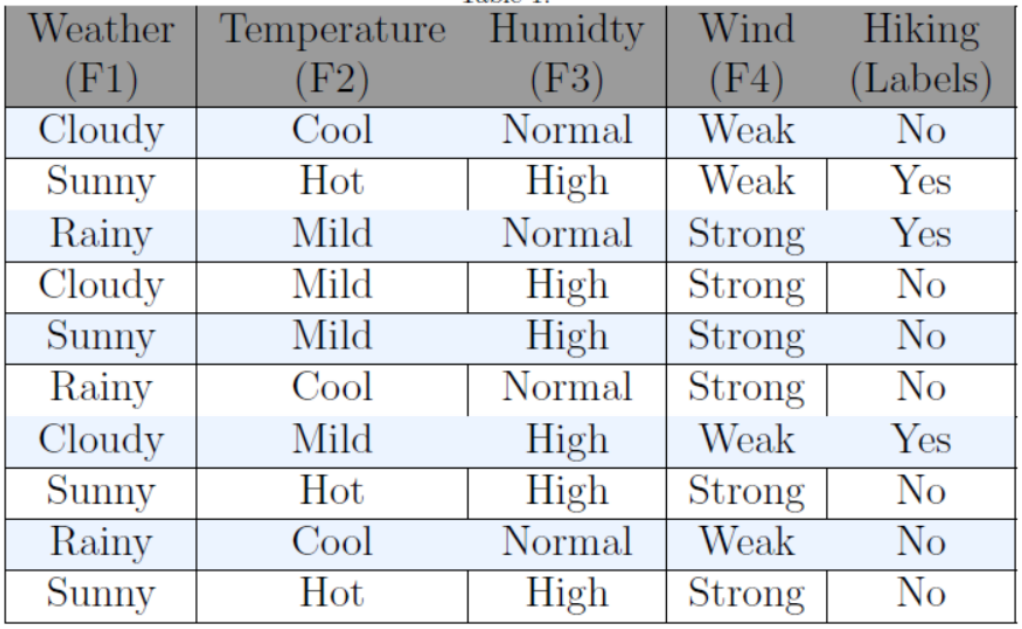
ELG5255 Applied Machine Learning

REPORT of: Group Assignment 4 (Group-18)

Part 1: Calculations

Table 1:



Weather (F1)	Temperature (F2)	Humidty (F3)	Wind (F4)	Hiking (Labels)
Cloudy	Cool	Normal	Weak	No
Sunny	Hot	High	Weak	Yes
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Rainy	Cool	Normal	Strong	No
Cloudy	Mild	High	Weak	Yes
Sunny	Hot	High	Strong	No
Rainy	Cool	Normal	Weak	No
Sunny	Hot	High	Strong	No

a) **Build a decision tree by using Gini Index (i.e., Gini = 1 - $\sum_{i=1}^{N_c} (p_i)^2$, where N_c is the number of class).**

Hiking (labels) ⇒P (Yes) = $\frac{3}{10}$, P (No) = $\frac{7}{10}$

We will calculate probabilities of classes in F1, F2, F3, and F4 in this table:

Weather (F1)	Temperature (F2)	Humidty (F3)	Wind(F4)
P(F1 = Cloudy) = $\frac{3}{10}$	P(F2 = Cool) = $\frac{3}{10}$	P(F3 = Normal) = $\frac{4}{10}$	P(F4 = Weak) = $\frac{4}{10}$
P(F1 = Sunny) = $\frac{4}{10}$	P(F2 = Hot) = $\frac{3}{10}$	P(F3 = High) = $\frac{6}{10}$	P(F4 = Strong) = $\frac{6}{10}$
P(F1 = Rainy) = $\frac{3}{10}$	P(F2 = Mild) = $\frac{4}{10}$		

We will calculate the Gini Index for Weather (F1)

Weather (F1)	
P(F1 = Cloudy and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Cloudy and Hiking = No) = $\frac{2}{3}$
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{4}$	P(F1 = Sunny and Hiking = No) = $\frac{3}{4}$
P(F1 = Rainy and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Rainy and Hiking = No) = $\frac{2}{3}$

Gini Index of Cloudy = $1-((\frac{1}{3})^2+(\frac{2}{3})^2)$ = 0.44

Gini Index of Sunny = $1-((\frac{1}{4})^2+(\frac{3}{4})^2)$ = 0.375

Gini Index of Rainy = $1-((\frac{1}{3})^2+(\frac{2}{3})^2)$ = 0.44

Weighted sum of the Gini Indices can be calculated as follows:

$$\text{Gini Index of Weather (F1)} = \frac{3}{10} * 0.44 + \frac{4}{10} * 0.375 + \frac{3}{10} * 0.44 = 0.414$$

We will calculate the Gini Index for Temperature (F2)

Temperature (F2)	
P(F2 = Cool and Hiking = Yes) = $\frac{0}{3}$	P(F2 = Cool and Hiking = No) = $\frac{3}{3}$
P(F2 = Hot and Hiking = Yes) = $\frac{1}{3}$	P(F2 = Hot and Hiking = No) = $\frac{2}{3}$
P(F2 = Mild and Hiking = Yes) = $\frac{2}{4}$	P(F2 = Mild and Hiking = No) = $\frac{2}{4}$

Gini Index of Cool = $1-((\frac{0}{3})^2+(\frac{3}{3})^2)$ = 0

Gini Index of Hot = $1-((\frac{1}{3})^2+(\frac{2}{3})^2)$ = 0.44

Gini Index of Mild = $1-((\frac{2}{4})^2+(\frac{2}{4})^2)$ = 0.5

Weighted sum of the Gini Indices can be calculated as follows:

$$\text{Gini Index of Temperature (F2)} = \frac{3}{10} * 0 + \frac{3}{10} * 0.44 + \frac{4}{10} * 0.5 = 0.332$$

We will calculate the Gini Index for Humidty (F3)

Humidty (F3)	
P(F3 = Normal and Hiking = Yes) = $\frac{1}{4}$	P(F3 = Normal and Hiking = No) = $\frac{3}{4}$
P(F3 = High and Hiking = Yes) = $\frac{2}{6}$	P(F3 = High and Hiking = No) = $\frac{4}{6}$

Gini Index of Normal = $1-((\frac{1}{4})^2+(\frac{3}{4})^2)$ = 0.375

Gini Index of High = $1-((\frac{2}{6})^2+(\frac{4}{6})^2)$ = 0.44

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Humidty (F3) = $\frac{4}{10} * 0.375 + \frac{6}{10} * 0.44 = 0.414$

We will calculate the Gini Index for Wind (F4)

Wind(F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{2}{4}$	P(F4 = Weak and Hiking = No) = $\frac{2}{4}$
P(F4 = Strong and Hiking = Yes) = $\frac{1}{6}$	P(F4 = Strong and Hiking = No) = $\frac{5}{6}$

Gini Index of Weak = $1-((\frac{2}{4})^2+(\frac{2}{4})^2)$ = 0.5

Gini Index of Strong = $1-((\frac{1}{6})^2+(\frac{5}{6})^2)$ = 0.278

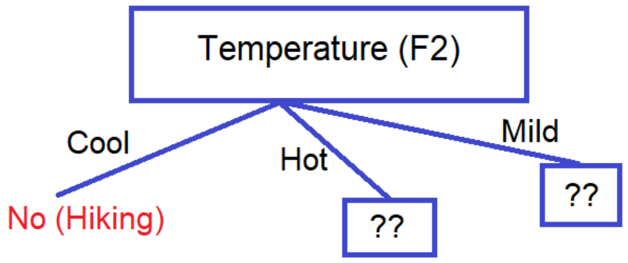
Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Wind (F4) = $\frac{4}{10} * 0.5 + \frac{6}{10} * 0.278 = 0.367$

Gini Index attributes or features

Weather (F1)	0.414
Temperature (F2)	**0.332**
Humidty (F3)	0.414
Wind (F4)	0.367

From the above table, we observe that 'Temperature (F2)' has the lowest Gini Index and hence it will be chosen as the root node for how decision tree works.



We will repeat the same procedure to determine the sub-nodes or branches of the decision tree.

We will calculate the Gini Index for the 'Hot' branch of Temperature (F2) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Strong	No

We will calculate probabilities of classes in F1, F3, and F4 in this table:

Weather (F1)	**Humidty (F3)**	**Wind(F4)**
P(F1 = Sunny) = $\frac{3}{3}$	P(F3 = High) = $\frac{3}{3}$	P(F4 = Weak) = $\frac{1}{3}$
		P(F4 = Strong) = $\frac{2}{3}$

We will calculate the Gini Index for Weather (F1)

Weather (F1)	
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Sunny and Hiking = No) = $\frac{2}{3}$

Gini Index of Sunny = $1-((\frac{1}{3})^2+(\frac{2}{3})^2) = 0.44$

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Weather (F1) = $\frac{3}{3} * 0.44 = 0.44$

We will calculate the Gini Index for Humidty (F3)

Humidty (F3)	
P(F3 = High and Hiking = Yes) = $\frac{1}{3}$	P(F3 = High and Hiking = No) = $\frac{2}{3}$

Gini Index of High = $1-((\frac{1}{3})^2+(\frac{2}{3})^2)$ = 0.44

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Humidty (F3) = $\frac{3}{3} * 0.44 = 0.44$

We will calculate the Gini Index for Wind (F4)

Wind (F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	
	P(F4 = Strong and Hiking = No) = $\frac{2}{2}$

Gini Index of Weak = $1- ((\frac{1}{1})^2)$ = 0

Gini Index of High = $1- ((\frac{2}{2})^2)$ = 0

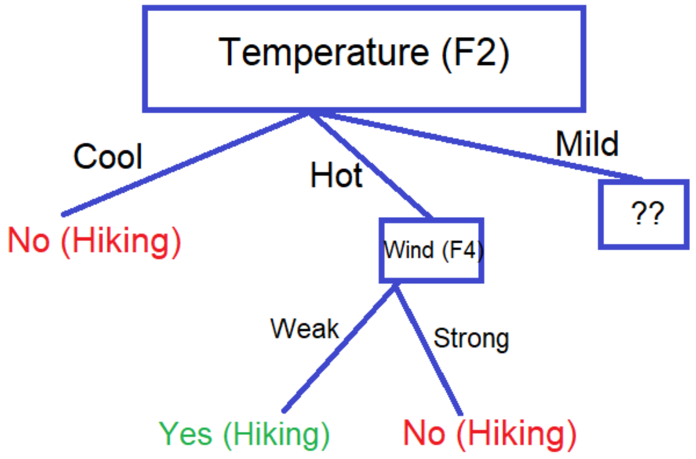
Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Wind (F4) = $\frac{1}{3} * 0 + \frac{2}{3} * 0 = 0$

Gini Index attributes or features

Weather (F1)	0.44
Humidty (F3)	0.44
Wind (F4)	**0**

From the above table, we observe that ‘Wind (F4)’ has the lowest Gini Index and hence it will be chosen as the child node for the ‘Hot’ branch of Temperature (F2).



We will calculate the Gini Index for the 'Mild' branch of Temperature (F2) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

We will calculate probabilities of classes in F1, F3, and F4 in the above table:

Weather (F1)	**Humidty (F3)**	**Wind(F4)**
$P(F1 = \text{Rainy}) = \frac{1}{4}$	$P(F3 = \text{Normal}) = \frac{1}{4}$	$P(F4 = \text{Strong}) = \frac{3}{4}$
$P(F1 = \text{Cloudy}) = \frac{2}{4}$	$P(F3 = \text{High}) = \frac{3}{4}$	$P(F4 = \text{Weak}) = \frac{1}{4}$
$P(F1 = \text{Sunny}) = \frac{1}{4}$		

We will calculate the Gini Index for Weather (F1)

Weather (F1)	
P(F1 = Rainy and Hiking = Yes) = $\frac{1}{1}$	
P(F1 = Cloudy and Hiking = Yes) = $\frac{1}{2}$	P(F1 = Cloudy and Hiking = No) = $\frac{1}{2}$
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{1}$	

Gini Index of Rainy = $1-((\frac{1}{1})^2) = 0$

Gini Index of Cloudy = $1-((\frac{1}{2})^2+(\frac{1}{2})^2) = 0.5$

Gini Index of Sunny = $1-((\frac{1}{1})^2) = 0$

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Weather (F1) = $\frac{1}{4} * 0 + \frac{2}{4} * 0.5 + \frac{1}{4} * 0 = 0.25$

We will calculate the Gini Index for Humidty (F3)

Humidty (F3)	
P(F3 = Normal and Hiking = Yes) = $\frac{1}{1}$	
P(F3 = High and Hiking = Yes) = $\frac{1}{3}$	P(F3 = High and Hiking = No) = $\frac{2}{3}$

Gini Index of Normal = $1-((\frac{1}{1})^2) = 0$

Gini Index of High = $1-((\frac{1}{3})^2+(\frac{2}{3})^2) = 0.44$

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Humidty (F3) = $\frac{1}{4} * 0 + \frac{3}{4} * 0.44 = 0.33$

We will calculate the Gini Index for Wind (F4)

Wind (F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	
P(F4 = Strong and Hiking = Yes) = $\frac{1}{3}$	P(F4 = Strong and Hiking = No) = $\frac{2}{3}$

Gini Index of Weak = $1-((\frac{1}{1})^2) = 0$

Gini Index of Strong = $1-((\frac{1}{3})^2+(\frac{2}{3})^2) = 0.44$

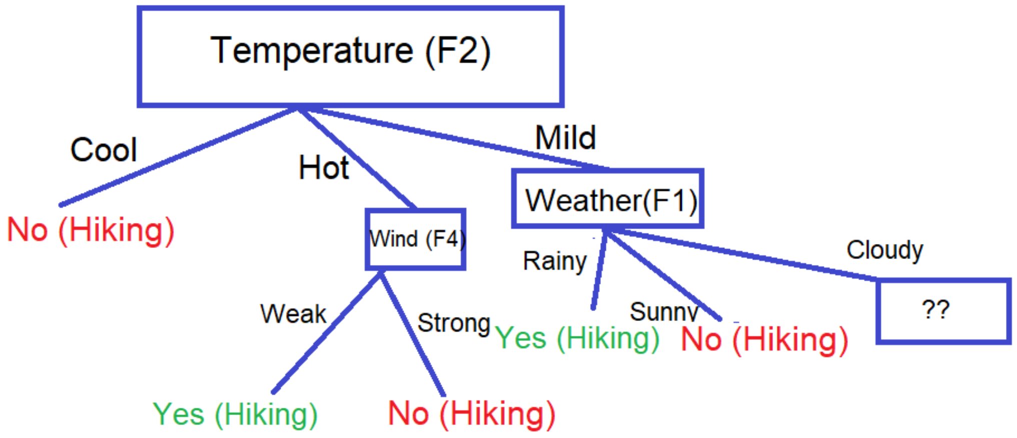
Weighted sum of the Gini Indices can be calculated as follows:

$$\text{Gini Index of Wind (F4)} = \frac{1}{4} * 0 + \frac{3}{4} * 0.44 = 0.33$$

Gini Index attributes or features

Weather (F1)	**0.25**
Humidty (F3)	0.33
Wind (F4)	0.33

From the above table, we observe that 'Weather (F1)' has the lowest Gini Index and hence it will be chosen as the child node for the 'Mild' branch of Temperature (F2).



We will calculate the Gini Index for the 'Cloudy' branch of Weather (F1) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Cloudy	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

We will calculate probabilities of classes in F1, F3, and F4 in this table:

Humidty (F3)	**Wind(F4)**
P(F3 = High) = $\frac{2}{2}$	P(F4 = Strong) = $\frac{1}{2}$
	P(F4 = Weak) = $\frac{1}{2}$

We will calculate the Gini Index for Humidty (F3)

Humidty (F3)	
P(F3 = High and Hiking = Yes) = $\frac{1}{2}$	P(F3 = High and Hiking = No) = $\frac{1}{2}$

Gini Index of High = 1-(($\frac{1}{2}$)^2+($\frac{1}{2}$)^2) = 0.5

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Humidty (F3) = $\frac{2}{2}$ * 0.5 = 0.5

We will calculate the Gini Index for Wind (F4)

Wind (F4)	
	P(F4 = Strong and Hiking = No) = $\frac{1}{1}$
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	

Gini Index of Strong = 1-(($\frac{1}{1}$)^2= 0.5

Gini Index of Weak = 1-(($\frac{1}{1}$)^2= 0.5

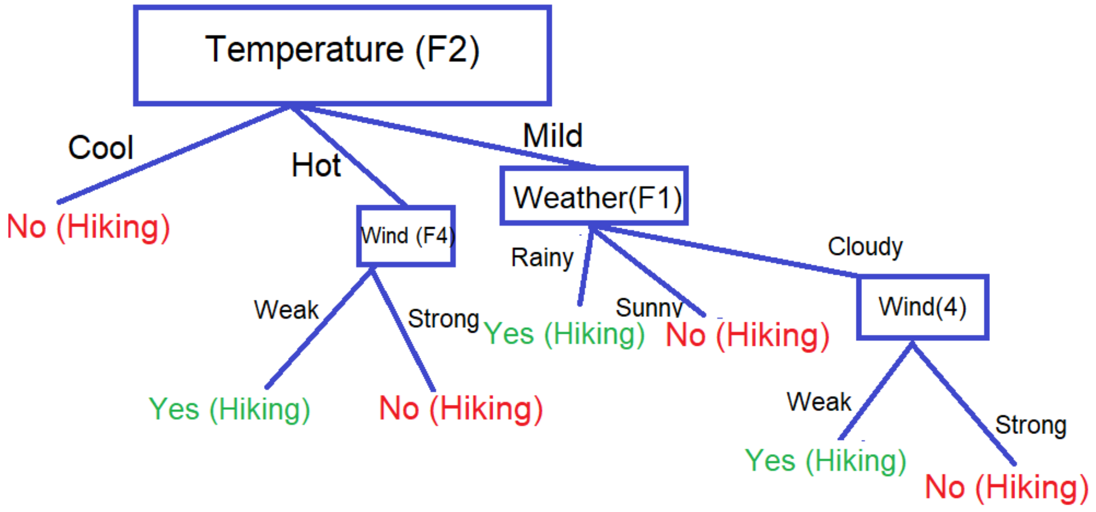
Weighted sum of the Gini Indices can be calculated as follows:

Gini Index of Wind (F4) = $\frac{1}{2}$ * 0 + $\frac{1}{2}$ * 0 = 0

Gini Index attributes or features

Humidty (F3)	0.5
Wind (F4)	**0**

From the above table, we observe that 'Wind (F4)' has the lowest Gini Index and hence it will be chosen as the child node for the 'Cloudy' branch of Weather (F1).



b) Build a decision tree by using Information Gain (i.e., IG (T, a) = Entropy (T) – Entropy (T |a), More information about IG).

The first thing that we need to do is work out which feature to use as the root node. We start by computing the entropy of hiking (labels):

 P (Yes) = $\frac{3}{10}$, P (No) = $\frac{7}{10}$

Entropy (Hiking) = - $\frac{3}{10}$ $log_2(\frac{3}{10})$ - $\frac{7}{10}$ $log_2(\frac{7}{10})$ = 0.881

We will calculate probabilities of classes in F1, F2, F3, and F4 in this table:

Weather (F1)	Temperature (F2)	Humidty (F3)	Wind(F4)
P(F1 = Cloudy) = $\frac{3}{10}$	P(F2 = Cool) = $\frac{3}{10}$	P(F3 = Normal) = $\frac{4}{10}$	P(F4 = Weak) = $\frac{4}{10}$
P(F1 = Sunny) = $\frac{4}{10}$	P(F2 = Hot) = $\frac{3}{10}$	P(F3 = High) = $\frac{6}{10}$	P(F4 = Strong) = $\frac{6}{10}$
P(F1 = Rainy) = $\frac{3}{10}$	P(F2 = Mild) = $\frac{4}{10}$		

We will calculate the Information Gain for Weather (F1)

Weather (F1)	
P(F1 = Cloudy and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Cloudy and Hiking = No) = $\frac{2}{3}$
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{4}$	P(F1 = Sunny and Hiking = No) = $\frac{3}{4}$
P(F1 = Rainy and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Rainy and Hiking = No) = $\frac{2}{3}$

GAIN (Hiking, Weather (F1)) = 0.881 - $\frac{|Hiking_{Clody}|}{10}$ Entropy $(Hiking_{Clody})$

- $\frac{|Hiking_{Sunny}|}{10}$ Entropy $(Hiking_{Sunny})$

- $\frac{|Hiking_{Rainy}|}{10}$ Entropy $(Hiking_{Rainy})$

GAIN (Hiking, Weather (F1)) = 0.881 - $\frac{3}{10}$ (- $\frac{1}{3}\ log_2(\frac{1}{3})$ - $\frac{2}{3}\ log_2(\frac{2}{3})$)

- $\frac{4}{10}$ (- $\frac{1}{4}\ log_2(\frac{1}{4})$ - $\frac{3}{4}\ log_2(\frac{3}{4})$)
- $\frac{3}{10}$ (- $\frac{1}{3}\ log_2(\frac{1}{3})$ - $\frac{2}{3}\ log_2(\frac{2}{3})$)

= 0.881 - 0.275 - 0.234 - 0.275 = 0.097

We will calculate the Information Gain for Temperature (F2)

Temperature (F2)	
P(F2 = Cool and Hiking = Yes) = $\frac{0}{3}$	P(F2 = Cool and Hiking = No) = $\frac{3}{3}$
P(F2 = Hot and Hiking = Yes) = $\frac{1}{3}$	P(F2 = Hot and Hiking = No) = $\frac{2}{3}$
P(F2 = Mild and Hiking = Yes) = $\frac{2}{4}$	P(F2 = Mild and Hiking = No) = $\frac{2}{4}$

GAIN (Hiking, Temperature (F2)) = 0.881 - $\frac{|Hiking_{Cool}|}{10}$ Entropy ($Hiking_{Cool}$)

- $\frac{|Hiking_{Hot}|}{10}$ Entropy ($Hiking_{Hot}$)
- $\frac{|Hiking_{Mild}|}{10}$ Entropy ($Hiking_{Mild}$)

GAIN (Hiking, Temperature (F2)) = 0.881 - $\frac{3}{10}$ (- $\frac{0}{3}\ log_2(\frac{0}{3})$ - $\frac{3}{3}\ log_2(\frac{3}{3})$)

- $\frac{3}{10}$ (- $\frac{1}{3}\ log_2(\frac{1}{3})$ - $\frac{2}{3}\ log_2(\frac{2}{3})$)
- $\frac{4}{10}$ (- $\frac{2}{4}\ log_2(\frac{2}{4})$ - $\frac{2}{4}\ log_2(\frac{2}{4})$)

= 0.881 - 0 - 0.275 - 0.4 = 0.206

We will calculate the Information Gain for Humidty (F3)

Humidty (F3)	
P(F3 = Normal and Hiking = Yes) = $\frac{1}{4}$	P(F3 = Normal and Hiking = No) = $\frac{3}{4}$
P(F3 = High and Hiking = Yes) = $\frac{2}{6}$	P(F3 = High and Hiking = No) = $\frac{4}{6}$

GAIN (Hiking, Humidty (F3)) = 0.881 - $\frac{|Hiking_{Normal}|}{10}$ Entropy ($Hiking_{Normal}$)

- $\frac{|Hiking_{High}|}{10}$ Entropy ($Hiking_{High}$)

GAIN (Hiking, Humidty (F3)) = 0.881 - $\frac{4}{10}$ (- $\frac{1}{4}$ $log_2(\frac{1}{4})$ - $\frac{3}{4}$ $log_2(\frac{3}{4})$)

- $\frac{6}{10}$ (- $\frac{2}{6}$ $log_2(\frac{2}{6})$ - $\frac{4}{6}$ $log_2(\frac{4}{6})$)

= 0.881 - 0.324 - 0.551 = 0.006

We will calculate the Information Gain for Wind (F4)

Wind(F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{2}{4}$	P(F4 = Weak and Hiking = No) = $\frac{2}{4}$
P(F4 = Strong and Hiking = Yes) = $\frac{1}{6}$	P(F4 = Strong and Hiking = No) = $\frac{5}{6}$

GAIN (Hiking, Wind (F4)) = 0.881 - $\frac{|Hiking_{Weak}|}{10}$ Entropy ($Hiking_{Weak}$)

- $\frac{|Hiking_{Strong}|}{10}$ Entropy ($Hiking_{Strong}$)

GAIN (Hiking, Wind (F4)) = 0.881 - $\frac{4}{10}$ (- $\frac{2}{4}$ $log_2(\frac{2}{4})$ - $\frac{2}{4}$ $log_2(\frac{2}{4})$)

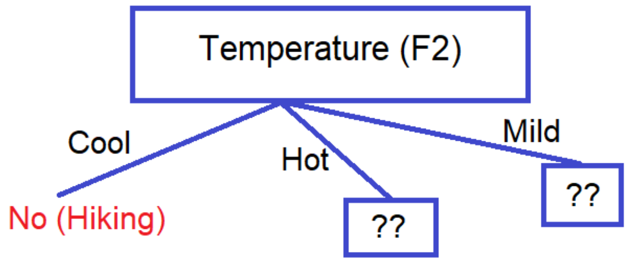
- $\frac{6}{10}$ (- $\frac{1}{6}$ $log_2(\frac{1}{6})$ - $\frac{5}{6}$ $log_2(\frac{5}{6})$)

= 0.881 - 0.4 - 0.39 = 0.091

Information Gain attributes or features

Weather (F1)	0.097
Temperature (F2)	**0.206**
Humidty (F3)	0.006
Wind (F4)	0.091

From the above table, we observe that 'Temperature (F2)' has the highest Information Gain and hence it will be chosen as the root node for how decision tree works.



We will repeat the same procedure to determine the sub-nodes or branches of the decision tree.

We will calculate the Information Gain for the 'Hot' branch of Temperature (F2) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Strong	No

We start by computing the entropy of hiking (labels) in the above table:

 P (Yes) = $\frac{1}{3}$, P (No) = $\frac{2}{3}$

Entropy (Hiking) = $-\frac{1}{3}\ log_2(\frac{1}{3}) - \frac{2}{3}\ log_2(\frac{2}{3})$ = 0.918

We will calculate probabilities of classes in F1, F3, and F4 in this table:

Weather (F1)	**Humidty (F3)**	**Wind(F4)**
P(F1 = Sunny) = $\frac{3}{3}$	P(F3 = High) = $\frac{3}{3}$	P(F4 = Weak) = $\frac{1}{3}$
		P(F4 = Strong) = $\frac{2}{3}$

We will calculate the Information Gain for Weather (F1)

Weather (F1)	
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{3}$	P(F1 = Sunny and Hiking = No) = $\frac{2}{3}$

GAIN (Hiking, Weather (F1)) = 0.918 - $\frac{|Hiking_{Sunny}|}{3}$ Entropy ($Hiking_{Sunny}$)

GAIN (Hiking, Weather (F1)) = 0.918 - $\frac{3}{3}$ (- $\frac{1}{3}$ $log_2(\frac{1}{3})$ - $\frac{2}{3}$ $log_2(\frac{2}{3})$)

= 0.918 - 0.918 = 0

We will calculate the Information Gain for Humidty (F3)

Humidty (F3)	
P(F3 = High and Hiking = Yes) = $\frac{1}{3}$	P(F3 = High and Hiking = No) = $\frac{2}{3}$

GAIN (Hiking, Humidty (F3)) = 0.918 - $\frac{|Hiking_{High}|}{3}$ Entropy ($Hiking_{High}$)

GAIN (Hiking, Humidty (F3)) = 0.918 - $\frac{3}{3}$ (- $\frac{1}{3}$ $log_2(\frac{1}{3})$ - $\frac{2}{3}$ $log_2(\frac{2}{3})$)

= 0.918 - 0.918 = 0

We will calculate the Information Gain for Wind (F4)

Wind (F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	
	P(F4 = Strong and Hiking = No) = $\frac{2}{2}$

GAIN (Hiking, Wind (F4)) = 0.918 - $\frac{|Hiking_{Weak}|}{3}$ Entropy ($Hiking_{Weak}$)

- $\frac{|Hiking_{strong}|}{3}$ Entropy ($Hiking_{strong}$)

GAIN (Hiking, Wind (F4)) = 0.918 - $\frac{1}{3}$ (- $\frac{1}{1}$ $log_2(\frac{1}{1})$)

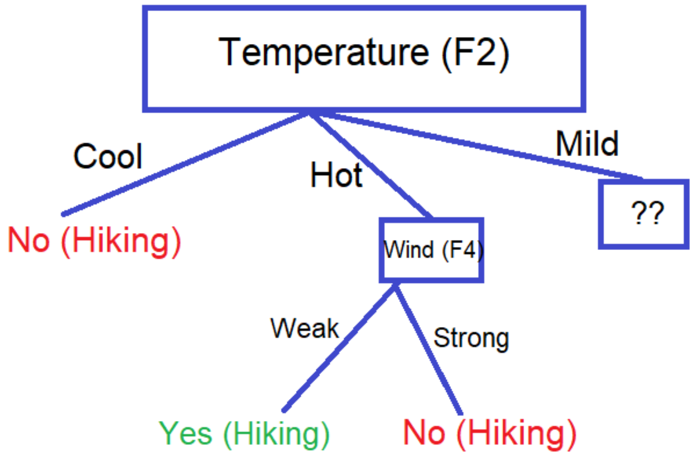
-$\frac{2}{3}$ (- $\frac{2}{2}$ $log_2(\frac{2}{2})$)

= 0.918 - 0 - 0 = 0.918

Information Gain attributes or features

Weather (F1)	0
Humidty (F3)	0
Wind (F4)	**0.918**

From the above table, we observe that 'Wind (F4)' has the highest Information Gain and hence it will be chosen as the child node for the 'Hot' branch of Temperature (F2).



We will calculate the Information Gain for the 'Mild' branch of Temperature (F2) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

We start by computing the entropy of hiking (labels) in the above table:

 P (Yes) = $\frac{2}{4}$, P (No) = $\frac{2}{4}$

Entropy (Hiking) = $-\frac{2}{4}\ log_2(\frac{2}{4}) - \frac{2}{4}\ log_2(\frac{2}{4}) = 1$

We will calculate probabilities of classes in F1, F3, and F4 in the above table:

Weather (F1)	**Humidty (F3)**	**Wind(F4)**
P(F1 = Rainy) = $\frac{1}{4}$	P(F3 = Normal) = $\frac{1}{4}$	P(F4 = Strong) = $\frac{3}{4}$
P(F1 = Cloudy) = $\frac{2}{4}$	P(F3 = High) = $\frac{3}{4}$	P(F4 = Weak) = $\frac{1}{4}$
P(F1 = Sunny) = $\frac{1}{4}$		

We will calculate the Information Gain for Weather (F1)

Weather (F1)	
P(F1 = Rainy and Hiking = Yes) = $\frac{1}{1}$	
P(F1 = Cloudy and Hiking = Yes) = $\frac{1}{2}$	P(F1 = Cloudy and Hiking = No) = $\frac{1}{2}$
P(F1 = Sunny and Hiking = Yes) = $\frac{1}{1}$	

GAIN (Hiking, Weather (F1)) = 1- $\frac{|Hiking_{Rainy}|}{4}$ Entropy ($Hiking_{Rainy}$)

- $\frac{|Hiking_{Cloudy}|}{4}$ Entropy ($Hiking_{Cloudy}$)

- $\frac{|Hiking_{Sunny}|}{4}$ Entropy ($Hiking_{Sunny}$)

GAIN (Hiking, Weather (F1)) = 1 - $\frac{1}{4}$ (- $\frac{1}{1}$ $log_2(\frac{1}{1})$) - $\frac{2}{4}$ (- $\frac{1}{2}$ $log_2(\frac{1}{2})$ - $\frac{1}{2}$ $log_2(\frac{1}{2})$) - $\frac{1}{4}$ (- $\frac{1}{1}$ $log_2(\frac{1}{1})$)

= 1 - 0 – 0.5 - 0 = 0.5

We will calculate Information Gain for Humidty (F3)

Humidty (F3)	
P(F3 = Normal and Hiking = Yes) = $\frac{1}{1}$	
P(F3 = High and Hiking = Yes) = $\frac{1}{3}$	P(F3 = High and Hiking = No) = $\frac{2}{3}$

GAIN (Hiking, Humidty (F3)) = 1- $\frac{|Hiking_{Normal}|}{4}$ Entropy ($Hiking_{Normal}$)

- $\frac{|Hiking_{High}|}{4}$ Entropy ($Hiking_{High}$)

GAIN (Hiking, Humidty (F3)) = 1 - $\frac{1}{4}$ (- $\frac{1}{1}$ $log_2(\frac{1}{1})$) - $\frac{3}{4}$ (- $\frac{1}{3}$ $log_2(\frac{1}{3})$ - $\frac{2}{3}$ $log_2(\frac{2}{3})$)

= 1 - 0 - 0.612 = 0.388

We will calculate the Information Gain for Wind (F4)

Wind (F4)	
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	
P(F4 = Strong and Hiking = Yes) = $\frac{1}{3}$	P(F4 = Strong and Hiking = No) = $\frac{2}{3}$

GAIN (Hiking, Wind (F4)) = 1- $\frac{|Hiking_{Weak}|}{4}$ Entropy $(Hiking_{Weak})$

- $\frac{|Hiking_{Strong}|}{4}$ Entropy $(Hiking_{Strong})$

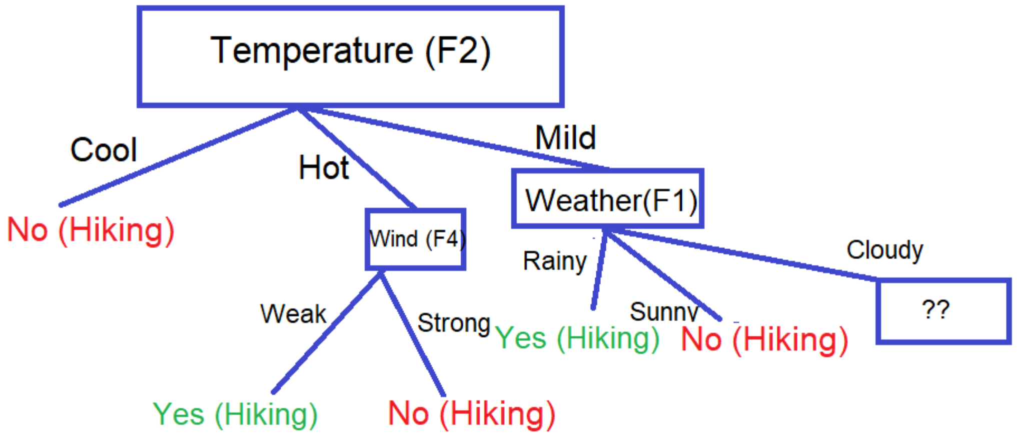
GAIN (Hiking, Wind (F4)) = 1 - $\frac{1}{4}$ (- $\frac{1}{1}$ $log_2(\frac{1}{1})$) - $\frac{3}{4}$ (- $\frac{1}{3}$ $log_2(\frac{1}{3})$ - $\frac{2}{3}$ $log_2(\frac{2}{3})$)

= 1 - 0 - 0.612 = 0.388

Information Gain attributes or features

Weather (F1)	**0.5**
Humidty (F3)	0.388
Wind (F4)	0.388

From the above table, we observe that 'Weather (F1)' has the highest Information Gain and hence it will be chosen as the child node for the 'Mild' branch of Temperature (F2).



We will calculate the Information Gain for the 'Cloudy' branch of Weather (F1) as follows:

Weather (F1)	**Temperature (F2)**	**Humidty (F3)**	**Wind(F4)**	**Hiking**
Cloudy	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

We start by computing the entropy of hiking (labels) in the above table:

- P (Yes) = $\frac{1}{2}$, P (No) = $\frac{1}{2}$

Entropy (Hiking) = $-\frac{1}{2}\ log_2(\frac{1}{2}) - \frac{1}{2}\ log_2(\frac{1}{2})$ = 1

We will calculate probabilities of classes in F1, F3, and F4 in this table:

Humidty (F3)	**Wind(F4)**
P(F3 = High) = $\frac{2}{2}$	P(F4 = Strong) = $\frac{1}{2}$
	P(F4 = Weak) = $\frac{1}{2}$

We will calculate the Information Gain for Humidty (F3)

Humidty (F3)	
P(F3 = High and Hiking = Yes) = $\frac{1}{2}$	P(F3 = High and Hiking = No) = $\frac{1}{2}$

GAIN (Hiking, Wind (F4)) = 1- $\frac{|Hiking_{High}|}{2}$ Entropy ($Hiking_{High}$)

GAIN (Hiking, Wind (F4)) = 1 - $\frac{2}{2}$ ($-\frac{1}{2}\ log_2\left(\frac{1}{2}\right) - \frac{1}{2}\ log_2(\frac{1}{2})$) = 1 – 1 = **0**

We will calculate the Gini Index for Wind (F4)

Wind (F4)	
	P(F4 = Strong and Hiking = No) = $\frac{1}{1}$
P(F4 = Weak and Hiking = Yes) = $\frac{1}{1}$	

GAIN (Hiking, Wind (F4)) = 1- $\frac{|Hiking_{strong}|}{2}$ Entropy ($Hiking_{strong}$)

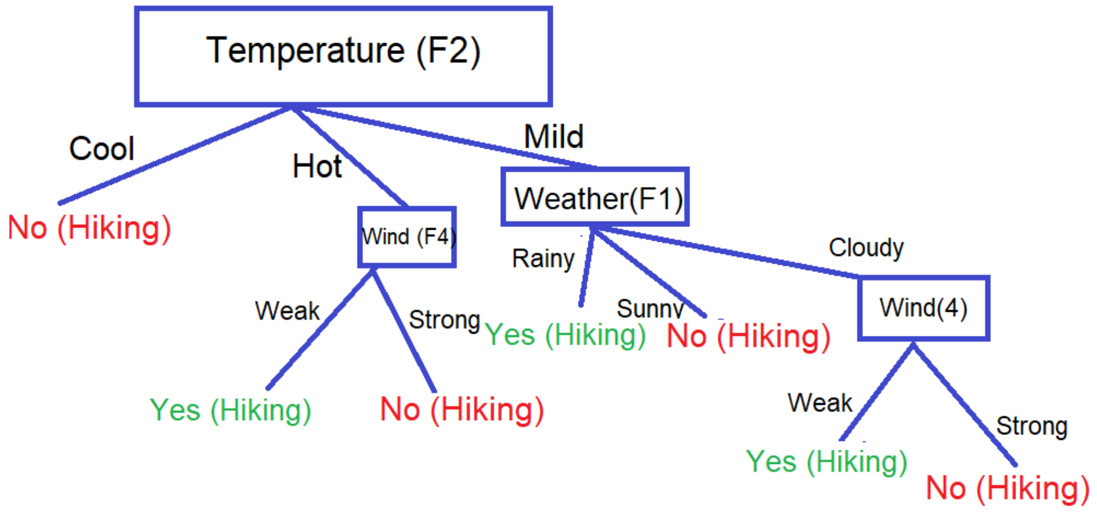
- $\frac{|Hiking_{Weak}|}{2}$ Entropy ($Hiking_{Weak}$)

GAIN (Hiking, Wind (F4)) = 1 - $\frac{1}{2}$ ($-\frac{1}{1}\ log_2\left(\frac{1}{1}\right)$) - $\frac{1}{2}$ ($-\frac{1}{1}\ log_2\left(\frac{1}{1}\right)$) = 1 – 0 - 0 = **1**

Information Gain attributes or features

Humidty (F3)	0
Wind (F4)	**1**

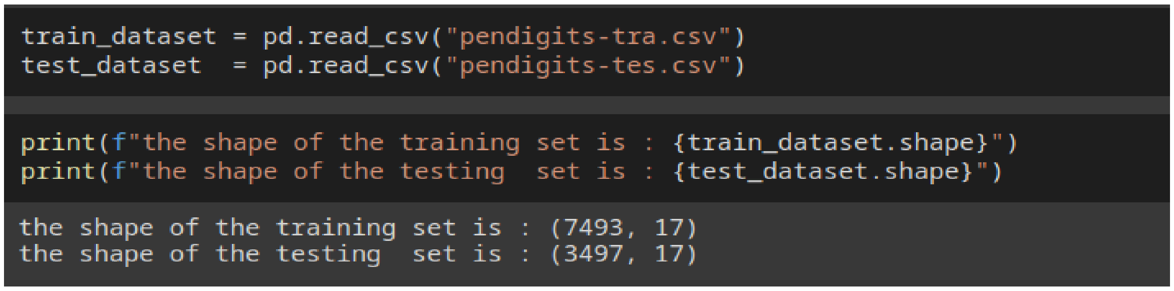
From the above table, we observe that 'Wind (F4)' has the highest Information Gain and hence it will be chosen as the child node for the 'Cloudy' branch of Weather (F1).



c) Compare the advantages and disadvantages between Gini Index and Information Gain.

	Gini Index	**Information Gain**
The advantages	□ It favors larger partitions (distributions) and is very easy to implement. □ It can handle the values that are non-negative because it is measured by subtracting the sum of squared probabilities of each class from one. □ It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient.	□ It favors partitions that have small counts but many distinct values. □ It measures the entropy differences before and after splitting and depicts the impurity in class variables. □ It uses Entropy as the base calculation; you have a wider range of results. □ It computes the difference between entropy before and after the split and specifies the impurity in-class elements. □ The feature with the highest information gain value is accounted for as the best feature to be chosen for the split.
The disadvantages	□ The Gini Index doesn’t have a wider range of results, but it caps at one. □ While working on categorical data variables, the Gini index results either in “success” or “failure” and only performs binary splitting. □ It is prone to systematic and random data errors. Therefore, inaccurate data can distort the validity of the coefficient.	□ It is not preferred as it involves a ‘log’ function that results in computational complexity. □ It can’t handle the values that are non-negative. □ It supports smaller partitions (distributions) with various distinct values; there is a need to perform an experiment with data and splitting criteria.

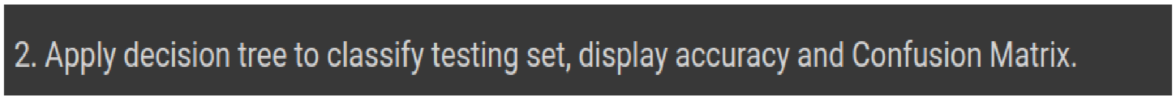
Part 2: Programming Questions



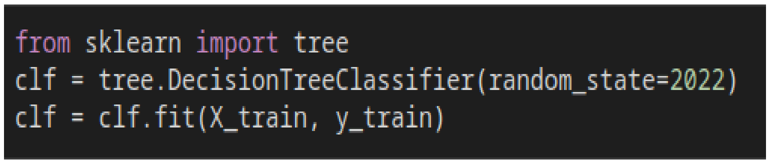
```
train_dataset = pd.read_csv("pendigits-tra.csv")
test_dataset  = pd.read_csv("pendigits-tes.csv")
```

```
print(f"the shape of the training set is : {train_dataset.shape}")
print(f"the shape of the testing  set is : {test_dataset.shape}")
```

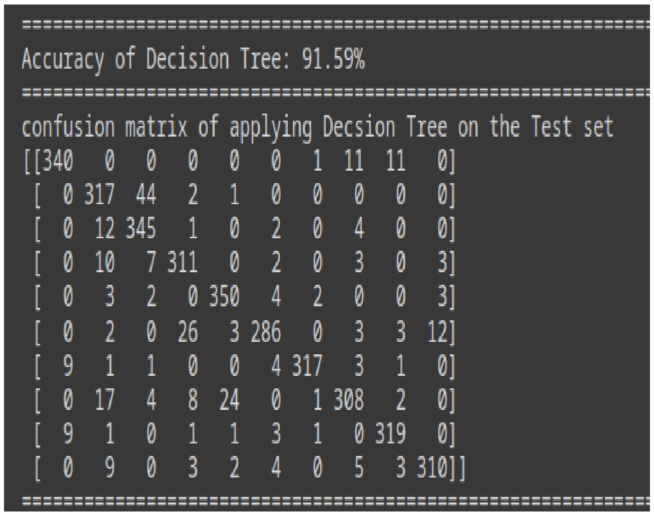
```
the shape of the training set is : (7493, 17)
the shape of the testing  set is : (3497, 17)
```



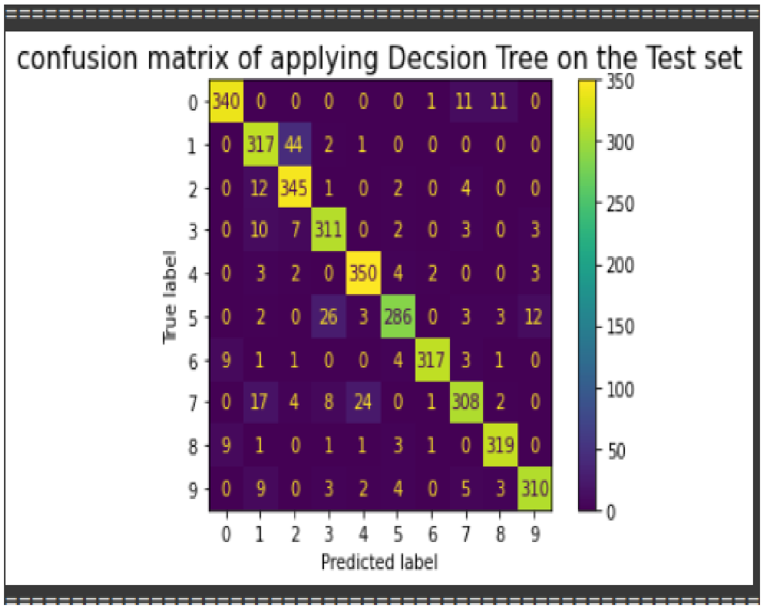
2. Apply decision tree to classify testing set, display accuracy and Confusion Matrix.

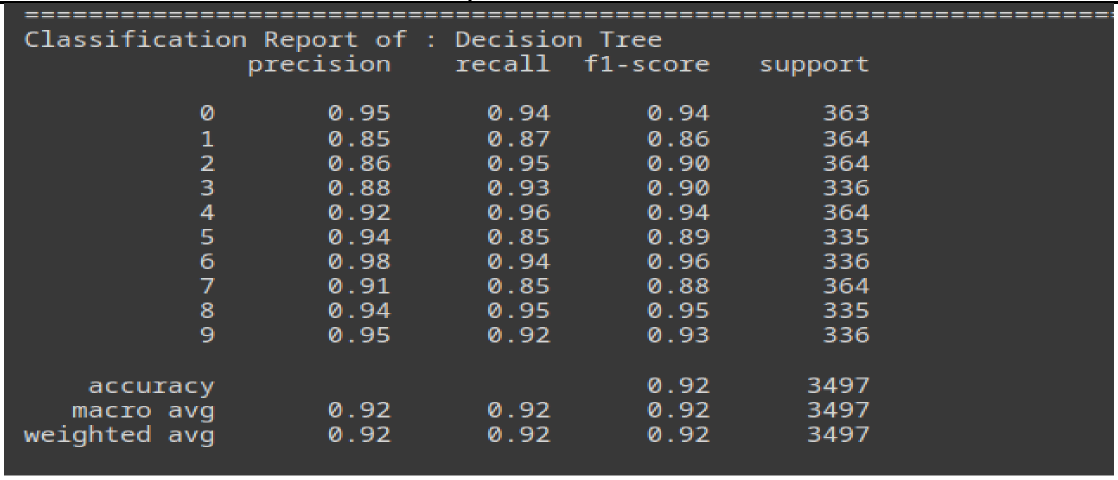


```
from sklearn import tree
clf = tree.DecisionTreeClassifier(random_state=2022)
clf = clf.fit(X_train, y_train)
```



```
==========================================================
Accuracy of Decision Tree: 91.59%
==========================================================
confusion matrix of applying Decsion Tree on the Test set
[[340   0   0   0   0   0   1  11  11   0]
 [  0 317  44   2   1   0   0   0   0   0]
 [  0  12 345   1   0   2   0   4   0   0]
 [  0  10   7 311   0   2   0   3   0   3]
 [  0   3   2   0 350   4   2   0   0   3]
 [  0   2   0  26   3 286   0   3   3  12]
 [  9   1   1   0   0   4 317   3   1   0]
 [  0  17   4   8  24   0   1 308   2   0]
 [  9   1   0   1   1   3   1   0 319   0]
 [  0   9   0   3   2   4   0   5   3 310]]
==========================================================
```





```
==========================================================
Classification Report of : Decision Tree
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       363
           1       0.85      0.87      0.86       364
           2       0.86      0.95      0.90       364
           3       0.88      0.93      0.90       336
           4       0.92      0.96      0.94       364
           5       0.94      0.85      0.89       335
           6       0.98      0.94      0.96       336
           7       0.91      0.85      0.88       364
           8       0.94      0.95      0.95       335
           9       0.95      0.92      0.93       336

    accuracy                           0.92      3497
   macro avg       0.92      0.92      0.92      3497
weighted avg       0.92      0.92      0.92      3497
```

Bagging

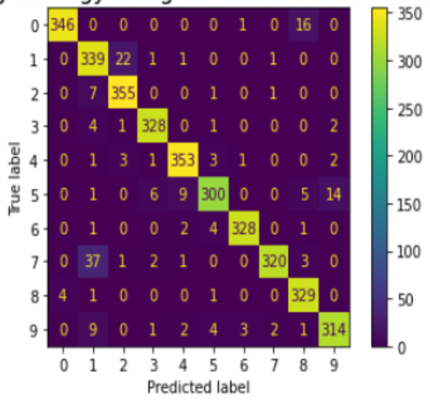
Apply bagging strategy to classify test set samples by using Decision Tree algorithm

```
applyBaggingStrategy(X_train, y_train,X_test,y_test,"the default Base estimtor (Decision Tree)",base=None)
```

```
======================================================================
Accuracy of Bagging strategy using the default Base estimtor (Decision Tree): 94.71%
======================================================================
confusion matrix of applying Bagging strategy using the default Base estimtor (Decision Tree) on the Test set
[[346   0   0   0   0   0   1   0  16   0]
 [  0 339  22   1   1   0   0   1   0   0]
 [  0   7 355   0   0   1   0   1   0   0]
 [  0   4   1 328   0   1   0   0   0   2]
 [  0   1   3   1 353   3   1   0   0   2]
 [  0   1   0   6   9 300   0   0   5  14]
 [  0   1   0   0   2   4 328   0   1   0]
 [  0  37   1   2   1   0   0 320   3   0]
 [  4   1   0   0   0   1   0   0 329   0]
 [  0   9   0   1   2   4   3   2   1 314]]
======================================================================
```

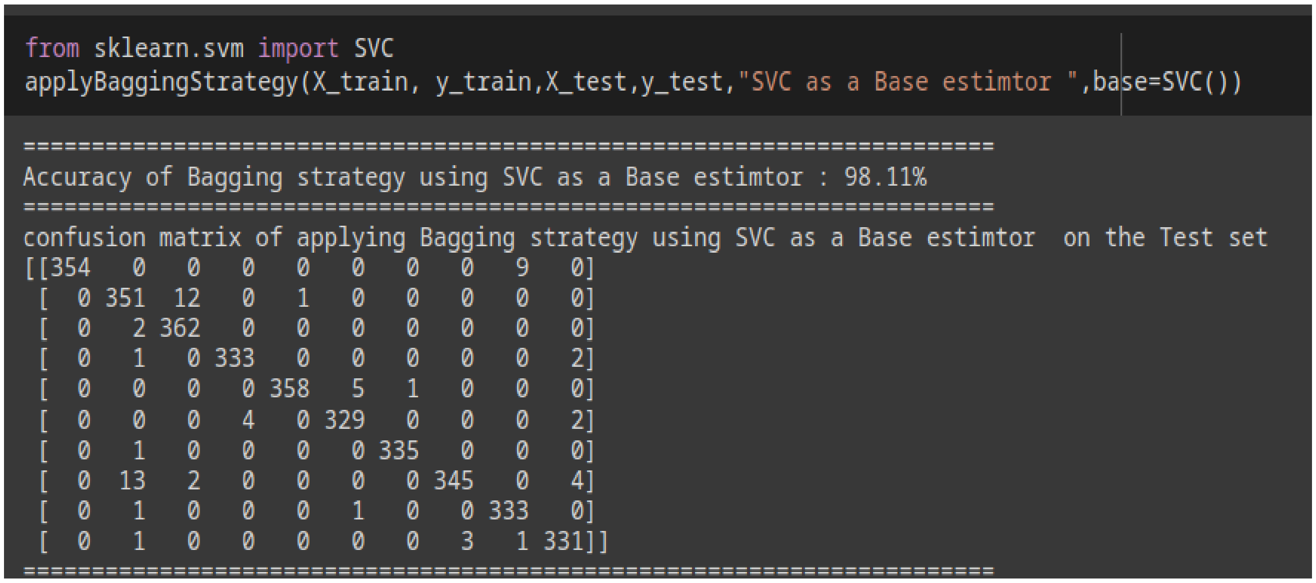
```
======================================================================
```

confusion matrix of applying Bagging strategy using the default Base estimtor (Decision Tree) on the Test set



```
======================================================================
Classification Report of : Bagging strategy using the default Base estimtor (Decision Tree)
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       363
           1       0.85      0.93      0.89       364
           2       0.93      0.98      0.95       364
           3       0.97      0.98      0.97       336
           4       0.96      0.97      0.96       364
           5       0.96      0.90      0.92       335
           6       0.98      0.98      0.98       336
           7       0.99      0.88      0.93       364
           8       0.93      0.98      0.95       335
           9       0.95      0.93      0.94       336

    accuracy                           0.95      3497
   macro avg       0.95      0.95      0.95      3497
weighted avg       0.95      0.95      0.95      3497
```

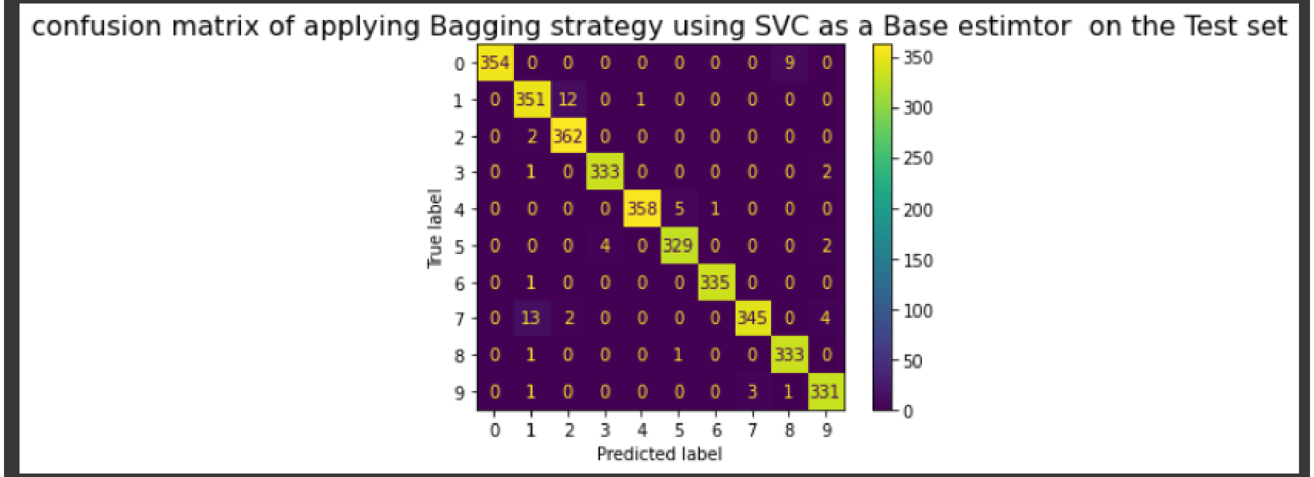
Apply bagging strategy to classify test set samples by using SVM algorithm



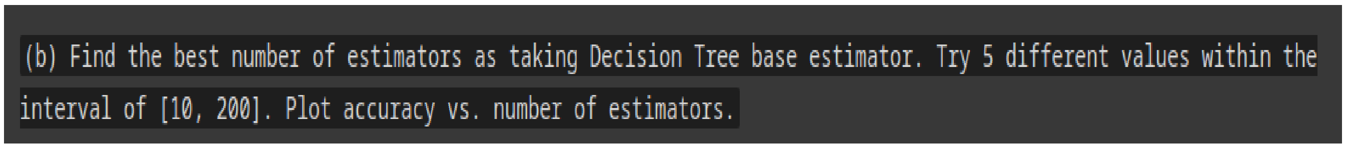
```python
from sklearn.svm import SVC
applyBaggingStrategy(X_train, y_train,X_test,y_test,"SVC as a Base estimtor ",base=SVC())
```

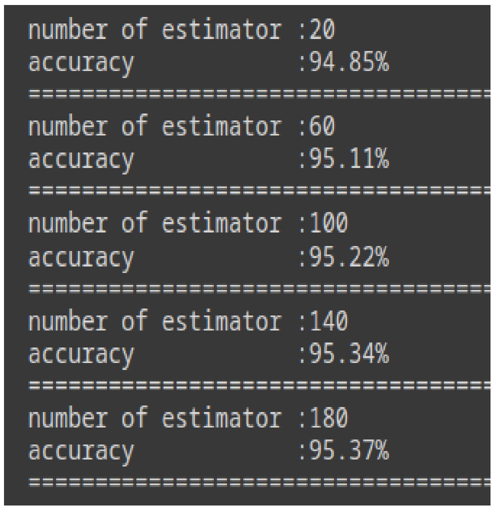
```
========================================================================
Accuracy of Bagging strategy using SVC as a Base estimtor : 98.11%
========================================================================
confusion matrix of applying Bagging strategy using SVC as a Base estimtor  on the Test set
[[354   0   0   0   0   0   0   0   9   0]
 [  0 351  12   0   1   0   0   0   0   0]
 [  0   2 362   0   0   0   0   0   0   0]
 [  0   1   0 333   0   0   0   0   0   2]
 [  0   0   0   0 358   5   1   0   0   0]
 [  0   0   0   4   0 329   0   0   0   2]
 [  0   1   0   0   0   0 335   0   0   0]
 [  0  13   2   0   0   0   0 345   0   4]
 [  0   1   0   0   0   1   0   0 333   0]
 [  0   1   0   0   0   0   0   3   1 331]]
========================================================================
```

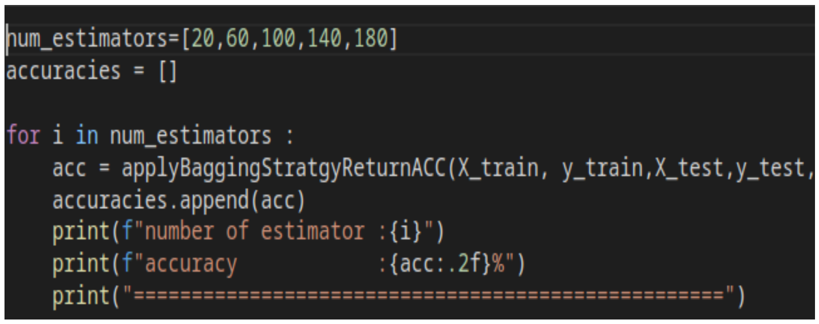
```
========================================================================
```

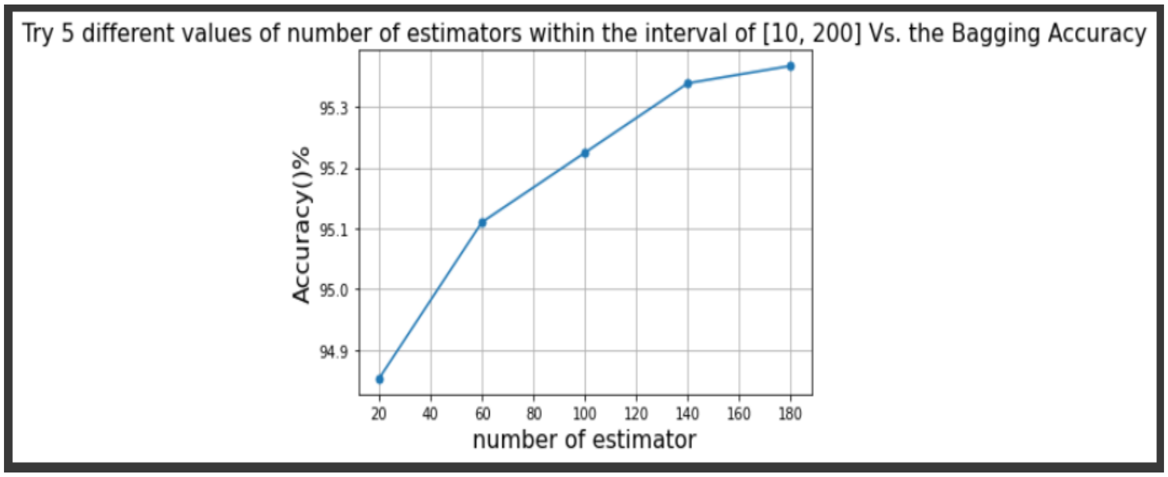


```
========================================================================
Classification Report of : Bagging strategy using SVC as a Base estimtor
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       363
           1       0.95      0.96      0.96       364
           2       0.96      0.99      0.98       364
           3       0.99      0.99      0.99       336
           4       1.00      0.98      0.99       364
           5       0.98      0.98      0.98       335
           6       1.00      1.00      1.00       336
           7       0.99      0.95      0.97       364
           8       0.97      0.99      0.98       335
           9       0.98      0.99      0.98       336

    accuracy                           0.98      3497
   macro avg       0.98      0.98      0.98      3497
weighted avg       0.98      0.98      0.98      3497
```


(b) Find the best number of estimators as taking Decision Tree base estimator. Try 5 different values within the interval of [10, 200]. Plot accuracy vs. number of estimators.


```
number of estimator :20
accuracy            :94.85%
====================================
number of estimator :60
accuracy            :95.11%
====================================
number of estimator :100
accuracy            :95.22%
====================================
number of estimator :140
accuracy            :95.34%
====================================
number of estimator :180
accuracy            :95.37%
====================================
```


```
num_estimators=[20,60,100,140,180]
accuracies = []

for i in num_estimators :
    acc = applyBaggingStratgyReturnACC(X_train, y_train,X_test,y_test,
    accuracies.append(acc)
    print(f"number of estimator :{i}")
    print(f"accuracy            :{acc:.2f}%")
    print("====================================================")
```



Comment: the greater the number of the estimators the higher the accuracy we got.

Boosting

(a) Use GradientBoosting classifier to classify test set samples.

- First, tune number of estimators parameter by trying 4 values in the interval of [10, 200].
- Then by using the tuned value for number of estimators, tune the learning rate parameter by trying 4 values within the range of [0.1, 0.9].
- Display accuracy and Confusion Matrix separately for the best value of both parameters (Number of estimators and learning rate).

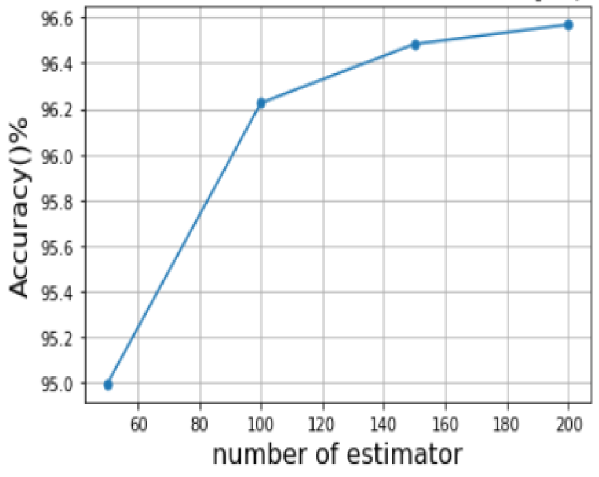
```python
from sklearn.ensemble import GradientBoostingClassifier

def applyBoostingAndReturnAcc(X_train,y_train,X_test,y_test,n=100,lr=0.1):
    estimator = GradientBoostingClassifier(n_estimators=n,learning_rate=lr,random_state=2022)
    estimator.fit(X_train, y_train)
    acc = getAccuracy(estimator, X_test, y_test)
    return acc
```

First, tune number of estimators parameter by trying 4 values in the interval of [10, 200].

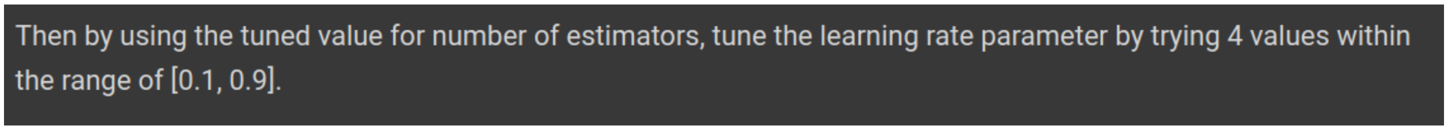
```
number of estimator :50
accuracy            :95.00%
=================================
number of estimator :100
accuracy            :96.23%
=================================
number of estimator :150
accuracy            :96.48%
=================================
number of estimator :200
accuracy            :96.57%
=================================
```

```python
num_estimators = [50,100,150,200]
Boosting_accuracies = []
for i in num_estimators:
    acc = applyBoostingAndReturnAcc(X_train,y_train,X_test,y_test,n=i)
    Boosting_accuracies.append(acc)
    print(f"number of estimator :{i}")
    print(f"accuracy            :{acc:.2f}%")
    print("==================================================")
```

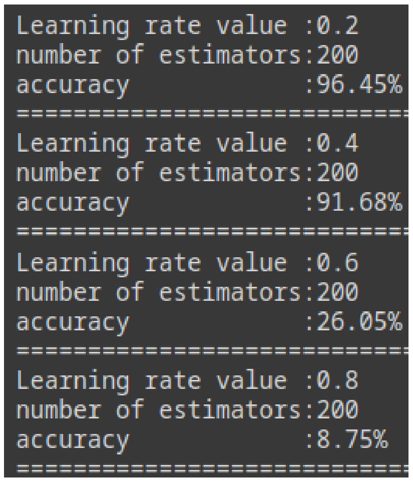
Try 5 different values of number of estimators within the interval of [10, 200] Vs. the Boosting Accuracy



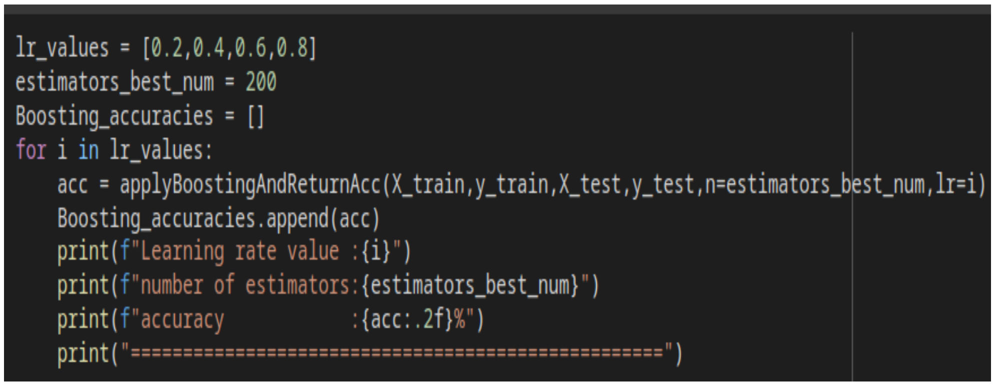
comment: the greater the number of estimators the higher the accuracy



Then by using the tuned value for number of estimators, tune the learning rate parameter by trying 4 values within the range of [0.1, 0.9].

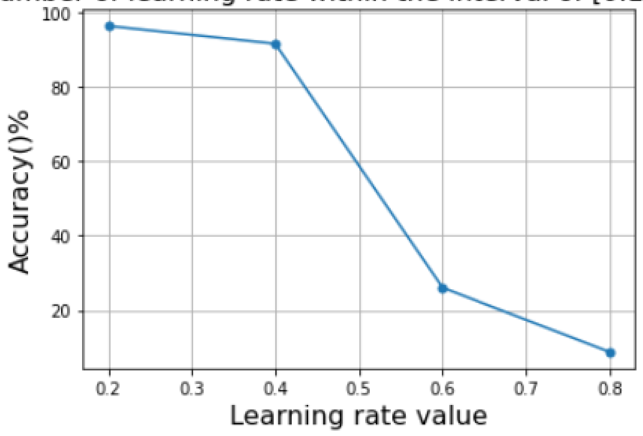


```
Learning rate value :0.2
number of estimators:200
accuracy            :96.45%
=============================
Learning rate value :0.4
number of estimators:200
accuracy            :91.68%
=============================
Learning rate value :0.6
number of estimators:200
accuracy            :26.05%
=============================
Learning rate value :0.8
number of estimators:200
accuracy            :8.75%
=============================
```



```python
lr_values = [0.2,0.4,0.6,0.8]
estimators_best_num = 200
Boosting_accuracies = []
for i in lr_values:
    acc = applyBoostingAndReturnAcc(X_train,y_train,X_test,y_test,n=estimators_best_num,lr=i)
    Boosting_accuracies.append(acc)
    print(f"Learning rate value :{i}")
    print(f"number of estimators:{estimators_best_num}")
    print(f"accuracy            :{acc:.2f}%")
    print("================================================")
```

Try 4 different values of number of learning rate within the interval of [0.1,0.9] Vs. the Boosting Accuracy

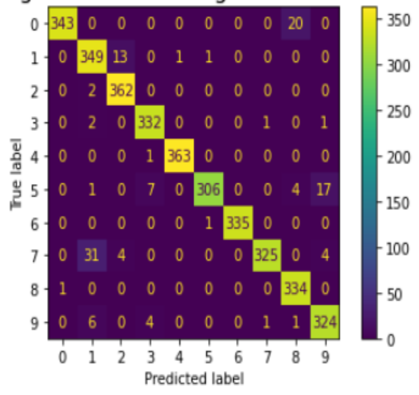


comment: the greater the learning rate the lower the accuracy of the model.

Display accuracy and Confusion Matrix separately for the best value of both parameters (Number of estimators and learning rate).

```
Learning rate value :0.2
number of estimators:200
accuracy            :96.45%
===================================================
========================================================================
Accuracy of Boosting strategy using GradientBoostingClassifier with 200 estimators and 0.2 Learning rate: 96.45%
========================================================================
confusion matrix of applying Boosting strategy using GradientBoostingClassifier with 200 estimators and 0.2 Learning rate on test data
[[343   0   0   0   0   0   0   0  20   0]
 [  0 349  13   0   1   1   0   0   0   0]
 [  0   2 362   0   0   0   0   0   0   0]
 [  0   2   0 332   0   0   0   1   0   1]
 [  0   0   0   1 363   0   0   0   0   0]
 [  0   1   0   7   0 306   0   0   4  17]
 [  0   0   0   0   0   1 335   0   0   0]
 [  0  31   4   0   0   0   0 325   0   4]
 [  1   0   0   0   0   0   0   0 334   0]
 [  0   6   0   4   0   0   0   1   1 324]]
========================================================================
========================================================================
```

confusion matrix of applying Boosting strategy using GradientBoostingClassifier with 200 estimators and 0.2 Learning rate on test data



```
========================================================================
Classification Report of : Boosting strategy using GradientBoostingClassifier with 200 estimators and 0.2 Learning rate
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.89      0.96      0.92       364
           2       0.96      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      1.00      1.00       364
           5       0.99      0.91      0.95       335
           6       1.00      1.00      1.00       336
           7       0.99      0.89      0.94       364
           8       0.93      1.00      0.96       335
           9       0.94      0.96      0.95       336

    accuracy                           0.96      3497
   macro avg       0.97      0.96      0.96      3497
weighted avg       0.97      0.96      0.96      3497
```

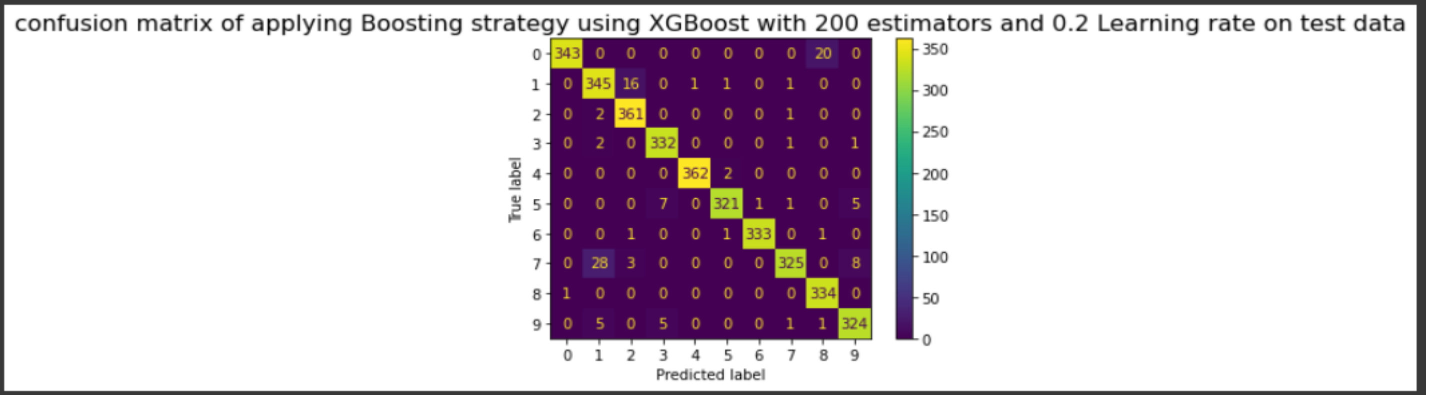
(b) Build XGBoost classifier with the same parameters that you obtained in question (4-a). Provide accuracy and Confusion Matrix.

```
# !pip install xgboost
import xgboost as xgb
```

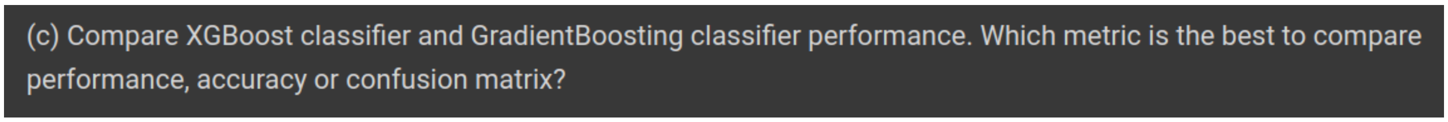
```
xgb_model = xgb.XGBClassifier(learning_rate=0.2,n_estimators=200, random_state=2022)
xgb_model.fit(X_train.values, y_train.values)
getAccuracyAndConfusionMatrix(xgb_model,X_test.values,y_test.values,f'Boosting strategy u
```

```
========================================================================
Accuracy of Boosting strategy using XGBoost with 200 estimators and 0.2 Learning rate: 96.65%
========================================================================
confusion matrix of applying Boosting strategy using XGBoost with 200 estimators and 0.2 Learning rate on test data
[[343   0   0   0   0   0   0   0  20   0]
 [  0 345  16   0   1   1   0   1   0   0]
 [  0   2 361   0   0   0   0   1   0   0]
 [  0   2   0 332   0   0   0   1   0   1]
 [  0   0   0   0 362   2   0   0   0   0]
 [  0   0   0   7   0 321   1   1   0   5]
 [  0   0   1   0   0   1 333   0   1   0]
 [  0  28   3   0   0   0   0 325   0   8]
 [  1   0   0   0   0   0   0   0 334   0]
 [  0   5   0   5   0   0   0   1   1 324]]
========================================================================
```

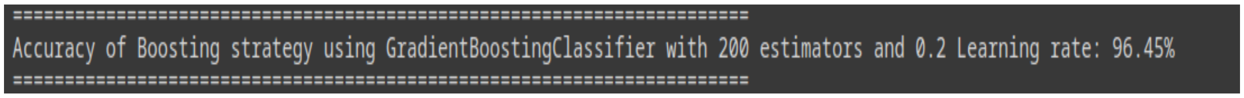
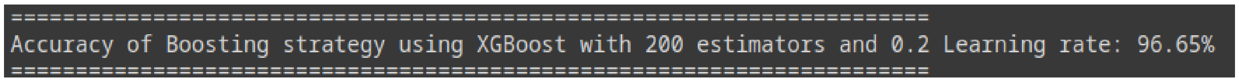
```
========================================================================
```

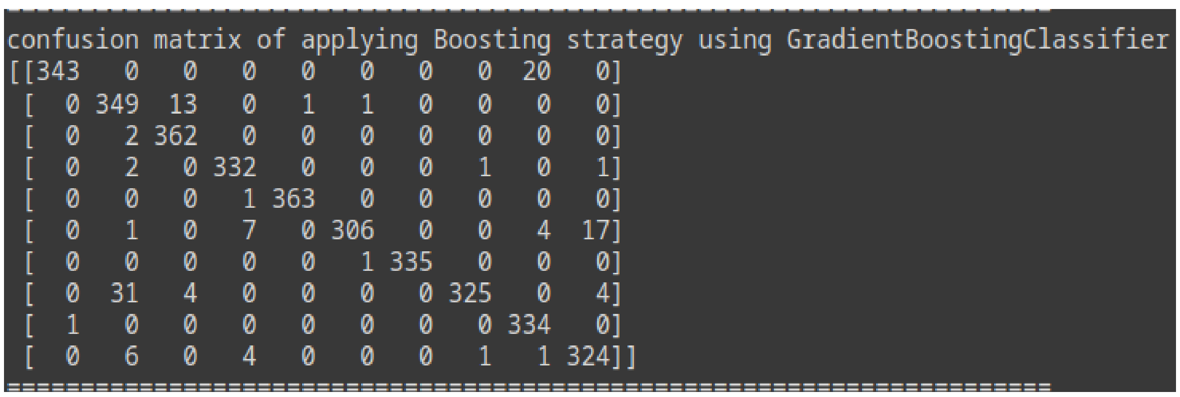
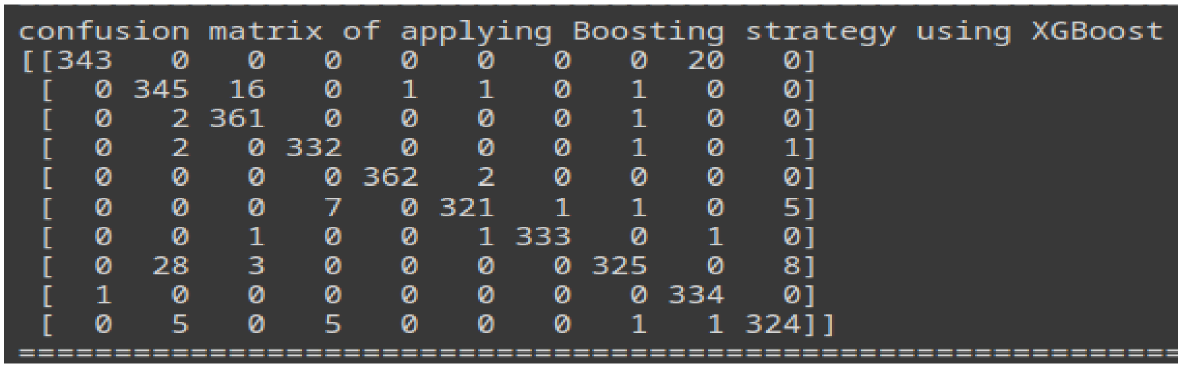


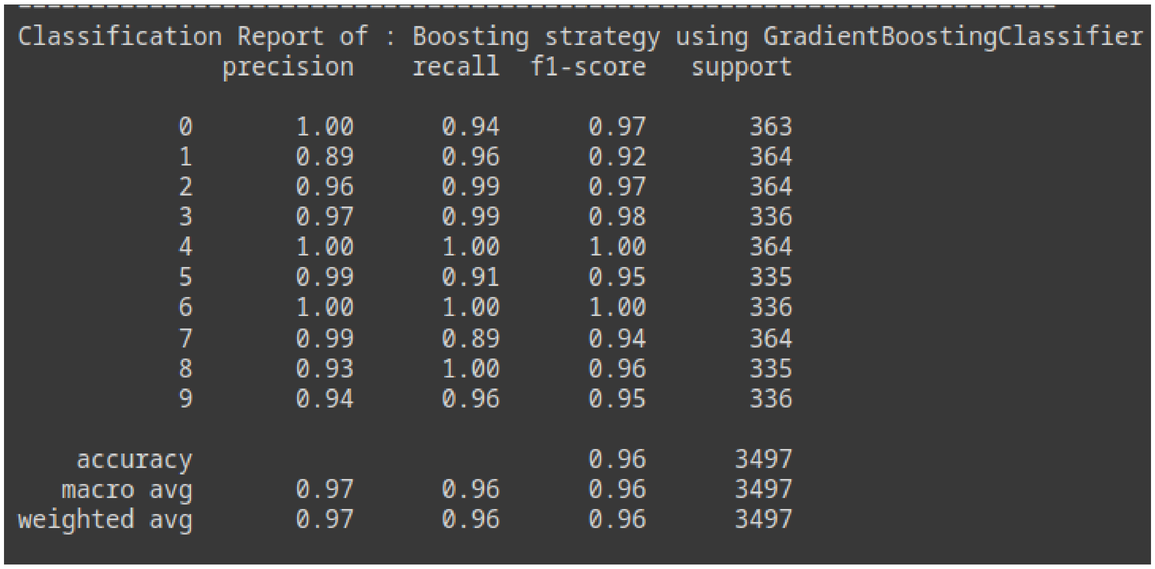
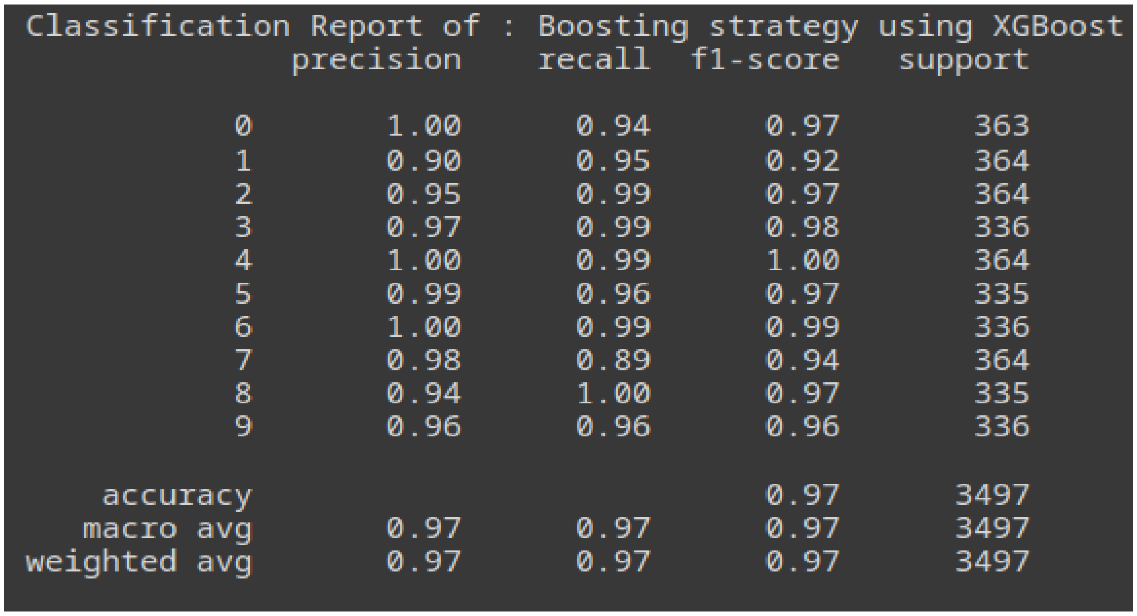
```
========================================================================
Classification Report of : Boosting strategy using XGBoost with 200 estimators and 0.2 Learning rate
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.90      0.95      0.92       364
           2       0.95      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      0.99      1.00       364
           5       0.99      0.96      0.97       335
           6       1.00      0.99      0.99       336
           7       0.98      0.89      0.94       364
           8       0.94      1.00      0.97       335
           9       0.96      0.96      0.96       336

    accuracy                           0.97      3497
   macro avg       0.97      0.97      0.97      3497
weighted avg       0.97      0.97      0.97      3497
```



(c) Compare XGBoost classifier and GradientBoosting classifier performance. Which metric is the best to compare performance, accuracy or confusion matrix?

	Accuracy
GradientBoosting Classifier	 == Accuracy of Boosting strategy using GradientBoostingClassifier with 200 estimators and 0.2 Learning rate: 96.45% ==
XGBoost Classifier	 == Accuracy of Boosting strategy using XGBoost with 200 estimators and 0.2 Learning rate: 96.65% ==

	Confusion Matrix
GradientBoosting Classifier	 confusion matrix of applying Boosting strategy using GradientBoostingClassifier [[343 0 0 0 0 0 0 0 20 0] [0 349 13 0 1 1 0 0 0 0] [0 2 362 0 0 0 0 0 0 0] [0 2 0 332 0 0 0 1 0 1] [0 0 0 1 363 0 0 0 0 0] [0 1 0 7 0 306 0 0 4 17] [0 0 0 0 0 1 335 0 0 0] [0 31 4 0 0 0 0 325 0 4] [1 0 0 0 0 0 0 0 334 0] [0 6 0 4 0 0 0 1 1 324]] ==
XGBoost Classifier	 confusion matrix of applying Boosting strategy using XGBoost [[343 0 0 0 0 0 0 0 20 0] [0 345 16 0 1 1 0 1 0 0] [0 2 361 0 0 0 0 1 0 0] [0 2 0 332 0 0 0 1 0 1] [0 0 0 0 362 2 0 0 0 0] [0 0 0 7 0 321 1 1 0 5] [0 0 1 0 0 1 333 0 1 0] [0 28 3 0 0 0 0 325 0 8] [1 0 0 0 0 0 0 0 334 0] [0 5 0 5 0 0 0 1 1 324]] ==

	Classification Report
GradientBoosting Classifier	
XGBoost Classifier	

```
--------------------------------------------------------------------
Classification Report of : Boosting strategy using GradientBoostingClassifier
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.89      0.96      0.92       364
           2       0.96      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      1.00      1.00       364
           5       0.99      0.91      0.95       335
           6       1.00      1.00      1.00       336
           7       0.99      0.89      0.94       364
           8       0.93      1.00      0.96       335
           9       0.94      0.96      0.95       336

    accuracy                           0.96      3497
   macro avg       0.97      0.96      0.96      3497
weighted avg       0.97      0.96      0.96      3497
```

```
Classification Report of : Boosting strategy using XGBoost
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.90      0.95      0.92       364
           2       0.95      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      0.99      1.00       364
           5       0.99      0.96      0.97       335
           6       1.00      0.99      0.99       336
           7       0.98      0.89      0.94       364
           8       0.94      1.00      0.97       335
           9       0.96      0.96      0.96       336

    accuracy                           0.97      3497
   macro avg       0.97      0.97      0.97      3497
weighted avg       0.97      0.97      0.97      3497
```

Comparison Between the Two Models

Regarding the accuracy as a performance measure

- with the same parameter **(n_estimators =200, LR=0.2)**the accuracy of xgBoost **(96.65%)**is a bit better than the accuracy of the Gradient Boosting Classifier**(96.45%).**
- **So by using the accuracy as a comparison measure we only understood which predicted better but we can't understand the error of the model. so we couldn’t handle which class the model couldn’t predict correctly.**

Regarding the confusion Matrix as a performance measure :

Confusion matrices can help with side-by-side comparisons of different classification methods: Precision, Recall, Accuracy, and F1 Score(The closer to 1, the better the model**)**.

regarding our dataset, we are dealing with a multiclass classification problem so, we will use the macro average or weighted average to compare the two models in terms of F-score.

The macro-averaged F1 score (or macro F1 score) is computed using the arithmetic mean (aka **unweighted** mean) of all the per-class F1 scores.

The weighted-average F1 score is calculated by taking the mean of all per-class F1 scores while considering each class's support.

So regarding the F-score weighted average

Gradient Boosting: 0.96

XGBoost classifier: 0.97

so XGBoost predicts a bit better than the Gradient Boosting in our case because it is closer to 1.

So regarding the recall weighted average

Gradient Boosting: 0.96

XGBoost classifiers: 0.97

so XGBoost have a higher recall than Gradient Boosting however both of them have the same parameter.

So regarding the precision weighted average

Gradient Boosting: 0.97

XGBoost classifiers: 0.97

both XGBoost and Gradient Boosting have the same value.

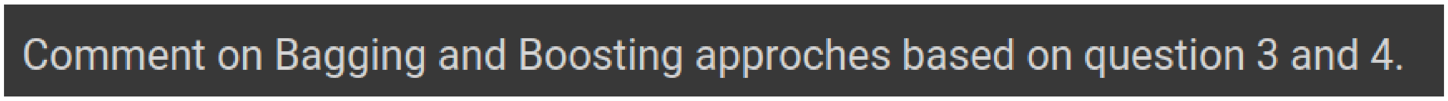
So regarding the Accuracy

Gradient Boosting: 0.96

XGBoost classifiers: 0.97

so in terms of accuracy XGBoost is a bit better.

So Confusion matrix is better as a performance measure because it provided us with more knowledge about the results than the Accuracy as a performance measure.


Comment on Bagging and Boosting approches based on question 3 and 4.

1. Bagging is a homogeneous weak learners' model that learns from each other independently in parallel and combines them for determining the model average.
2. Boosting is also a homogeneous weak learners' model but works differently from Bagging. In this model, learners learn sequentially and adaptively to improve model predictions of a learning algorithm.

	Decision tree **(n_estimator = 10,LR=0.1)**	**SVM** **(n_estimator = 10,LR=0.1)**
Bagging accuracy	**94.71 %**	**98.11 %**

	Gradiant Boosting **(n_estimator = 200,LR=0.2)**	**xgBoost** **(n_estimator = 200,LR=0.2)**
Boosting accuracy	**96.45%**	**96.65%**

So regarding to our experiment in the assignment the best bagging model with SVM of estimator = 10 and learning rate = 0.1 with accuracy 98.11 %

and the best boosting model was the xgboost model (n_estimator = 200,LR=0.2) with accuracy 96.65%

so in general the best model in our problem is by applying Bagging strategy with SVM as a base estimator with number of estimator = 10 and learning rate = 0.1 .

References:

[1] https://www.statology.org/sklearn-classification-report/

[2] https://scikit-learn.org/stable/modules/tree.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[5] https://xgboost.readthedocs.io/en/latest/index.html

[6] https://xgboost.readthedocs.io/en/stable/parameter.html

[7] https://www.analyticssteps.com/blogs/what-gini-index-and-information-gain-decision-trees

[8] https://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

[9] https://www.upgrad.com/blog/bagging-vs-boosting/