the difference between
% prediction and actual test scores
figure;
RFPLT1 = plot(prediction,'.');  
hold on  
RFPLT2 = plot(Ytest,'--');  
%hold on  
%plot(check);  
title({'Predicting G2 using only G1 results','1000 tress and minimum leaf size  
25'});  
xlim([0 205]);  
ylim([5 20]);  
xlabel('Students');  
ylabel('Score for G2');  
legend('Prediction','Actual results');  
predict_colour1 = [63 110 222] ./ 255;  
actual_colour = [149 125 155] ./ 255;  
RFPLT1.Color = predict_colour1;  
RFPLT2.Color = actual_colour;  

% Predicting G2 using more variables

% Treebagger method for 1000 trees and Minimum leaf size 25
% tic/toc used to measure time taken
% Training the model
rng(1); % For reproducibility
 tic  
Mdl2 = TreeBagger(nTrees,Xtrain2,Ytrain2,'MinLeafSize',b,'OOBPrediction','On',...  
'Method','classification')  
t2 = toc  
%view(Mdl.Trees{1},'Mode','graph')

% Show difference between prediction and actual results of each student
prediction2 = Mdl2.predict(Xtest2);

% Show difference between prediction and actual results of each student
prediction2 = str2double(prediction2);
check2 = (Ytest2 - prediction2);
disp(check2);
% Table to show classification error
G2table = tabulate(abs(check2))

% Plot for prediction of G2 using more variables and the difference between
% prediction and actual test scores
figure;
RFPLT3 = plot(prediction2,'.');  
hold on  
RFPLT4 = plot(Ytest2,'--');  
%hold on  
%plot(check2);  
xlim([0 205]);
StudentsMatlabCode.txt

title({'Predicting G2 using more variables','1000 trees and minimum leaf size 25'});
xlabel('Students');
ylabel('Score for G2');
legend('Prediction', 'Actual results');
ylim([5 20]);
predict_colour2 = [18 250 246] ./ 255;
RFPLOT3.Color = predict_colour2;
RFPLOT4.Color = actual_colour;

%%% K NEAREST NEIGHBOUR MODEL

% Fit a knn models with optimized hyperparameters (Number of Neighbors and Distance)
tic
Mdlknn1 = fitcknn(Xtrain3,Ytrain3,'OptimizeHyperparameters','auto',...    
    'HyperparameterOptimizationOptions',...    
    struct('AcquisitionFunctionName','expected-improvement-plus'))
t3 = toc
% Train a KNN model for the training set, Xtrain,Ytrain, default distance
% Euclidian
%Mdlknn1 = fitcknn(Xtrain3,Ytrain3,'NumNeighbors',5);

% Resubstitution loss - the fraction of misclassifications from the predictions of Mdl1
rloss1 = resubLoss(Mdlknn1)

% 5 fold cross validated classifier from mdl1
rng(1); % For reproducibility
CVMdknn1 = crossval(Mdlknn1,'KFold',5);

% Examine the cross validation loss
kloss1 = kfoldLoss(CVMdknn1);

% Predict the classification of a mark in G3 from G1 and G2.
mrk1 = mean(X3);
mrkClass1 = predict(Mdlknn1, Xtest3);

% Compare the prediction to the actual values of the testing set
StudentsMatlabCode.txt

Comp1=abs(mrkClass1-Ytest3);
% plot predictions vs actual values
figure;
kPLOT1 = plot(mrkClass1,'.');
hold on
kPLOT2 = plot(Ytest3,'--');
hold off
%plot(Comp1);
legend('Prediction','Actual results')
title({'Predicting G2 using G1 only','K = '})
xlabel('Students');
ylabel('Score for G2');
xlim([0 200]);
predict_colour3 = [168 40 249] ./ 255
kPLOT1.Color = predict_colour3;
kPLOT2.Color = actual_colour;

% G3table = tabulate(Comp1);

% Prediction for G2 from additional variables

% Train a KNN model for the training set, Xtrain,Ytrain
%Mdlknn2 = fitcknn(Xtrain4,Ytrain4);
tic
Mdlknn2 = fitcknn(Xtrain4,Ytrain4,'OptimizeHyperparameters','auto',...    
    'HyperparameterOptimizationOptions',...           
    struct('AcquisitionFunctionName','expected-improvement-plus'))
t4 = toc

%Mdlknn2.NumNeighbors = 17;

% Resubstitution loss - the fraction of misclassifications from the predictions of Mdl1
rloss2 = resubLoss(Mdlknn2)

% 5 fold cross validated classifier from mdl1
rng(1); % For reproducibility
CVMdknn2 = crossval(Mdlknn2,'KFold',5);

% Examine the cross validation loss
kloss2 = kfoldLoss(CVMdknn2);
% Predict the classification of a mark in G3 from G1 and G2.
mrk2 = mean(X4);
mrkClass2 = predict(Mdlknn2, Xtest4);

% Compare the prediction to the actual values of the testing set
Comp2 = abs(mrkClass2 - Ytest4);
% plot predictions vs actual values

figure;
kPLOT3 = plot(mrkClass2,'.');
hold on
kPLOT4 = plot(Ytest4,'--');
%plot(Comp2);
hold off
legend('Prediction','Actual results');
title({'Predicting G2 using more features', 'K='});
xlabel('Students');
ylabel('Score for G2');
predict_colour4 = [253 6 219] ./ 255;
kPLOT3.Color = predict_colour4;
kPLOT4.Color = actual_colour;

%
G4table = tabulate(Comp2);

%% Confusion matrix analysis i)
close all
% Random forests model
[Yfit,Sfit] = predict(Mdl2,Xtest2);
q1 = cellfun(@str2double,Yfit);
C1 = confusionmat(Ytest2, q1)

% Score from 5-19 correlate to 1-15 dimensions in matrix
for a = 1:1:15
B1 = C1(a,:);
figure;
bar(B1);
title('Accuracy of each classification score');
xlabel('Score (1-15 corresponds to 5-19 score');
ylabel('Estimate count');
end
%% Confusion matrix analysis ii)
% K-NN model
%markClass from earlier is equal to Yfit
close all
q2 = mrkClass2
C2 = confusionmat(Ytest4, q2)
    for a = 1:1:15
        B2 = C2(a,:);
        figure;
        bar(B2);
        title('Accuracy of each classification score');
        xlabel('Score (1-15 corresponds to 5-19 score');
        ylabel('Estimate count');
    end

%%

%% Comparison of confusion matrices for each model

% Sorting confusion matrices into two parts, for each score we measure how
% far out the prediction is from its correct value to create the upper and
% lower bounds of the error plots
C1edit = C1;
C1edit(logical(eye(size(C1edit)))) = 0
C1edit(C1edit > 0) = 1;
C1edit2 = tril(C1edit);
C1edit3 = triu(C1edit);

C2edit = C2;
C2edit(logical(eye(size(C2edit)))) = 0
C2edit(C2edit > 0) = 1;
C2edit2 = tril(C2edit);
C2edit3 = triu(C2edit);

for i = 1:1:15;
        C1min(i) = sum(C1edit2(i,:));
        C1max(i) = sum(C1edit3(i,:));
        C2min(i) = sum(C2edit2(i,:));
        C2max(i) = sum(C2edit3(i,:));

        Rrf1(i) = [C1min(:,i)];
        Rrf2(i) = [C1max(:,i)];
        Rk1(i) = [C2min(:,i)];
        Rk2(i) = [C2max(:,i)];
close all
for a = 1:1:15;
    Brf(a) = C1(a,a);
    Bk(a) = C2(a,a);
    C1total(a) = sum(C1(a,:));
    C2total(a) = sum(C2(a,:));
    BrfC1total(a) = (Brf(:,a)./C1total(:,a)).*100;
    BkC2total(a) = (Bk(:,a)./C2total(:,a)).*100;
end

% Plots in reverse order so array is flipped
    Brf2 = flip(Brf);
    Bk2 = flip(Bk);

% Plot bar comparison
    close all
    bars = [Brf2;Bk2];
    bars=rot90(bars);
    Xinp = [1:15; 1:15];
    figure;
    barplot = bar(bars);
    legend('Random Forests','K-NN')
    title('Number of correct classifications at each score');
    xlabel('Score');
    ylabel('Total number of correct estimates');
    xticklabels({'5','6','7','8','9','10','11','12','13','14','15','16','17','18','19'})
    ylim([0 20]);
    block_colour1 = [121 217 249] ./ 255;
    block_colour2 = [239 127 217] ./ 255;
    edge_colour1 = [45 253 234] ./ 255;
    barplot(1).FaceColor = block_colour1;
    barplot(2).FaceColor = block_colour2;

% Plot error bar comparison
    figure;
    h=gca;
    hold on
    errorbar1 = errorbar(1:15,[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0],[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0],Rrf1,Rrf2,'s')
    errorbar2 = errorbar(1:15,[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0],Rk1, Rk2, '.')
    errorbar2.Color = [254 84 222] ./ 255;  
    errorbar1.CapSize = 12;  
    errorbar2.CapSize = 5;  
    set(h,'Xtick',[1:31])  
    refl = reffline(0,0);
 StudentsMatlabCode.txt

%ref1.Color = 'black';
ylim([-6 6]);
xlim([0 16]);
hold off

title('Range of the estimates made by both models at each score');
legend('Random Forests', 'K-NN');
xlabel('Score');
xticklabels({'5','6','7','8','9','10','11','12','13','14','15','16','17','18','19'})

ylabel({'min/max estimate made at each score'})