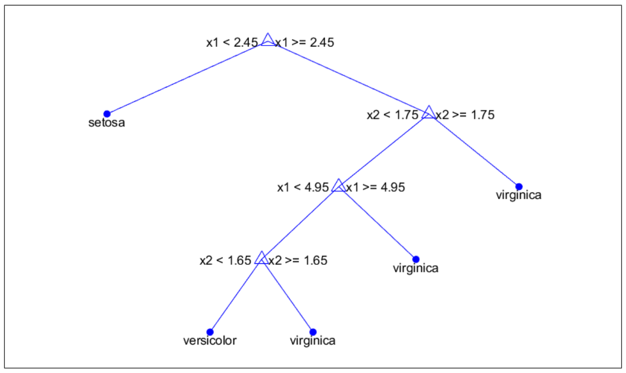
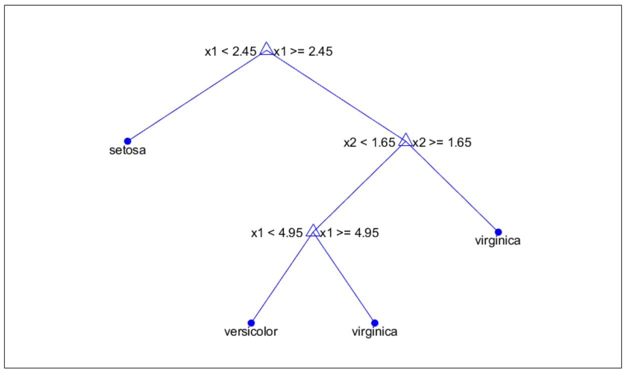
EE 583 Pattern Recognition HW5

Eda Özkaynar 2375582

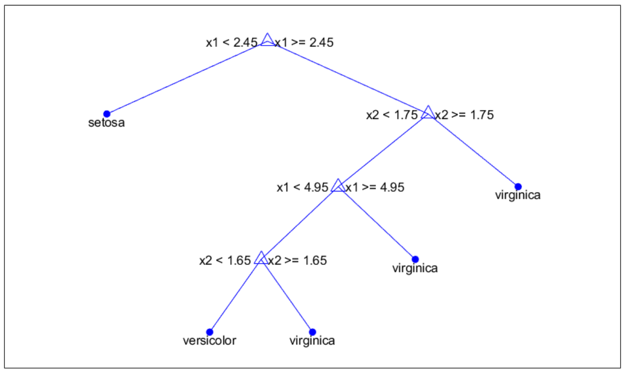
Question 1 Grow a Classification Tree



(a) Classification Tree with Default Parameters (Gini)

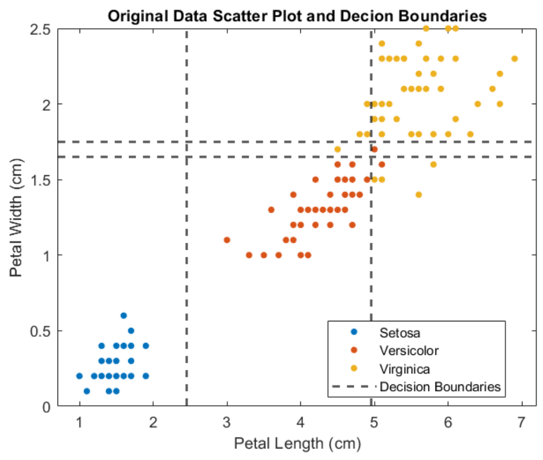


(b) Classification Tree Maximum Number of Splits 7

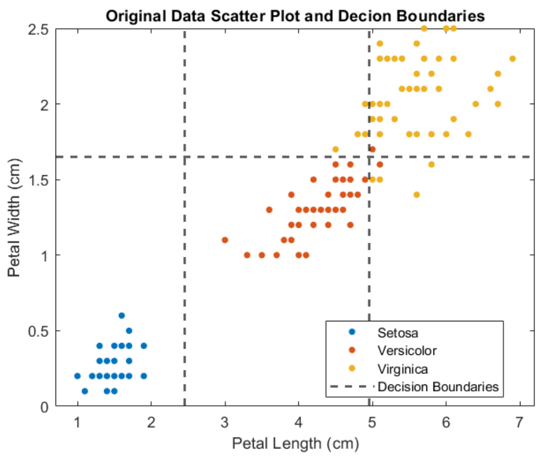


(c) Classification Tree with Entropy SplitCriterion

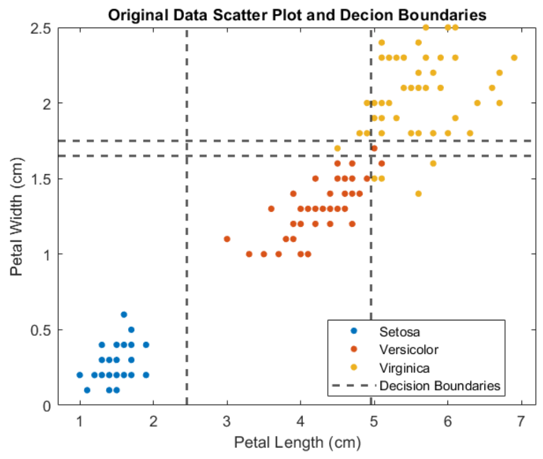
Fig. 1: Classification Trees



(a) Classification Tree with Default Parameters (Gini)



(b) Classification Tree Maximum Number of Splits 7



(c) Classification Tree with Entropy SplitCriterion

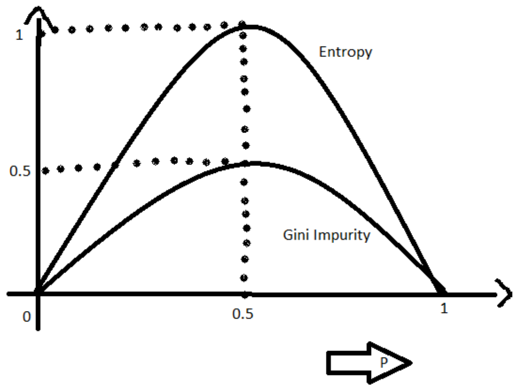
Fig. 2: Decision Boundaries

Model	**Classification Error**
Default Model	0.0400
Max Split Model	0.0533
Split Criterion Model	0.0333

TABLE I: Classification errors for different models.

The model tries all possibilities to determine which feature (Petal Length or Petal Width) and which threshold provides the best split. For example, x_1 = 2.45 is the optimal threshold value because splitting the tree at this point minimizes the impurity. Similarly, other thresholds that further reduce the impurity are identified at subsequent splits, ensuring that each division improves the homogeneity of the resulting subsets.

Reducing the maximum number of splits decreases model complexity, mitigating the overfitting risk. As shown in Figures 6a and 6c, the final partition in the classification tree could lead to overfitting. However, an insufficient number of splits can result in underfitting, preventing the model from capturing the underlying patterns in the data effectively. As shown in Table I, the classification error increases when the maximum number of splits is reduced. However, it is expected that, with a new dataset, the error could be lower because the model's complexity is reduced, thereby decreasing the risk of overfitting.



$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

Fig. 3: Entropy and Gini Impurities Comparison

The reason why the Entropy SplitCriterion reduces the error rate is that it measures the confusion more precisely and optimizes the class separation better during the splits. Entropy has a wider measurement range than Gini, which allows the model to perform better on imbalanced data sets or small class differences. This is clearly seen in the graphs and formulas in Figure 4. Therefore, classification error is reduced as can be seen in Table I.

Question 2 Train Boosted Classification Ensemble

AdaBoost (Adaptive Boosting) is an ensemble learning technique designed for both classification and regression tasks. It is more robust to overfitting compared to many other machine learning algorithms.

AdaBoost incrementally builds a single strong classifier through multiple iterations. By combining weak learners, models that are only slightly better than random guessing, AdaBoost constructs a strong classifier. During each iteration, a new weak learner is added to the ensemble, and a weighting vector is updated to give more focus to previously misclassified examples. This iterative process results in a final classifier that achieves higher accuracy than any of the individual weak learners.

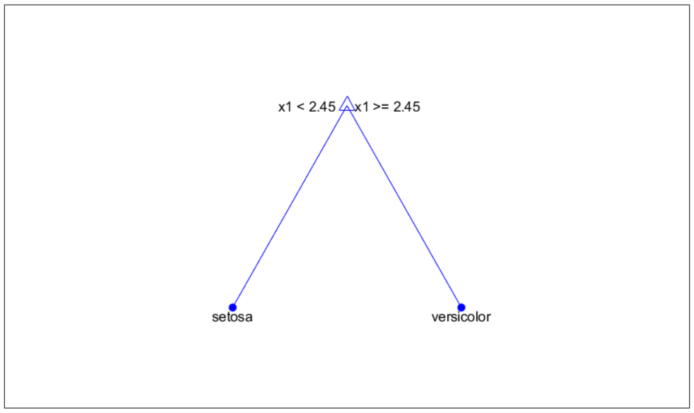


Fig. 4: First Weak Learner

The accuracy of the AdaBoost ensemble is **0.98667**, while the accuracy of the first tree (weak learner) is **0.66667**.

The performance of AdaBoost is based on its ability to compensate for the errors of weak learners and create a stronger ensemble model. Compared to the low accuracy rate of the initial tree, this method achieves high accuracy rates even for difficult classification problems. It increases the generalization power of the model and minimizes errors. Therefore, ensemble models like AdaBoost are a powerful method that significantly improves the performance of weak learners.

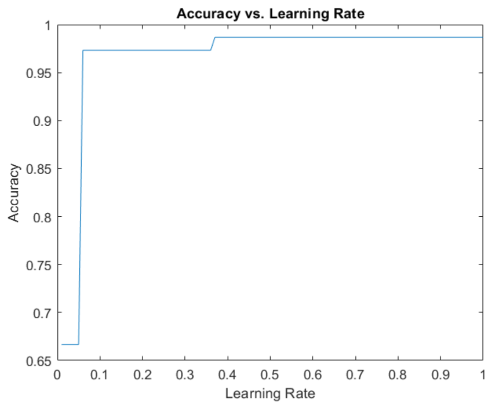


Fig. 5: Learning Rate vs. Accuracy

$$\alpha_t = \eta \cdot \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right)$$

where:

- α_t: Weight of the weak learner at iteration t
- η: Learning rate (a scalar between 0 and 1)
- e_t: Error rate of the weak learner at iteration t

With a small learning rate, the model learns slowly, making it difficult to construct strong learners. As a result, the accuracy remains low when the learning rate is small. A high learning rate can create a robust model, but an excessively high learning rate can cause the model to become unstable. While this issue is not visible in the graph, this can increase the risk of overfitting on some datasets. As can be seen in Figure 7, the optimal learning rate is in the middle range. In this range the model can correct errors at a reasonable rate in each iteration, and weak learners form a strong ensemble model.

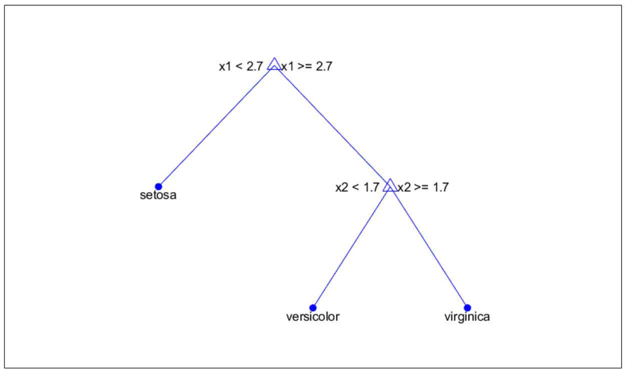
Question 3 TreeBagger

Random Forest is a machine learning algorithm that consists of multiple decision trees and uses ensemble learning technique. The concept of random comes from the various random elements used in both data samples and feature selection. These randomness make the model more generalized and robust to errors.

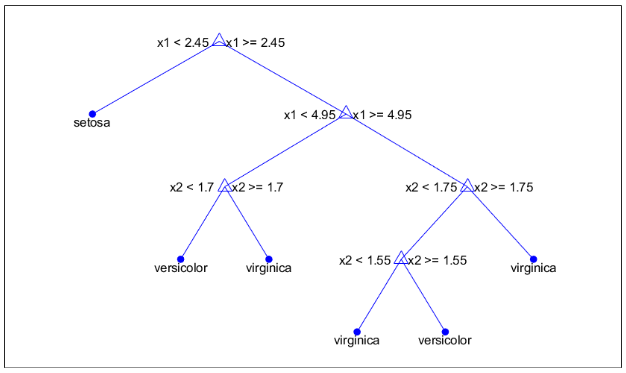
Random Forest Creation Steps

Random Forest is an ensemble learning technique that utilizes randomness in both data sampling and feature selection to build a robust model composed of multiple decision trees. Below are the key steps for building a Random Forest:

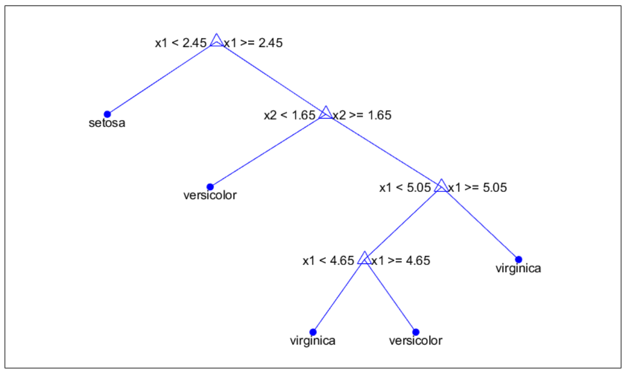
1) **Bootstrap Sampling (Random Sampling):**
   - Randomly sample the training data with replacement to create multiple bootstrap datasets.
   - Each bootstrap dataset is used to train an individual decision tree.
   - Data points not selected in a bootstrap sample are considered *Out-of-Bag (OOB)* data and can be used for validation.
2) **Random Feature Selection:**
   - At each node of a decision tree, select a random subset of features from the total feature set.
   - Use this subset to determine the best split at that node.
3) **Decision Tree Training:**
   - Train each decision tree on its corresponding bootstrap sample and the randomly selected feature subset.
   - Allow the tree to grow fully.
4) **Building the Forest:**
   - Repeat Steps 2–4 to create n decision trees, where n is the number of trees specified by the user.
   - Each tree is built independently.
5) **Prediction Aggregation:**
   - For classification problems:
     - Use **Majority Voting**: The class predicted by the majority of the trees is selected.
   - For regression problems:
     - Use **Averaging**: Take the mean of the predictions from all trees.



(a) First Decision Tree Using Random Seed = 1



(b) First Decision Tree Using Random Seed = 11



(c) First Decision Tree Using Random Seed = 100

Fig. 6: First Decision Trees of Different Random Forests

As shown in Figure 6 and Table II, different random seeds result in different Random Forests. This randomness increases model diversity, thereby reducing the risk of overfitting. Another advantage of randomness is that it enhances the model's generalization ability and robustness to noisy data.

Random Seed (rnd)	**Accuracy of AdaBoost Ensemble**	**Accuracy of the First Tree**
1	0.98667	0.92000
11	0.98667	0.96000
100	0.98667	0.90667

TABLE II: Comparison of accuracies for different random seeds.

Question 4 Jackknife Resampling

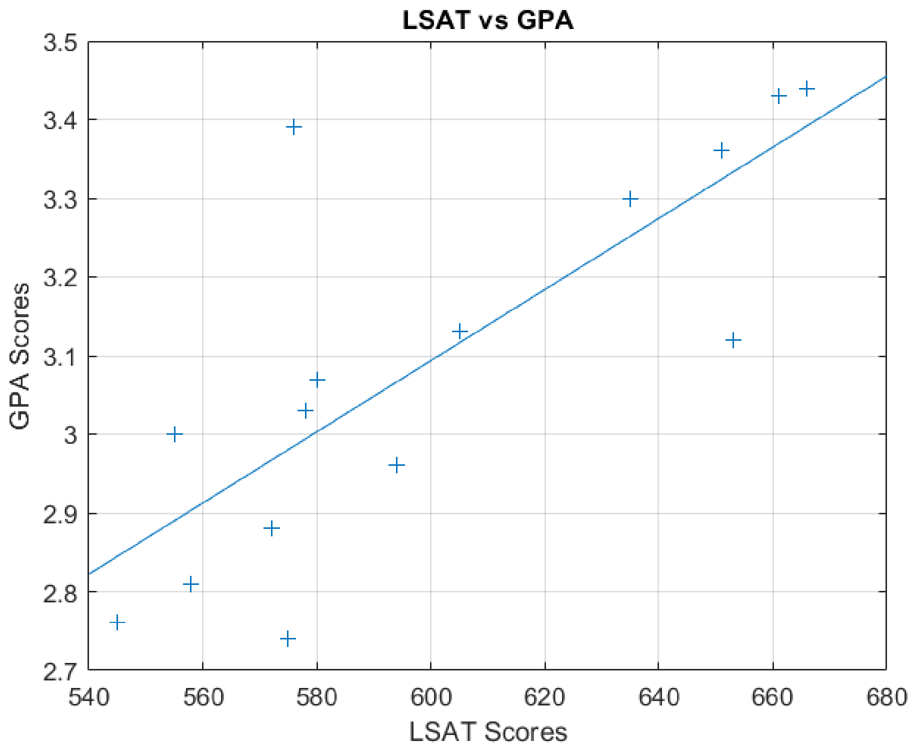


Fig. 7: Learning Rate vs. Accuracy

Jackknife sampling is a resampling method used in statistical analysis. This method involves removing each observation in turn from a data set and making statistical estimates on the remaining data. This allows the bias and variance of the estimates to be assessed.

TABLE III: Jackknife Estimates and Bias for Correlation and Medians

Statistic	**Value**
Sample Mean	0.77637
Mean Jackknife Correlation Coefficient	0.77591
Jackknife Estimate of Bias	-0.0064736
Sample Median LSAT	580
Jackknife Average Median LSAT	583.2
Jackknife Bias LSAT	44.8
Sample Median GPA	3.07
Jackknife Average Median GPA	3.0727
Jackknife Bias GPA	0.037333

For LSAT: The sample median is 580, while the Jackknife mean median is 583.2. The bias estimate is 3.2. This indicates that the sample median has a slight bias. For GPA: The sample median is 3.07, the Jackknife mean median is 3.0727, and the bias estimate is 0.0027. This indicates that the GPA median estimate is highly reliable and contains minimal bias.

```
clear; clc; close all;
%% Question 1 %%

load fisheriris
X = meas(:,3:4);
Y = species;

rng(1); % For reproducibility



% Train a classification tree with default parameters
MdlDefault = fitctree(X,Y,'CrossVal','on');

% Plot the tree
view(MdlDefault.Trained{1},'Mode','graph')

% Cross validation classification error
classErrorDefault = kfoldLoss(MdlDefault);

% Classification Region Plot
cutPredictors   = MdlDefault.Trained{1}.CutPredictor;     % Cut Predictors
cutPoints       = MdlDefault.Trained{1}.CutPoint;         % Cut points
% Scatter plot of the original data
figure;
gscatter(X(:,1), X(:,2), species);
title('Original Data Scatter Plot and Decion Boundaries');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
hold on; % Mevcut scatter plot üzerine çizim yap

for i = 1:length(cutPoints)
    if ~isnan(cutPoints(i)) % Eğer bir bölme noktası varsa
        if strcmp(cutPredictors{i}, 'x1') % Petal Length (x1)
            xline(cutPoints(i), '--k', 'LineWidth', 1.5); % Dikey çizgi
        elseif strcmp(cutPredictors{i}, 'x2') % Petal Width (x2)
            yline(cutPoints(i), '--k', 'LineWidth', 1.5); % Yatay çizgi
        end
    end
end

legend('Setosa', 'Versicolor', 'Virginica', 'Decision Boundaries');
hold off;

% Train a classification tree with maximum number of splits at 7
Mdl7 = fitctree(X,Y,'MaxNumSplits',7);

% Plot the tree
view(Mdl7.Trained{1},'Mode','graph')

% Cross validation classification error
classErrorMdl7 = kfoldLoss(Mdl7);
```

```matlab
% Classification Region Plot
cutPredictors    = Mdl7.Trained{1}.CutPredictor;      % Cut Predictors
cutPoints        = Mdl7.Trained{1}.CutPoint;          % Cut points
% Scatter plot of the original data
figure;
gscatter(X(:,1), X(:,2), species);
title('Original Data Scatter Plot and Decion Boundaries');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
hold on; % Mevcut scatter plot üzerine çizim yap

for i = 1:length(cutPoints)
    if ~isnan(cutPoints(i)) % Eğer bir bölme noktası varsa
        if strcmp(cutPredictors{i}, 'x1') % Petal Length (x1)
            xline(cutPoints(i), '--k', 'LineWidth', 1.5); % Dikey çizgi
        elseif strcmp(cutPredictors{i}, 'x2') % Petal Width (x2)
            yline(cutPoints(i), '--k', 'LineWidth', 1.5); % Yatay çizgi
        end
    end
end

legend('Setosa', 'Versicolor', 'Virginica', 'Decision Boundaries');
hold off;


% Train a classification tree with Entropy SplitCriterion
MdlSplit = fitctree(X,Y,'SplitCriterion','deviance','CrossVal','on');

% Plot the tree
view(MdlSplit.Trained{1},'Mode','graph')

% Cross validation classification error
classErrorMdlSplit = kfoldLoss(MdlSplit);

% Classification Region Plot
cutPredictors    = MdlSplit.Trained{1}.CutPredictor;      % Cut Predictors
cutPoints        = MdlSplit.Trained{1}.CutPoint;          % Cut points
% Scatter plot of the original data
figure;
gscatter(X(:,1), X(:,2), species);
title('Original Data Scatter Plot and Decion Boundaries');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
hold on; % Mevcut scatter plot üzerine çizim yap

for i = 1:length(cutPoints)
    if ~isnan(cutPoints(i)) % Eğer bir bölme noktası varsa
        if strcmp(cutPredictors{i}, 'x1') % Petal Length (x1)
            xline(cutPoints(i), '--k', 'LineWidth', 1.5); % Dikey çizgi
        elseif strcmp(cutPredictors{i}, 'x2') % Petal Width (x2)
            yline(cutPoints(i), '--k', 'LineWidth', 1.5); % Yatay çizgi
        end
    end
```

```
end

legend('Setosa', 'Versicolor', 'Virginica', 'Decision Boundaries');
hold off;
clc; clear; close all;
%% Question 2 %%
rng(100);
load fisheriris
X = meas(:,3:4);
Y = species;

% Partition the data into two halves
cv = cvpartition(Y, 'Holdout', 0.5); % Split data into training and testing sets

idxX1 = training(cv);                 % Indices for X1
idxX2 = test(cv);                     % Indices for X2
idxY1 = training(cv);                 % Indices for Y1
idxY2 = test(cv);                     % Indices for Y2

X1 = X(idxX1, :);
Y1 = Y(idxY1);
X2 = X(idxX2, :);
Y2 = Y(idxY2);

weak_learner = templateTree("MaxNumSplits",1);
% First half of the observation
Mdl = fitcensemble(X1,Y1,'Method','AdaBoostM2','NumLearningCycles',25,'Learners',↙
weak_learner);
% Plot the tree
view(Mdl.Trained{1},'Mode','graph')

% Predict the labels from second half of the data using ensemble model
PredictYensemble = predict(Mdl,X2);

% Predict the labels from second half of the data using first tree
PredictYfirsttree = predict(Mdl.Trained{1},X2);

% Calculate classification accuracy for both
accuracyEnsemble    = 0;
accuracyFirstTree   = 0;
for i = 1:numel(Y2)
    accuracyEnsemble  = accuracyEnsemble  + strcmp(PredictYensemble{i},Y2{i});
    accuracyFirstTree = accuracyFirstTree + strcmp(PredictYfirsttree{i}, Y2{i});
end
accuracyEnsemble = accuracyEnsemble / numel(Y2);
accuracyFirstTree = accuracyFirstTree / numel(Y2);
% Display classification accuracies
disp(['Accuracy of AdaBoost ensemble: ', num2str(accuracyEnsemble)]);
disp(['Accuracy of the first tree: ', num2str(accuracyFirstTree)]);

% LearnRate Parameters

lr = 0.01:0.01:1;
```

```
acc = zeros(size(lr));
for i = 1:length(lr)
    weak_learner = templateTree("MaxNumSplits",1);
    Mdl = fitcensemble(X1,Y1,'Method','AdaBoostM2','NumLearningCycles',↙
25,'Learners',weak_learner,"LearnRate",lr(i));
    Yensemble = predict(Mdl,X2);
    Yfirsttree = predict(Mdl.Trained{1},X2);

    % Calculate classification accuracy
    accEnsemble    = 0;
    for ii = 1:numel(Y2)
        accEnsemble  = accEnsemble  + strcmp(Yensemble{ii},Y2{ii});
    end
    accEnsemble = accEnsemble / numel(Y2);
    acc(i) = accEnsemble;
end

figure;
plot(lr,acc)
xlabel("Learning Rate")
ylabel("Accuracy")
title("Accuracy vs. Learning Rate")
clc; clear; close all;
%% Question 3 %%
rng(100);
load fisheriris
X = meas(:,3:4);
Y = species;

% Partition the data into two halves
cv = cvpartition(Y, 'Holdout', 0.5); % Split data into training and testing sets

idxX1 = training(cv);                 % Indices for X1
idxX2 = test(cv);                     % Indices for X2
idxY1 = training(cv);                 % Indices for Y1
idxY2 = test(cv);                     % Indices for Y2

X1 = X(idxX1, :);
Y1 = Y(idxY1);
X2 = X(idxX2, :);
Y2 = Y(idxY2);
% rng(100);

Mdl = TreeBagger(25,X1,Y1,'SampleWithReplacement','on','OOBPrediction',"on","↙
Method","classification");

view(Mdl.Trees{1},'Mode','graph')
%Prediction
% Predict the labels from second half of the data using ensemble model
PredictYensemble = predict(Mdl,X2);

% Predict the labels from second half of the data using first tree
PredictYfirsttree = predict(Mdl.Trees{1},X2);
```

```
% Calculate classification accuracy for both
accuracyEnsemble    = 0;
accuracyFirstTree   = 0;
for i = 1:numel(Y2)
    accuracyEnsemble  = accuracyEnsemble  + strcmp(PredictYensemble{i},Y2{i});
    accuracyFirstTree = accuracyFirstTree + strcmp(PredictYfirsttree{i}, Y2{i});
end
accuracyEnsemble = accuracyEnsemble / numel(Y2);
accuracyFirstTree = accuracyFirstTree / numel(Y2);
% Display classification accuracies
disp(['Accuracy of AdaBoost ensemble: ', num2str(accuracyEnsemble)]);
disp(['Accuracy of the first tree: ', num2str(accuracyFirstTree)]);

plot(oobError(Mdl))
xlabel("Number of Grown Trees")
ylabel("Out-of-Bag Classification Error")
clc; clear; close all;
%% Question 4 Jackknife sampling %%

% First compute the sample correlation on the data
load lawdata
rhohat = corr(lsat,gpa);

plot(lsat,gpa,'+')
lsline
xlabel('LSAT Scores')   % Label x-axis
ylabel('GPA Scores')    % Label y-axis
title('LSAT vs GPA')    % Plot title
grid on                 % Add grid for better visualization

% Next compute the correlations for jackknife samples, and compute their mean
rng default;  % For reproducibility
jackrho = jackknife(@corr,lsat,gpa);
meanrho = mean(jackrho);

% Compute an estimate of the bias
n = length(lsat);
biasrho = (n-1) * (meanrho-rhohat);

% histogram(jackrho,30,'FaceColor',[.8 .8 1])

% Median Calculation for LSAT
medhat_lsat = median(lsat);
jackmed_lsat = jackknife(@median,lsat);

biasmed_lsat = (n-1) * (mean(jackmed_lsat)- (medhat_lsat));

% Median Calculation for GPA
medhat_gpa = median(gpa);
jackmed_gpa = jackknife(@median,gpa);
```

```
biasmed_gpa = (n-1) * (mean(jackmed_gpa)- mean(medhat_gpa));

disp(['Sample Mean : ', num2str((rhohat))]);
disp(['Mean Jackknife Correlation Coefficient: ', num2str(meanrho)]);
disp(['Jackknife Estimate of Bias: ', num2str(biasrho)]);


disp(['Sample Median LSAT: ', num2str((medhat_lsat))]);
disp(['Jackknife Average Median LSAT: ', num2str(mean(jackmed_lsat))]);
disp(['Jackknife Bias LSAT: ', num2str(biasmed_lsat)]);

disp(['Sample Median GPA: ', num2str(medhat_gpa)]);
disp(['Jackknife Average Median GPA: ', num2str(mean(jackmed_gpa))]);
disp(['Jackknife Bias GPA: ', num2str(biasmed_gpa)]);
```