HW2: Information Based Tree

Ques 3. The table below lists a sample of data from a census.

ID	Age	Education	Marital Status	Occupation	Annual Income
1	39	bachelors	never married	transport	25K–50K
2	50	bachelors	married	professional	25K–50K
3	18	high school	never married	agriculture	≤ 25K
4	28	bachelors	married	professional	25K–50K
5	37	high school	married	agriculture	25K–50K
6	24	high school	never married	armed forces	≤ 25K
7	52	high school	divorced	transport	25K–50K
8	40	doctorate	married	professional	≥ 50K

There are four descriptive features and one target feature in this dataset:

- AGE, a continuous feature listing the age of the individual
- EDUCATION, a categorical feature listing the highest education award achieved by the individual (high school, bachelors, doctorate)
- MARITAL STATUS (never married, married, divorced)
- OCCUPATION (transport = works in the transportation industry; professional = doctors, lawyers, etc.; agriculture = works in the agricultural industry; armed forces = is a member of the armed forces)
- ANNUAL INCOME, the target feature with 3 levels (<25K, 25K–50K, >50K)

1. **Calculate the entropy for this dataset.**

   Ans: The formula for entropy is shown below:

$$H(t, \mathcal{D}) = -\sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l)))$$

   Hence, for the target feature Annual income which consist of three categories ie. values >= 50k, values <= 25k and values between 25k and 50k.
   So, out of total 8 values.
   H(Annual Income) = $-((5/8)\log_2(4/8) + (1/8)\log_2(1/8) + (2/8)\log_2(2/8))$ = **1.2988 bits**

2. **Calculate the Gini index for this dataset.**
   Ans: The formula to calculate Gini index is shown below:

$$Gini(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2$$

Gini Index = 1- ((5/8)^2 + (2/8)^2 + (1/8)^2) = 0.53125 = **0.5313**

3. When building a decision tree, the easiest way to handle a continuous feature is to define a threshold around which splits will be made. What would be the optimal threshold to split the continuous AGE feature (use information gain based on entropy as the feature selection measure)?

Ans: To find the optimal threshold point to split the continuous values, first continuous values are sorted and then we will find adjacent values having different target levels. Then the boundary value of each of these values is calculated using average of elevation values.

Id	Sorted Age Values	Annual Income
3	18	<= 25k
6	24	<= 25k
4	28	25k – 50k
5	37	25k – 50k
1	39	25k – 50k
8	40	>= 50k
2	50	25k – 50k
7	52	25k – 50k

Here we can see that there are three pairs of adjacent values 24 and 28, 39 and 40, 40 and 52.

The boundary of these values are (24+28)/2, (39+40)/2 and (40+50)/2. These points are 26,39.5, 45.

The value of threshold points is given below:

For Split Feature > 26:

Partition Entropy for Instances (which means False) : d3, d6 = 0

Partition Entropy for Instances (which means True): d1,d2,d4,d5,d7,d8 = -((5/6) * log2(5/6) + (1/6) * log2(1/6)) = 0.6500

Rem. = (6/8)*(0.6500) = 0.4875

Information Gain = Total Entropy– Rem= 1.2988- 0.4875= **0.8113 bits**

For Split Feature > 39.5:

Partition Entropy for Instances (which means False): d1,d3,d4,d5,d6 = -((2/5) * log2(2/5) + (3/5) * log2(3/5)) = 0.9707

Partition Entropy for Instances (which means True): d2,d7,d8 = -((1/3) * log2(1/3) + (2/3) * log2(2/3))= 0.917

Rem. = (5/8)*(0.9707) + (3/8)*(0.917) = 0.949

Information Gain = Total Entropy– Rem= 1.2988- 0.949= **0.3498 bits**

For Split Feature > 45:

Partition Entropy for Instances (which means False): d1,d3,d4,d5,d6,d8 = -((3/6) * log2(3/6) + (2/6) * log2(2/6) + (1/6) * log2(1/6)) = 1.4591

Partition Entropy for Instances (which means True): d2,d7 = 0

Rem. = (6/8)*(1.4591) = 1.0944

Information Gain = Total Entropy– Rem= 1.2988- 1.0944= **0.2044 bits**

From the Information gain calculated above, we can see that the higher information gain is for **split feature > 26**. Hence, this is the most optimal threshold point.

Table showing Information gain score for threshold points

Split by Feature	Level	Instances	Partition Entropy	Rem.	Information Gain
>26	true	d1, d2, d4, d5, d7, d8	0.65	0.4875	0.8113
	false	d3, d6	0		
>39.5	true	d2, d7, d8	0.9183	0.949	0.3498
	false	d1, d3, d4, d5, d6,	0.971		
>45	true	d2, d7	0	1.0944	0.2044
	false	d1, d3, d4,d5,d6,d8	1.4591		

4. **Calculate information gain (based on entropy) for the EDUCATION, MARITAL STATUS, and OCCUPATION features.**

   Ans: The entropy for the entire dataset was calculated above which is 1.2988 bits

   Information gain scores for Education

   Partition Entropy for high school= -((2/4 * log2(2/4)) + (2/4 * log2(2/4))) = 1.00 bits

   Partition Entropy for bachelors = - ((3/3 * log2(3/3) = 0 bits

   Partition Entropy for doctorate = - ((1/1 * log2(1/1) = 0 bits

   Rem. = (4/8 * 1) + (3/8 * 0) + (1/8 *0) = 0.5 bits

   Information gain score= 1.2988- 0.5 = **0.7988 bits**

   Information gain scores for Marital Status

   Partition entropy for never married: -((2/3 $\log_2$ 2/3) + (1/3 $\log_2$ 1/3) = 0.9183

   Partition entropy for married : -((3/4 $\log_2$ 3/4) + (1/4 $\log_2$ 1/4) = 0.8113

   Partition entropy for divorce: - ((1/1 $\log_2$ 1/1)) = 0

   Rem = (3/8*0.9183) + (4/8 * 0.8113) = 0.75 bits

   Information Gain = 1.2988- 0.75 = **0.5488 bits**

Information gain scores for Occupation

Partition entropy for transport: - ((2/2 $\log_2$ 2/2)) = 0

Partition entropy for professional : - ((2/3 $\log_2$ 2/3) + (1/3 $\log_2$ 1/3)) = 0.9183

Partition entropy for agriculture: - ((1/2 $\log_2$ 1/2) + (1/2 $\log_2$ 1/2)) = 1

Partition entropy for armed forces: - ((1/1 $\log_2$ 1/1)) = 0

Rem = (3/8*0.9183) + (2/8 * 1) = 0.5944

Information Gain = 1.2988- 0.5944 = **0.7044 bits**

Table showing Information gain score for descriptive features

Feature	**Level**	**Instances**	**Partition Entropy**	**Rem.**	**Information Gain**
Education	high school	d3, d5, d6, d7	01	0.5	0.7988
	bachelors	d1, d2, d4	0		
	doctorate	d8	0		
Marital Status	Never married	d1, d3, d6	0.9183	0.75	0.5488
	married	d2, d4, d5, d8	0.8113		
	divorce	d7	0		
Occupation	transport	d1, d7	0	0.5944	0.7044
	professional	d2, d4, d8	0.9183		
	agriculture	d3, d5	1		
	Armed forces	d6	0		

5.Calculate the information gain ratio (based on entropy) for EDUCATION, MARITAL STATUS, and OCCUPATION features.

Ans: Formula to calculate information gain is given below:

information gain scores by the respective entropy/ Entropy of each feature

Information Gain Ratio for Education

Information Gain for Education was calculated above which is : **0.7988 bits**

Entropy of Education is : -((4/8 $\log_2$*4/8) + (3/8 $\log_2$*3/8)+ (1/8 $\log_2$*1/8))= 1.4056 bits

Information Gain Ratio= 0.7988/1.4056 = **0.5683 bits**

Information Gain Ratio for Marital Status

Information Gain for Marital Status was calculated above which is **: 0.5488 bits**

Entropy of Marital Status is : -((3/8 $\log_2$*3/8) + (4/8 $\log_2$*4/8)+ (1/8 $\log_2$*1/8))= **1.4056 bits**

Information Gain Ratio= 0.5488/1.4056 **= 0.3904 bits**

Information Gain Ratio for Occupation

Information Gain for Occupation was calculated above which is : **0.7044 bits**

Entropy of Occupation is : -((2/8 $\log_2$*2/8) + (3/8 $\log_2$*3/8)+ (2/8 $\log_2$*2/8) + (1/8 $\log_2$*1/8))= 1.9056 bits

Information Gain Ratio= 0.7044/1.9056 = **0.3696 bits**

6. Calculate information gain using the Gini index for the EDUCATION, MARITAL STATUS, and OCCUPATION features.

Ans: Gini Index for the entire dataset:

Gini(Annual Income) = 1- ((2/8)**2 + (5/8)**2 + (1/8)**2) = 0.53125 = 0.5313

Information gain using the Gini index is calculated below:

Information Gain for Education:

Partition Gini index for high school: 1- ((2/4) **2 + (2/4) **2) = 0.5
Partition Gini index for bachelors : 1- ((3/3) **2) = 0
Partition Gini index for doctorate: 1- ((1/1) **2) =0
Rem = (4/8 * 0.5) + (3/8 * 0) + (1/8 * 0) = 0.25
Information Gain = (Gini Index of entire dataset)- (Gini index of education)= 0.5313- 0.25= **0.2813**

Information Gain for Marital Status:

Partition Gini index for never married: 1- ((2/3) **2 + (1/3) **2) = 0.4444
Partition Gini index for married : 1- ((3/4) **2 + (1/4) **2) = 0.375
Partition Gini index for divorce: 1- ((1/1) **2) = 0
Rem = (3/8*0.4444) + (4/8 * 0.375) = 0.3542
Information Gain = 0.5313- 0.3542= **0.1771**

Information Gain for Occupation:

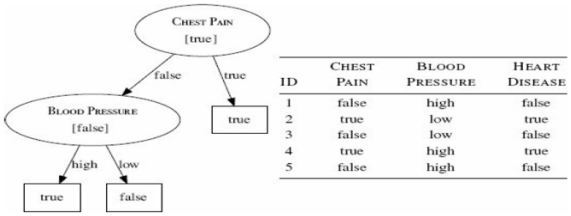
Partition Gini index for transport: 1- ((2/2) **2) = 0
Partition Gini index for professional : 1- ((2/3) **2 + (1/3) **2) = 0.4444
Partition Gini index for agriculture: 1- ((1/2) **2 + (1/2) **2) = 0.5
Partition Gini index for armed forces: 1- ((1/1) **2) = 0
Rem = (3/8*0.4444) + (2/8 * 0.5) = 0.2917

Information Gain = 0.5313- 0.2917= **0.2396**

Table showing Information gain for all features

Feature	Level	Instances	Partition Gini Index	Rem.	Information Gain
Education	high school	d3, d5, d6, d7	0.5	0.25	0.2813
	bachelors	d1, d2, d4	0		
	doctorate	d8	0		
Marital Status	Never married	d1, d3, d6	0.4444	0.3542	0.1771
	married	d2, d4, d5, d8	0.375		
	divorce	d7	0		
Occupation	transport	d1, d7	0	0.2917	0.2396
	professional	d2, d4, d8	0.4444		
	agriculture	d3, d5	0.5		
	Armed forces	d6	0		

Ques 4. The diagram below shows a decision tree for the task of predicting heart disease. The descriptive features in this domain describe whether the patient suffers from chest pain (CHEST PAIN) as well as the blood pressure of the patient (BLOOD PRESSURE). The binary target feature is HEART DISEASE. The table beside the diagram lists a pruning set from this domain. pain (CHEST PAIN) as well as the blood pressure of the patient (BLOOD PRESSURE). The binary target feature is HEART DISEASE. The table beside the diagram lists a pruning set from this domain.



ID	CHEST PAIN	BLOOD PRESSURE	HEART DISEASE
1	false	high	false
2	true	low	true
3	false	low	false
4	true	high	true
5	false	high	false

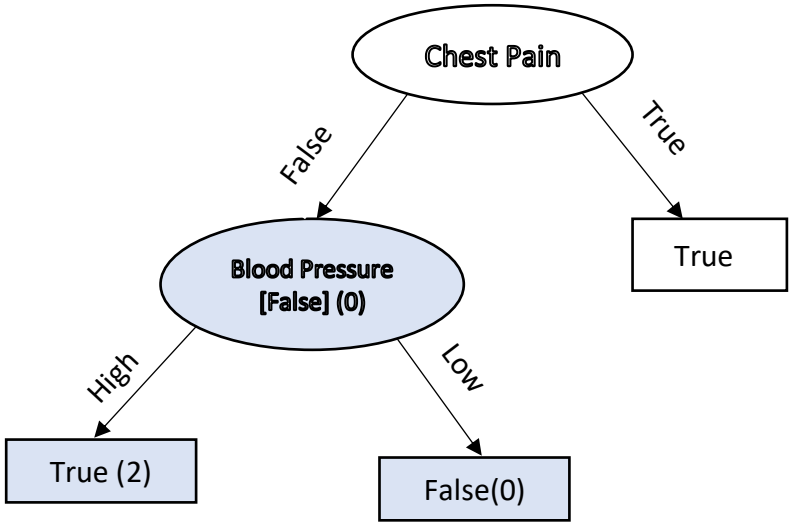
Using the pruning set, apply reduced error pruning to the decision tree. Assume that the algorithm is applied in a bottom-up, left-to-right fashion. For each iteration of the algorithm, indicate the subtrees considered as pruning candidates, explain why the algorithm chooses to prune or leave these subtrees in the tree, and illustrate the tree that results from each iteration

Ans: Reduced error pruning is one of the most popular version of post-pruning based on error rates. In this method, a decision tree is build and then the tree in search in an iterative bottom up left to right fashion so that sub trees can be pruned.

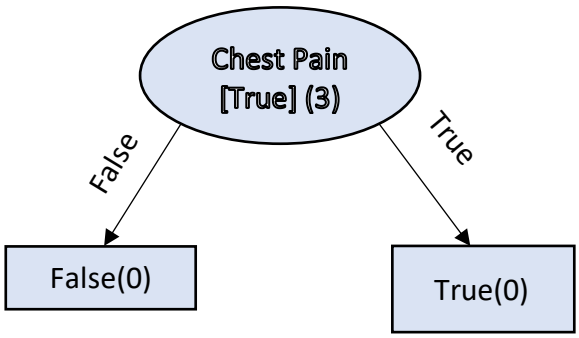
The error rate of the root node of the subtree is compared with the error rate of leaves node. If the error rate at the subtree root node is less than or equal to the combined error rate at the leaves, the subtree is pruned.

The pruning algorithm considers the subtree under the blood pressure node for pruning.

Figure shows the pruning. "The nodes in the blue shows the subtree and the square bracket is the majority target level for that node, and the round brackets shows the error rate. "



1. With reference to the figure above, the root node of this subtree returns false i.e for instance (d1,d3,d5). So the error rate of this root node is 0.
2. However, the prediction made on the leaf node is incorrect because d1, d5 instances are predicted as false as per the data but decision tree predicts it as true. so the error rate for the leaves node of this sub tree is 0+2.
3. As the error rate of the sub tree is higher than the root node of the subtree then this subtree is pruned and replace by the leaf node.
4. The figure below depicts the structure of the tree after the tree under the blood pressure node is pruned.



5. From the figure below we can see that the root node of this subtree return true as a prediction which doesn't match the validation set (Where 3 instances from the data are predicted as false and 2 instances as true). Hence its error rate is 3.

6. On the other hand, the error rate made at the leaf node is 0. As the predictions made are true for instance (d2, d4) and false for instance (d1, d3, d5) respectively which matches with the prediction data given.

7. As the error rate at the leaf node is less than the error rate at the root node, this tree will not be further pruned. Therefore, the pruning algorithm will stop.

Ques 5.The following table lists a dataset containing the details of five participants in a heart disease study, and a target feature RISK which describes their risk of heart disease.
Each patient is described in terms of four binary descriptive features
EXERCISE, how regularly do they exercise
SMOKER, do they smoke
OBESE, are they overweight
FAMILY, did any of their parents or siblings suffer from heart disease

ID	EXERCISE	SMOKER	OBESE	FAMILY	RISK
1	daily	false	false	yes	low
2	weekly	true	false	yes	high
3	daily	false	false	no	low
4	rarely	true	true	yes	high
5	rarely	true	true	no	high

a. As part of the study researchers have decided to create a predictive model to screen participants based on their risk of heart disease. You have been asked to implement this screening model using a random forest. The three tables below list three bootstrap samples that have been generated from the above dataset. Using these bootstrap samples create the decision trees that will be in the random forest model (use entropy based information gain as the feature selection criterion).

ID	EXERCISE	FAMILY	RISK
1	daily	yes	low
2	weekly	yes	high
2	weekly	yes	high
5	rarely	no	high
5	rarely	no	high

Bootstrap Sample A

ID	SMOKER	OBESE	RISK
1	false	false	low
2	true	false	high
2	true	false	high
4	true	true	high
5	true	true	high

Bootstrap Sample B

ID	OBESE	FAMILY	RISK
1	false	yes	low
1	false	yes	low
2	false	yes	high
4	true	yes	high
5	true	no	high

Bootstrap Sample C

Ans: **The entropy for bootstrap sample A**

H(sample A) = -((1/5 $\log_2$ (1/5) + (4/5 $\log_2$ (4/5)) = 0.7219 bits

Information Gain for feature Exercise:

Partition Entropy for daily = -(1/1 $\log_2$1/1) = 0

Partition Entropy for weekly = -(2/2 $\log_2$2/2) = 0

Partition Entropy for rarely = -(2/2 $\log_2$2/2) = 0

Rem = 0

Information Gain = 0.7219- 0 = **0.7219 bits**

Information Gain for feature Family:

Partition Entropy for yes = -((1/3 $\log_2$1/3) + (2/3 $\log_2$2/3)) = 0.9183

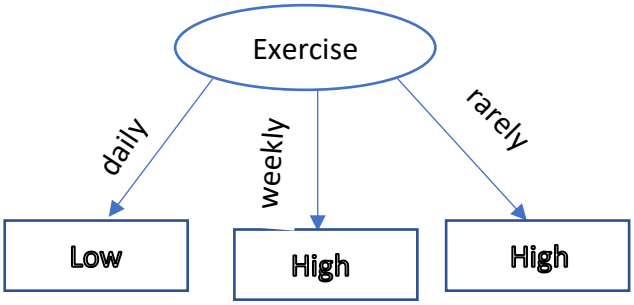
Partition Entropy for no = -(2/2 $\log_2$2/2) = 0

Rem = 3/5*0.9183 = 0.5516

Information Gain = 0.7219- 0.5516 = 0.1703 bits

So, these calculation shows that the information gain for feature exercise is more. Hence, exercise will be added as root node for this decision tree from bootstrap sample A. Also, splitting on the exercise will generate pure sets so the decision tree doesn't need to further split.

Tree for Bootstrap Sample A



The entropy for bootstrap sample B

H(sample B) = -((1/5 $\log_2$ (1/5) + (4/5 $\log_2$ (4/5)) = 0.7219 bits

Information Gain for feature smoker:

Partition Entropy for false = -(1/1 $\log_2$1/1) = 0

Partition Entropy for true= -(4/4 $\log_2$4/4) =0

Rem = 0

Information Gain = 0.7219- 0 = **0.7219 bits**

Information Gain for feature obese:

Partition Entropy for true = -((2/2 $\log_2$2/2) = 0

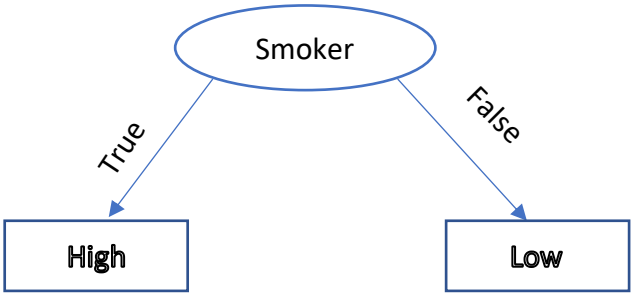
Partition Entropy for false = -((2/3 $\log_2$2/3) + (1/3 $\log_2$ 1/3)) = 0.9183 bits

Rem = 3/5*0.9183 = 0.5516 bits

Information Gain = 0.7219- 0.5516 = 0.1703 bits

So, these calculation shows that the information gain for feature smoker is more. Hence, smoker will be added as root node for this decision tree from bootstrap sample B. Also, splitting on the smoker will generate pure sets so the decision tree doesn't need to further split.

Tree for Bootstrap Sample B



The entropy for bootstrap sample C

H(sample C) = -((2/5 $\log_2$ (2/5) + (3/5 $\log_2$ (3/5)) = 0.9710 bits

Information Gain for feature family:

Partition Entropy for yes = -((2/4 $\log_2$2/4) + (2/4 $\log_2$2/4)) = 1 bit

Partition Entropy for no= -(1/1 $\log_2$1/1) =0

Rem = 4/5*(1)= 0.8 bits

Information Gain = 0.9710- 0.8 = 0.171 bits

Information Gain for feature obese:

Partition Entropy for true = -((2/2 $\log_2$2/2) = 0

Partition Entropy for false = -((2/3 $\log_2$2/3) + (1/3 $\log_2$ 1/3)) = 0.9183 bits

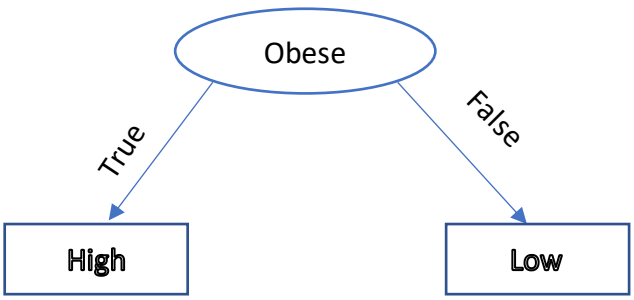
Rem = 3/5*0.9183 = 0.5510 bits

Information Gain = 0.9710- 0.5510 = **0.4200 bits**

So, these calculation shows that the information gain for feature obese is more. Hence, obese will be added as root node for this decision tree from bootstrap sample C. Also, splitting on the obese creates one pure partition for obese= True where the risk is high and one impure partition where two instances have risk= low and one instance has risk= high.

Normally , in this situation we further split on impure partition but in this case there is only feature family where all the instances have the same level for the family= yes. Hence, instead of splitting this further we create a leaf node where the majority target level in the partition is risk = low.

Tree for Bootstrap Sample C



b. Assuming the random forest model you have created uses majority voting, what prediction will it return for the following query:
EXERCISE=rarely, SMOKER=false, OBESE=true, FAMILY=yes
Ans: from the decision trees drawn above. Each of these tree will give the output as follows:

Tree1 : Exercise = rarely will give the risk = high (bootstrap sample A)
Tree2 : smoker = false will give the risk = low (bootstrap sample B)
Tree3 : obese = true will give the risk = high (bootstrap sample C)

Therefore, majority voting is for risk = high. So, the prediction model will return **risk= high** for the query.

6. The following table lists a dataset containing the details of six patients. Each patient is described in terms of three binary descriptive features (OBESE, SMOKER, and DRINKS ALCOHOL) and a target feature (CANCER RISK).

ID	OBESE	SMOKER	DRINKS ALCOHOL	CANCER RISK
1	true	false	true	low
2	true	true	true	high
3	true	false	true	low
4	false	true	true	high
5	false	true	false	low
6	false	true	true	high

a. Which of the descriptive features will the ID3 decision tree induction algorithm choose as the feature for the root node of the decision tree?
Ans: The ID3 decision tree select the root node by calculating the information gain for all descriptive features in the training data. The feature having the highest information gain is chosen as the root node for that decision tree.

So the first step is to calculate the entropy for the entire dataset wrt. to the target feature. Then calculate the information gain for each descriptive features.

calculate the entropy of the entire dataset.

H(Cancer Risk) = -((3/6 $\log_2$ 3/6) + (3/6 $\log_2$ 3/6)) = 1 bits

Information Gain for feature obese:
Partition entropy for true(d1,d2,d3) = -((2/3 $\log_2$ 2/3) + (1/3 $\log_2$ 1/3)) = 0.9183 bits
Partition entropy for false(d4,d5,d6) = -((2/3 $\log_2$ 2/3) + (1/3 $\log_2$ 1/3)) = 0.9183 bits
Rem = (3/6* 0.9183) + (3/6* 0.9183) = 0.9183 bits
Information Gain = 1- 0.9183 = 0.0817 bits

Information Gain for feature smoker:
Partition entropy for true (d2,d4,d5,d6) = -((3/4 $\log_2$ 3/4) + (1/4 $\log_2$ 1/4)) = 0.8113 bits
Partition entropy for false (d1,d3) = -((2/2 $\log_2$ 2/2)) = 0
Rem = (4/6* 0.8113) = 0.5409 bits
Information Gain = 1- 0.5409 = **0.4591 bits**

Information Gain for feature drinks alcohol:
Partition entropy for true (d1,d2,d3,d4,d6) = -((3/5 $\log_2$ 3/5) + (2/5 $\log_2$ 2/5)) = 0.9709 bits
Partition entropy for false (d5) = -((1/1 $\log_2$ 1/1)) = 0
Rem = (5/6* 0.9709) = 0.8091 bits
Information Gain = 1- 0.8091 = 0.1909 bits

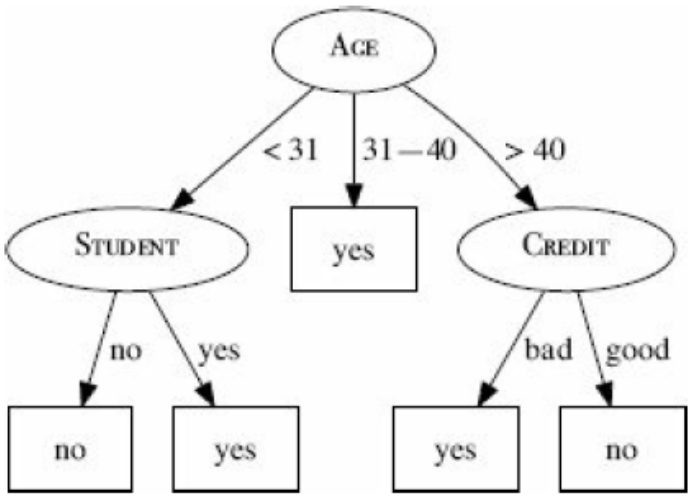
Hence, from the information gain calculated above we can that **smoker** has the highest information gain ie. **0.4591 bits.** So the ID3 decision tree will chose this smoker as the root node of the decision tree.

b. When designing a dataset, it is generally a bad idea if all the descriptive features are indicators of the target feature taking a particular value. For example, a potential criticism of the design of the dataset in this question is that all the descriptive features are indicators of the CANCER RISK target feature taking the same level, high. Can you think of any descriptive features that could be added to this dataset that are indicators of the low target level?
Ans: Nutrition level(such as diet rich in vitamins and iron could be categorised as low and high nutrition level) , and physically active life (can be categorized as high ,low and moderate) contributes to the lower risk of cancer. Hence, this data set could include features such as nutrition level and active life of the patients.

Ques. 7. The following table lists a dataset collected in an electronics shop showing details of customers and whether they responded to a special offer to buy a new laptop.

ID	AGE	INCOME	STUDENT	CREDIT	BUYS
1	< 31	high	no	bad	no
2	< 31	high	no	good	no
3	31 – 40	high	no	bad	yes
4	> 40	med	no	bad	yes
5	> 40	low	yes	bad	yes
6	> 40	low	yes	good	no
7	31 – 40	low	yes	good	yes
8	< 31	med	no	bad	no
9	< 31	low	yes	good	yes
10	> 40	med	yes	bad	yes
11	< 31	med	yes	good	yes
12	31 – 40	med	no	good	yes
13	31 – 40	high	yes	bad	yes
14	> 40	med	no	good	no

This dataset has been used to build a decision tree to predict which customers will respond to future special offers. The decision tree, created using the ID3 algorithm, is shown below.



a. The information gain (calculated using entropy) of the feature AGE at the root node of the tree is 0.247. A colleague has suggested that the STUDENT feature would be better at the root node of the tree. Show that this is not the case.

Ans: Calculate the entropy of the entire dataset:
H(Buys) = -((9/14 $\log_2$ 9/14) +(5/14 $\log_2$ 5 /14)) = **0.9403 bits**

Information gain for student:
Partition entropy for no: -((4/7 $\log_2$ 4/7) + (3/7 $\log_2$ 3/7)) = 0.9852
Partition entropy for yes: -((6/7 $\log_2$ 6/7) + (1/7 $\log_2$ 1/7))= 0.5917
Rem = (7/14*0.9852) + (7/14 * 0.5917) = 0.78845 = 0.7785
Information Gain : 0.9403- 0.7885 = **0.1581 bits**

Hence from the information gain calculate we can conclude that information for student is less than the information gain for age which is 0.247. Therefore, student should not be made root node for this decision tree.

b. Yet another colleague has suggested that the ID feature would be a very effective at the root node of the tree. Would you agree with this suggestion?
Ans: Each instance for the feature ID is unique and due to this it would not be a good feature for root node of the tree. Also, it doesn't contains any information and the resulting decision tree would be overfitted to the training data ie. the variance would be very high. We can overcome this by using other measure such as information gain ratio instead of entropy. To clarify further information gain for ID feature is calculated below:

As the values in ID are continuous we shall find the boundaries where the adjacent value for target are different. The threshold point are 2.5, 5.5, 6.5,7.5,8.5 and 13.5.

Information gain for split feature > 2.5
For true:
Instances are d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14
Partition entropy : 0.8113
For False:
Instances are : d1,d2
Partition entropy : 0
Rem: 12/14*0.8113= 0.6954
Information gain = 0.9403- 0.6954= 0.2449 bits

Information gain for split feature > 5.5
For true:
Instances are d6,d7,d8,d9,d10,d11,d12,d13,d14
Partition entropy : 0.9183
For False:
Instances are : d1,d2, d3,d4,d5
Partition entropy : 0.971
Rem: (9/14*0.9183) + (5/14* 0.971)= 0.9371
Information gain = 0.9403- 0.9371= 0.0032 bits

Information gain for split feature > 6.5

For true:

Instances are: d7,d8,d9,d10,d11,d12,d13,d14

Partition entropy : 0.8113

For False:

Instances are : d1,d2, d3,d4,d5, d6

Partition entropy : 1

Rem: (8/14*0.8113) + (6/14* 1)= 0.8922

Information gain = 0.9403- 0.8922= 0.0481 bits

Information gain for split feature > 7.5

For true:

Instances are: d8,d9,d10,d11,d12,d13,d14

Partition entropy : 0.8631

For False:

Instances are : d1,d2, d3,d4,d5, d6,d7

Partition entropy : 0.9852

Rem: (7/14*0.8631) + (7/14* 0.9852)= 0.9242

Information gain = 0.9403- 0.9242= 0.0161 bits

Information gain for split feature > 8.5

For true:

Instances are: d9,d10,d11,d12,d13,d14

Partition entropy : 0.65

For False:

Instances are : d1,d2, d3,d4,d5, d6,d7,d8

Partition entropy : 0.1

Rem: (6/14*0.65) + (8/14*1)= 0.85

Information gain = 0.9403- 0.85**= 0.0903** bits

Information gain for split feature > 13.5

For true:

Instances are: d14

Partition entropy : 0

For False:

Instances are : d1,d2, d3,d4,d5, d6,d7,d8, d9,d10,d11,d12,d13

Partition entropy : 0.8905

Rem: (1/14*0) + (13/14*0.8905)= 0.8269

Information gain = 0.9403- 0.8269= 0.1134 bits

From the information gain calculated above we can observe that information gain for ID feature is less than the information gain for feature Age . Therefore, we should not consider this suggestion of making ID as the root node of the decision tree.

Ques. use sklearn.datasets import load_iris dataset. Can you show how the tree classification performances are affected by various ensemble techniques?

Step 1: Iris data set is imported from sklearn datasets.

In [2]:
```
data = load_iris()
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df1 = pd.DataFrame(load_iris().target, columns=['Species'])
df = pd.concat([df,df1], axis=1)
df.head()
```

Out[2]:

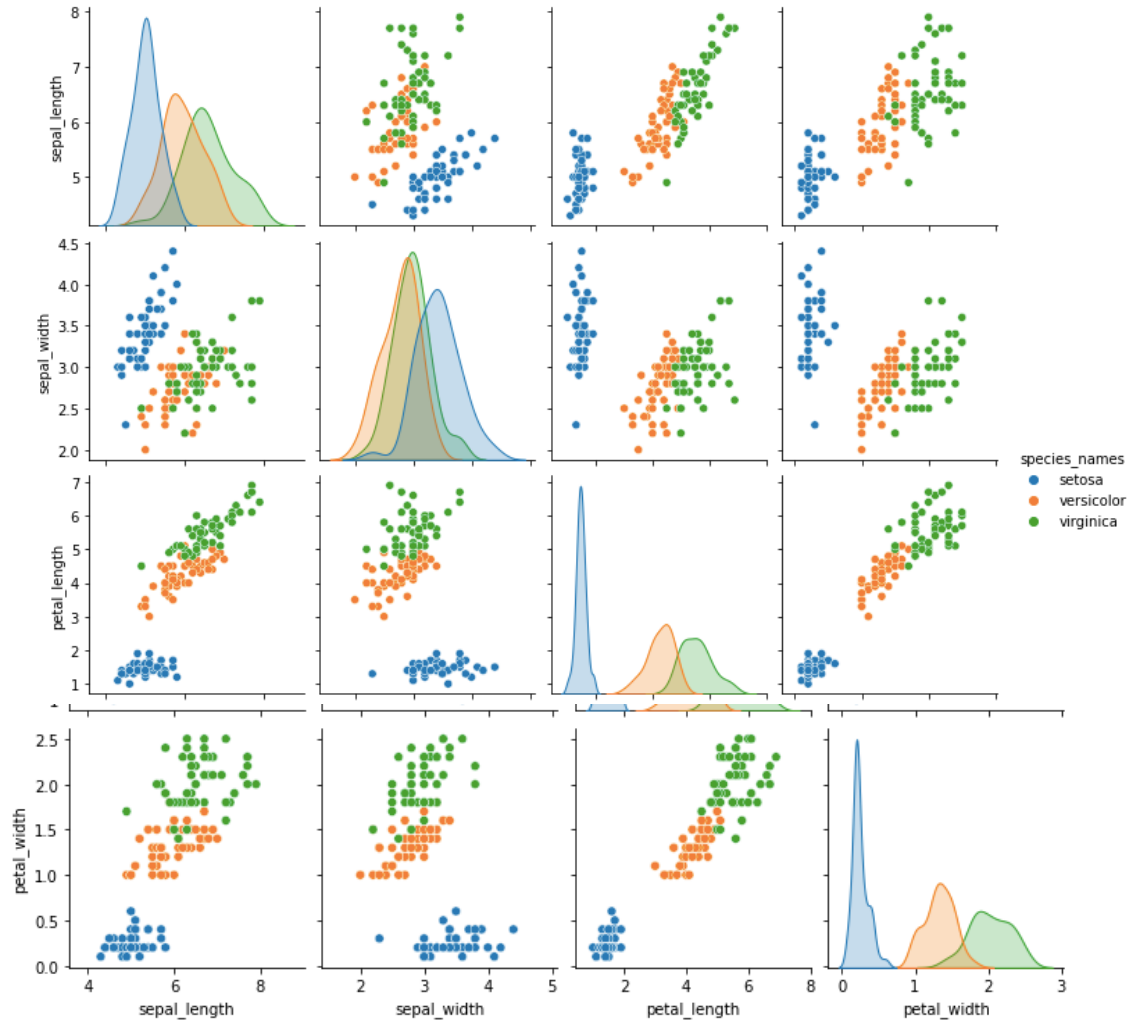
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [3]:
```
df.columns= ["sepal_length", "sepal_width","petal_length","petal_width","species"]
```

In [4]:
```
df.info()
# checking the data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

Step 2: Pairplot is drawn to visualize relationship between different features



Step 3: decision tree model is drawn with max depth = 2 and without any max depth. Accuracy of the simple decision tree model is 60% only.

Decision Tree with max depth =2

In [10]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

In [11]:
```
iris= load_iris()
X1 = iris.data[:, 2:] # petal length and width
y1 = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X1, y1)
```

Out[11]:
```
DecisionTreeClassifier(max_depth=2, random_state=42)
```

In [12]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X1, y1, test_size=0.3, random_state=0)
Decision_Tree = DecisionTreeClassifier(criterion='gini',min_samples_split=1.0,max_depth=2)
Decision_Tree.fit(X_train, Y_train)
Y_pred = Decision_Tree.predict(X_test)
```

In [13]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print('Classification report:')
print(classification_report(Y_test,Y_pred))
```

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.00      0.00      0.00        18
           2       0.38      1.00      0.55        11

    accuracy                           0.60        45
   macro avg       0.46      0.67      0.52        45
weighted avg       0.45      0.60      0.49        45
```

:
```
print('Confusion Matrix:')
print(confusion_matrix(Y_test,Y_pred))
```

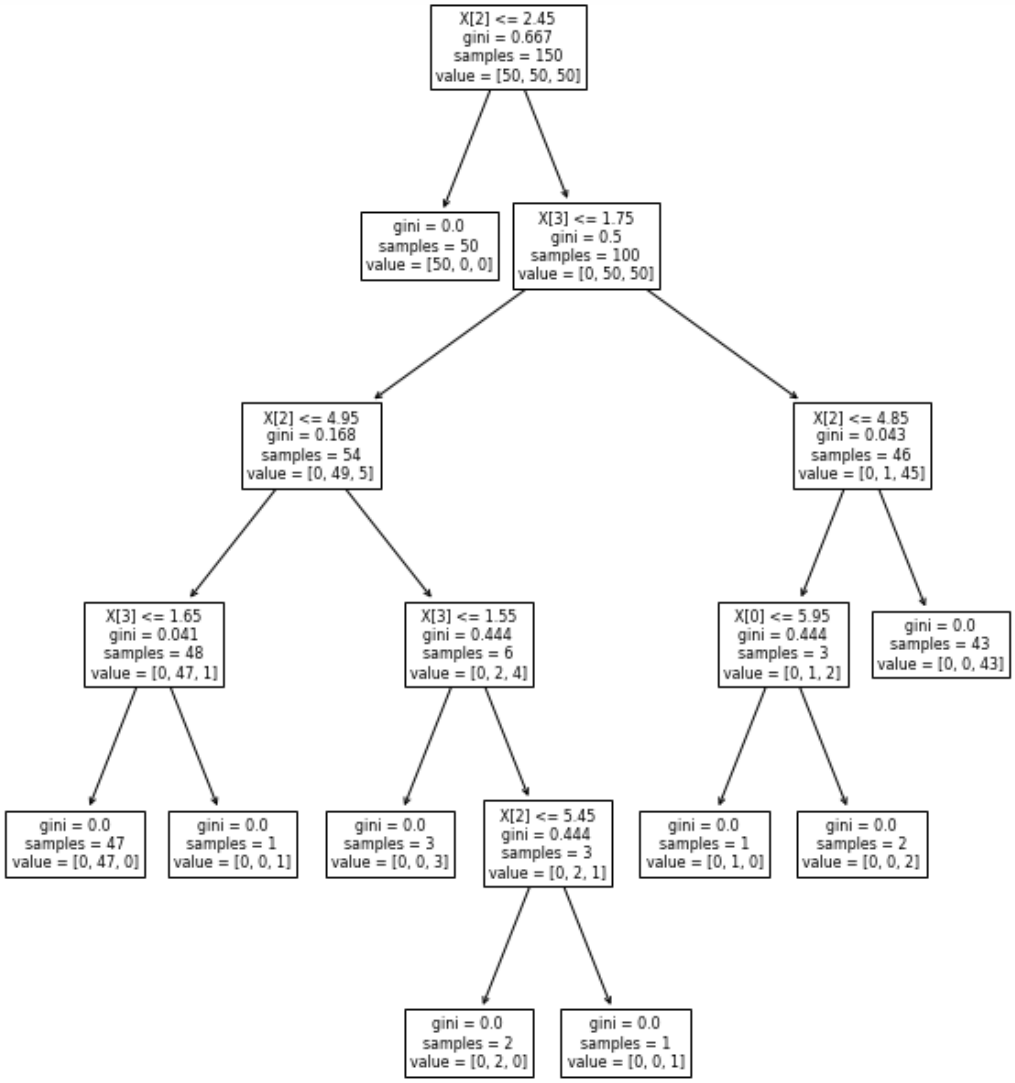
```
Confusion Matrix:
[[16  0  0]
 [ 0  0 18]
 [ 0  0 11]]
```

:
```
print('Accuracy Score:')
print(accuracy_score(Y_test,Y_pred))
```

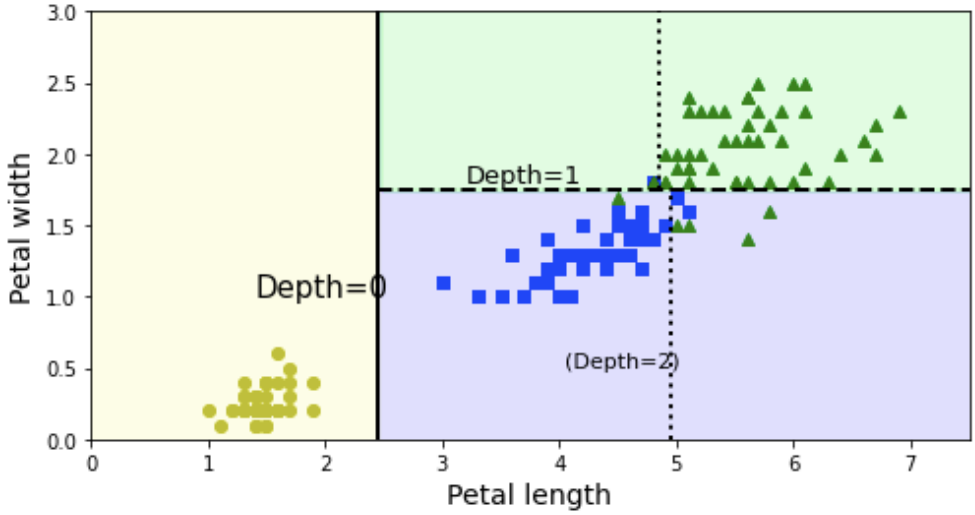
```
Accuracy Score:
0.6
```

Step 4: Tree is drawn showing Gini index and scores at all nodes.

```
clf = clf.fit(iris.data, iris.target)
fig = plt.figure(figsize=(12,14))
tree.plot_tree(clf)
plt.show()
```



Step 5: decision boundaries was drawn for the above decision tree Model



Step 6 : Now different ensemble model techniques are implemented. First is the voting classifier. It's a combination of different ML models used on same training set.

In [23]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

In [24]:
```
X = iris.data
y = iris.target
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y,random_state= 42)
```

In [26]:
```
svc = SVC()
knn = KNN()
dt = DecisionTreeClassifier()

```

In [27]:
```
classifiers = [('Support Vector classification', svc),('K Nearest Neighbours', knn),('Classification Tree', dt)]

```

In [28]:
```
for clf_name, clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print('{:s} : {:.3f}'.format(clf_name, accuracy_score(y_test, y_pred)))
```

```
Support Vector classification : 0.956
K Nearest Neighbours : 0.978
Classification Tree : 0.933
```

In [29]:
```
vc = VotingClassifier(estimators=classifiers)
```

In [30]:
```
vc.fit(X_train, y_train)
y_pred = vc.predict(X_test)
```

In [31]:
```
from sklearn import metrics
print('Accuracy of Voting Classifier: {:.3f}%'.format(metrics.accuracy_score(y_test, y_pred)*100))
```

```
Accuracy of Voting Classifier: 95.556%
```

```
Observation:
Accuracy increased from 60% to 95% using a combination of models
```

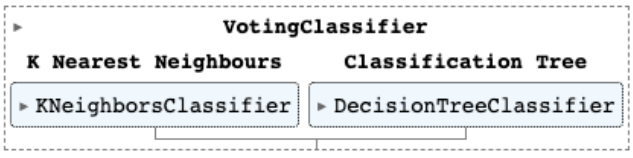
Plot

In [32]:
```
X1 = iris.data[:, 2:] # petal length and width
y1 = iris.target
```

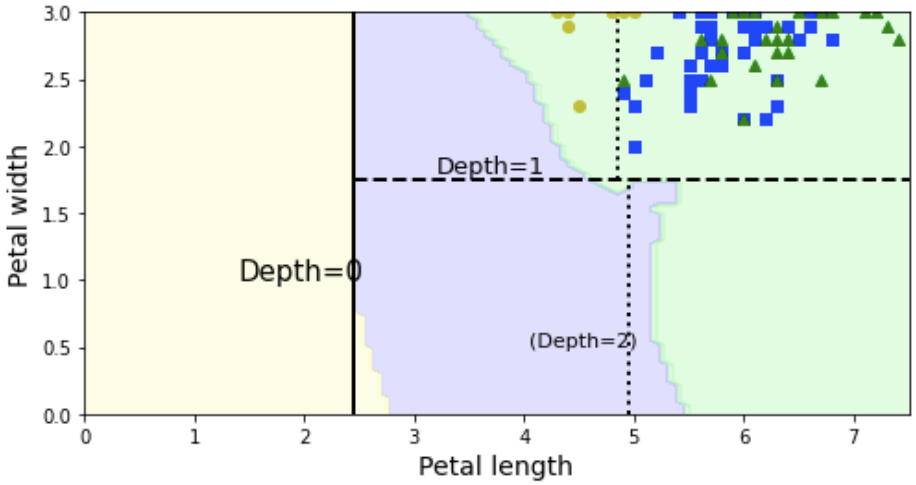
In [33]:
```
classifiers1 = [('K Nearest Neighbours', knn),('Classification Tree', dt)]
```

In [34]:
```
vc1 = VotingClassifier(estimators=classifiers1)
vc1.fit(X1, y1)
```

Out[34]:



```
plt.figure(figsize=(8, 4))
plot_decision_boundary(vc1, X, y)
plt.plot([2.45, 2.45], [0, 3], "k-", linewidth=2)
plt.plot([2.45, 7.5], [1.75, 1.75], "k--", linewidth=2)
plt.plot([4.95, 4.95], [0, 1.75], "k:", linewidth=2)
plt.plot([4.85, 4.85], [1.75, 3], "k:", linewidth=2)
plt.text(1.40, 1.0, "Depth=0", fontsize=15)
plt.text(3.2, 1.80, "Depth=1", fontsize=13)
plt.text(4.05, 0.5, "(Depth=2)", fontsize=11)
plt.show()
```



Observation:
Accuracy increased from 60% to 95% using a combination of models.

Step 7: <u>Bagging: In this we have same algorithm but different subsets of training set. The samples from the data are taken with replacement.</u>

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

```
X = iris.data
y = iris.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y,random_state= 42)
```

```
dt = DecisionTreeClassifier(max_depth=4, min_samples_leaf=0.16, random_state=42, criterion= 'gini')
bc = BaggingClassifier(base_estimator=dt, n_estimators=500, n_jobs=-1, max_samples= 100,bootstrap=True)
# n_jobs means the number of jobs to run in parallel for both fit and predict.
# setting to -1 means using all processors.
```

```
bc.fit(X_train, y_train)
y_pred = bc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy of Bagging Classifier: {:.3f}%'.format(accuracy*100))
```

```
Accuracy of Bagging Classifier: 93.333%
```

```
# now checking the accuracy by calculating the information gain using shannon entropy
```

```
dt1 = DecisionTreeClassifier( min_samples_leaf=0.16, random_state=42, criterion= 'entropy')
bc1 = BaggingClassifier(base_estimator=dt1, n_estimators=500, n_jobs=1, max_samples= 100,bootstrap=True)
# setting the n_jobs to 1 this time and max_Depth None
```

```
bc1.fit(X_train, y_train)
y_pred = bc1.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy of Bagging Classifier: {:.3f}%'.format(accuracy*100))
```

```
Accuracy of Bagging Classifier: 93.333%
```

```
# the above results shows that accuracy is same with both the criterion ie gini index and entropy
```

OOB (Out of Bag Evaluation) : In bagging(which uses bootstrapping sampling) some data points might not be called at all. On average apprx 63% of the samples are trained , rest other are not seen by the model at all. Hence we use these data points to estimate the performance without the need to cross validation.

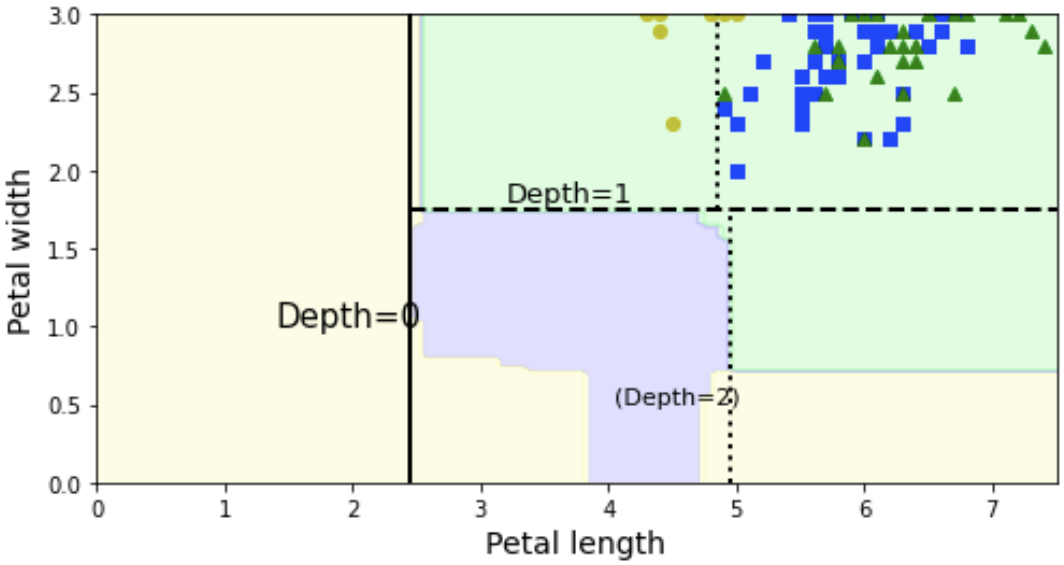
```
dt = DecisionTreeClassifier(max_depth=4, min_samples_leaf=0.16, random_state=42, criterion= 'gini')
bc = BaggingClassifier(base_estimator=dt, n_estimators=500, n_jobs=-1, max_samples= 100,bootstrap=True,oob_score=Tr
# setting the oob_score to True to estimate the performance of the model
```

```
bc.fit(X_train, y_train)
y_pred = bc.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)
oob_accuracy = bc.oob_score_
print('Test set accuracy: {:.3f}5'.format(test_accuracy*100))
print('OOB accuracy: {:.3f}%'.format(oob_accuracy*100))
```

```
Test set accuracy: 93.3335
OOB accuracy: 95.238%
```

```
Observation:
Accuracy increased from 60% to 95.2% throgh bootstrapping sampling
```

Observation:
Accuracy increased from 60% to 95.2% throgh bootstrapping sampling



Step 8: Random Forest : In this base estimator is decision tree. Each estimator is trained on different bootstrap sample. However, it has more randomness by using k features at each node without replacement where k < total features.

```
seed= 1 # setting the seed for reproducibilty
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y,random_state=seed )
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
rf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
test_accuracy1= metrics.accuracy_score(y_test, y_pred)
print('Random Forest Accuracy is: {:.3f}'.format(test_accuracy1))
```

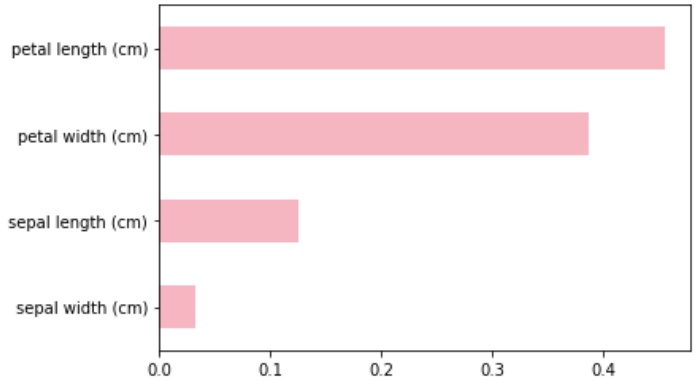
```
Random Forest Accuracy is: 0.978
```

```
Observation:
Accuracy increased from 60% to 97.8% throgh random forest model
```

Observation:
Accuracy increased from 60% to 97.8% through random forest model.

Calculating the Feature Importance

```
import matplotlib.pyplot as plt
importances_rf = pd.Series(rf.feature_importances_, index= iris["feature_names"])
sorted_importances_rf = importances_rf.sort_values()
sorted_importances_rf.plot(kind='barh', color='lightpink'); plt.show()
```




```
for name, importance_score in zip(iris["feature_names"], rf.feature_importances_):
    print(name, importance_score)
```

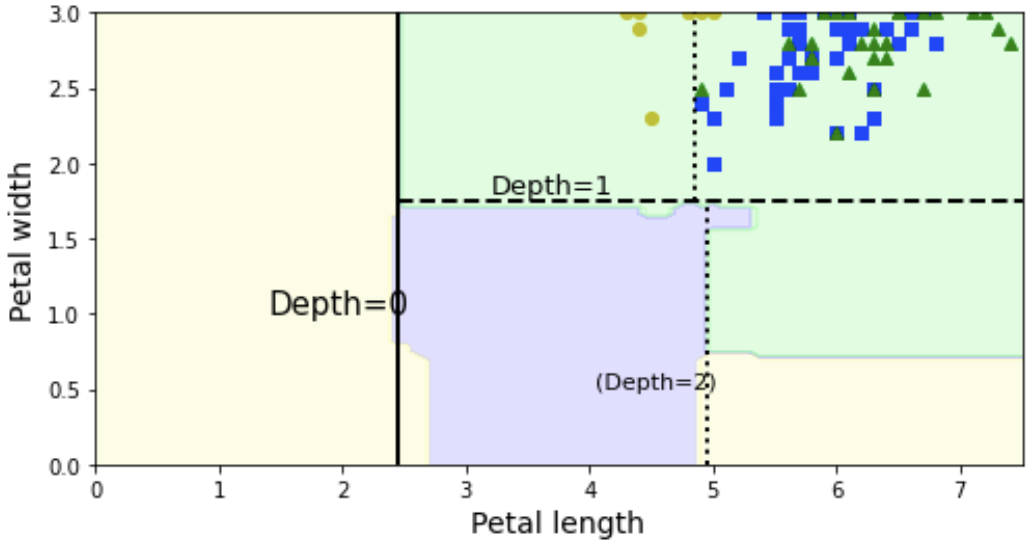
```
sepal length (cm) 0.12602766868662005
sepal width (cm) 0.03311849005976816
petal length (cm) 0.45468186118159715
petal width (cm) 0.3861719800720147
```

Plot

```
rf2 = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, random_state=42)
rf2.fit(X1, y1)
```

```
RandomForestClassifier
RandomForestClassifier(max_leaf_nodes=16, n_estimators=500, random_state=42)
```

```
plt.figure(figsize=(8, 4))
plot_decision_boundary(rf2, X, y)
plt.plot([2.45, 2.45], [0, 3], "k-", linewidth=2)
plt.plot([2.45, 7.5], [1.75, 1.75], "k--", linewidth=2)
plt.plot([4.95, 4.95], [0, 1.75], "k:", linewidth=2)
plt.plot([4.85, 4.85], [1.75, 3], "k:", linewidth=2)
plt.text(1.40, 1.0, "Depth=0", fontsize=15)
plt.text(3.2, 1.80, "Depth=1", fontsize=13)
plt.text(4.05, 0.5, "(Depth=2)", fontsize=11)
plt.show()
```



Step 9 : Boosting : This is done by incrementally adapting the dataset used to train the models. Most popular methods are ADA(adaptive) boosting and gradient boosting.

ADA Boosting

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import roc_auc_score
```

```
seed=1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y,random_state= seed)
```

```
dt = DecisionTreeClassifier(max_depth=1)

adaclf = AdaBoostClassifier(base_estimator=dt, n_estimators=10, algorithm="SAMME.R", learning_rate=0.01, random_
adaclf.fit(X_train, y_train)
```

AdaBoostClassifier
base_estimator: DecisionTreeClassifier
DecisionTreeClassifier

```
y_pred_proba = adaclf.predict_proba(X_test)
roc_auc_score = metrics.roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
```

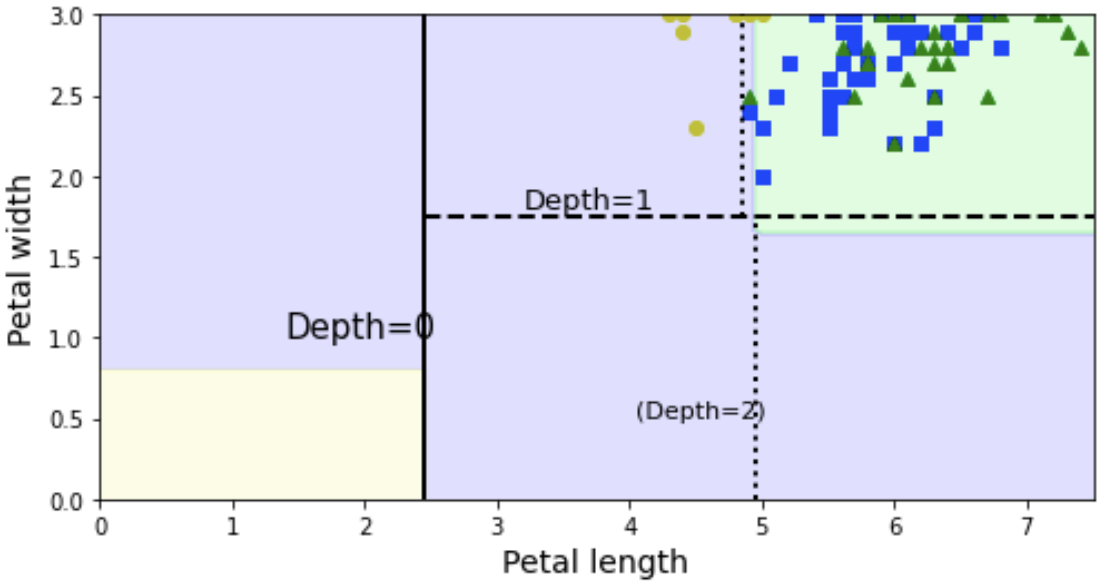
```
test_Accuracy= adaclf.score(X_test, y_test)
print('Accuracy Score: {:.3f}'.format(test_Accuracy))
```

```
Accuracy Score: 0.956
```

```
print('ROC AUC Score: {:.3f}'.format(roc_auc_score))
```

```
ROC AUC Score: 0.988
```

Observation:
Accuracy increased from 60% to 95.6% through ADA boosting at learning rate 0.01. ROC-AUC score is 98%



Gradient boosting
Learning rate 0.01

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,stratify=y,random_state= 42)
```

```
# taking the learning rate as 0.01
gbt = GradientBoostingClassifier(n_estimators=300, max_depth=1, random_state=42, learning_rate= 0.01)
```

```
gbt.fit(X_train, y_train)
```

```
GradientBoostingClassifier(learning_rate=0.01, max_depth=1, n_estimators=300,
                           random_state=42)
```

```
y_pred = gbt.predict(X_test)
```

```
test_Accuracy= gbt.score(X_test, y_test)
print('Accuracy Score: {:.3f}%'.format(test_Accuracy*100))
```

```
Accuracy Score: 97.778%
```

Learning rate 0.1 and 1

```
# taking the learning rate as 0.1
gbt = GradientBoostingClassifier(n_estimators=300, max_depth=1, random_state=42, learning_rate= 0.1)
```

```
gbt.fit(X_train, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(max_depth=1, n_estimators=300, random_state=42)
```

```
y_pred = gbt.predict(X_test)
```

```
test_Accuracy= gbt.score(X_test, y_test)
print('Accuracy Score: {:.3f}%'.format(test_Accuracy*100))
```

```
Accuracy Score: 93.333%
```

```
# taking the learning rate as 1
gbt = GradientBoostingClassifier(n_estimators=300, max_depth=1, random_state=42, learning_rate= 1)
```

```
gbt.fit(X_train, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=1, max_depth=1, n_estimators=300,
                           random_state=42)
```

```
y_pred = gbt.predict(X_test)
```

```
test_Accuracy= gbt.score(X_test, y_test)
print('Accuracy Score: {:.3f}%'.format(test_Accuracy*100))
```

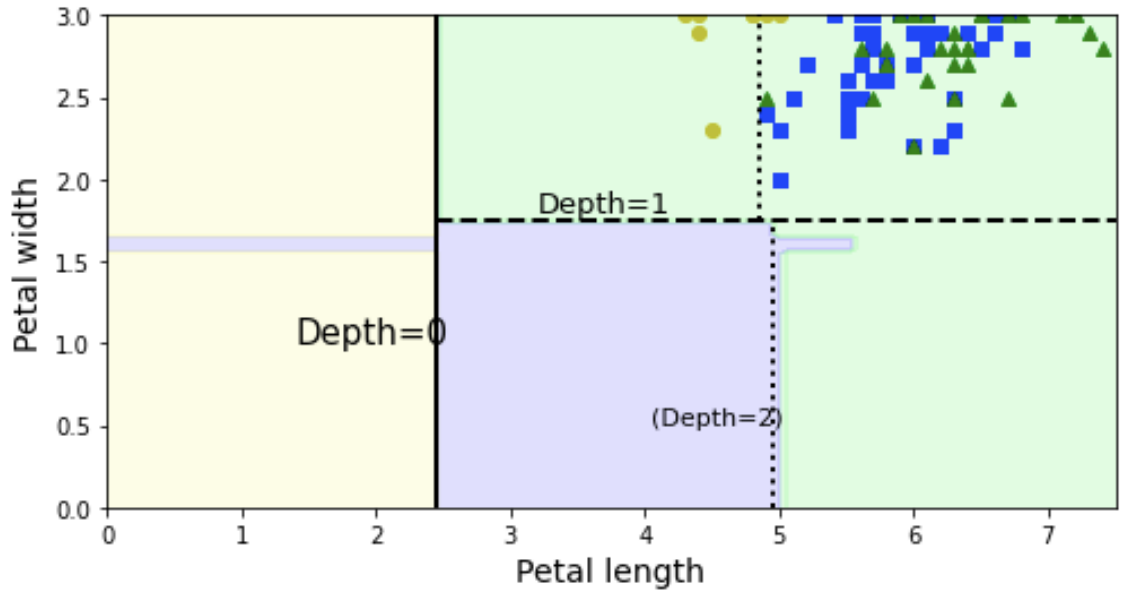
```
Accuracy Score: 93.333%
```

```
Observation:
Accuracy increased from 60% to 97.8% throgh gradient boosting at learning rate 0.01
```

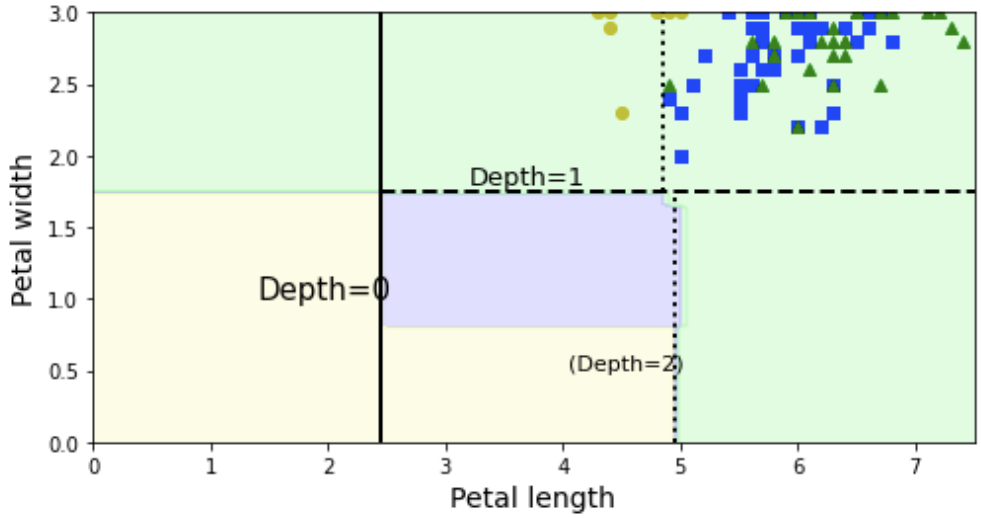
Observation:

Accuracy increased from 60% to 97.8% through gradient boosting at learning rate 0.01.

Plot at learning rate : 1



Plot at learning rate: 0.01



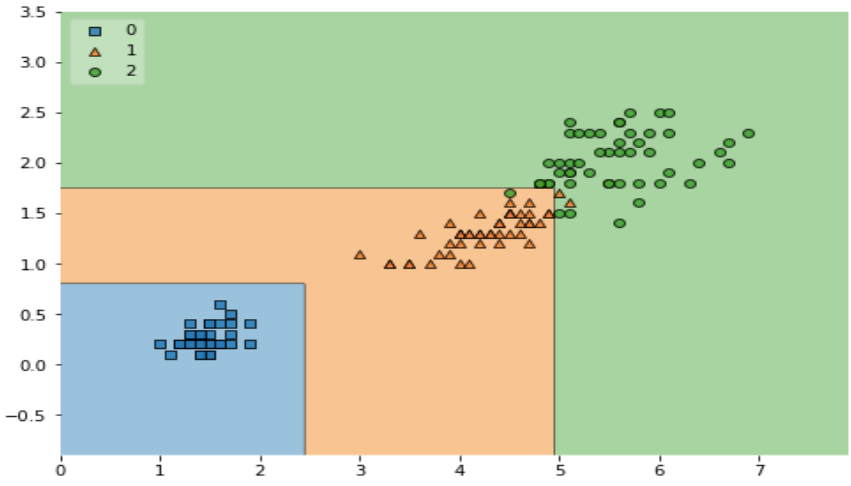
Step 10: Measuring the accuracy through cross validation score for different models

```
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from mlxtend.classifier import EnsembleVoteClassifier
from sklearn.model_selection import cross_val_score

adaboosting = AdaBoostClassifier(n_estimators=100)
gradientboosting = GradientBoostingClassifier(n_estimators=100)


ensemblemodel = EnsembleVoteClassifier(clfs=[adaboosting, gradientboosting], voting='hard')
labels = ['Ada Boosting', 'Gradient Boosting', 'Ensemble model']

for model, labels in zip([adaboosting, gradientboosting, ensemblemodel], labels):
    scores = cross_val_score(model, X1, y1, cv=4, scoring='accuracy')
    print("Accuracy: {0:.3f}%, Variance:{1:.3f} [{2}]".format(scores.mean()*100, scores.std(), labels))
```

```
Accuracy: 95.324%, Variance:0.040 [Ada Boosting]
Accuracy: 96.675%, Variance:0.022 [Gradient Boosting]
Accuracy: 95.324%, Variance:0.040 [Ensemble model]
```

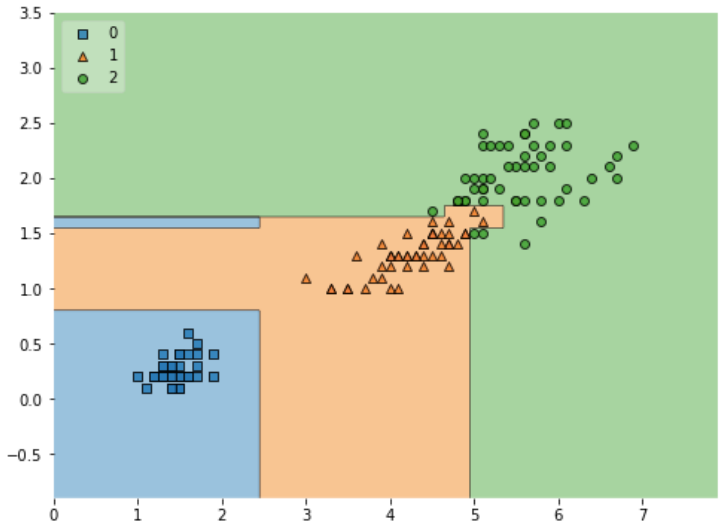
Plot for Adaptive Boosting

```
from mlxtend.plotting import plot_decision_regions
fig = plt.figure(figsize=(8,6))
clf= adaboosting.fit(X1,y1)
plot_decision_regions(X1, y1, clf=clf, legend=2)
plt.show()
```



Plot for Gradient Boosting

```
from mlxtend.plotting import plot_decision_regions
fig = plt.figure(figsize=(8,6))

clf= gradientboosting.fit(X1,y1)
plot_decision_regions(X1, y1, clf=clf, legend=2)
plt.show()
```



Results :

1.Bagging approach has low accuracy(ie. 0.933) compared to different boosting methods

2. Different Models tested above showed that accuracy for random forest models and ADA boosting models is high. Around 97% for random forest model and 96% for ADA boosting model at learning rate 0.01.

3.Also, ADA boosting model has a good RUC AOC score of 98.8% . Roc AUC score shows the performance measurement for the ADA boosting model.

4. Gradient boosting model showed good accuracy ie. 97 % at learning rate 0.01. However, when learning increased to 0.1 its accuracy dropped to 93%. This is because high learning rate does not learn from the training set accurately as it takes higher jumps.

5. From the feature extraction, we noticed that Petal width and Petal length have greater feature importance