uOttawa

ELG 5255:
Applied Machine Learning
Assignment 4

Dr. Murat Simsek

GROUP 14

Table of Contents

Table of Figures

Table of Tables

Part 1:

Question (1):

a)

Table 1 Data

Weather (F1)	Temperature (F2)	Humidity (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

- First, Gini split will be calculated for the four features to decide which feature will be used at the root node.

$$\boldsymbol{Gini_{split}(Weather)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Cloudy}}{n} Gini(Cloudy) + \frac{n_{Sunny}}{n} Gini(Sunny) + \frac{n_{Rainy}}{n} Gini(Rainy)$$
$$= \frac{3}{10}(1 - P(Yes|Cloudy)^2 - P(No|Cloudy)^2) + \frac{4}{10}(1 - P(Yes|Sunny)^2 - P(No|Sunny)^2)$$
$$+ \frac{3}{10}(1 - P(Yes|Rainy)^2 - P(No|Rainy)^2)$$
$$= \frac{3}{10}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) + \frac{4}{10}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) + \frac{3}{10}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = \mathbf{0.4167}$$

$$\boldsymbol{Gini_{split}(Temperature)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Cool}}{n} Gini(Cool) + \frac{n_{Hot}}{n} Gini(Hot) + \frac{n_{Mild}}{n} Gini(Mild)$$
$$= \frac{3}{10}(1 - P(Yes|Cool)^2 - P(No|Cool)^2) + \frac{3}{10}(1 - P(Yes|Hot)^2 - P(No|Hot)^2)$$
$$+ \frac{4}{10}(1 - P(Yes|Mild)^2 - P(No|Mild)^2)$$
$$= \frac{3}{10}\left(1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2\right) + \frac{3}{10}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) + \frac{4}{10}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = \mathbf{0.3333}$$

$$\boldsymbol{Gini_{split}(Humidity)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Normal}}{n} Gini(Normal) + \frac{n_{High}}{n} Gini(High)$$
$$= \frac{4}{10}(1 - P(Yes|Normal)^2 - P(No|Normal)^2) + \frac{6}{10}(1 - P(Yes|High)^2 - P(No|High)^2)$$
$$= \frac{4}{10}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) + \frac{6}{10}\left(1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2\right) = \mathbf{0.4167}$$

$$\boldsymbol{Gini_{split}(Wind)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Weak}}{n} Gini(Weak) + \frac{n_{Strong}}{n} Gini(Strong)$$
$$= \frac{4}{10}(1 - P(Yes|Weak)^2 - P(No|Weak)^2) + \frac{6}{10}(1 - P(Yes|Strong)^2 - P(No|Strong)^2)$$
$$= \frac{4}{10}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) + \frac{6}{10}\left(1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2\right) = \mathbf{0.367}$$

Table 2 Gini split results for different features

Feature	Gini
Weather	0.4167
Temperature	0.3333 (Chosen to be used at the root node)
Humidity	0.417
Wind	0.367

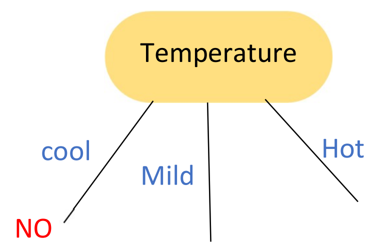

Figure 1 Decision tree after first step

- Second, Gini split will be calculated for the remaining three features to decide which feature will be used at the intermediate nodes in both Hot and Mild branches.

Table 3 The data that will be used to calculate the Gini split for the Hot branch

Weather (F1)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Sunny	High	Weak	Yes
Sunny	High	Strong	No
Sunny	High	Strong	No

$$\boldsymbol{Gini_{split}(Weather)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Sunny}}{n} Gini(Sunny) = \frac{3}{3}(1 - P(Yes|Sunny)^2 - P(No|Sunny)^2)$$
$$= \left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = \mathbf{0.444}$$

$$\boldsymbol{Gini_{split}(Humidity)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{High}}{n} Gini(High) = \frac{3}{3}(1 - P(Yes|High)^2 - P(No|High)^2)$$
$$= \left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = \mathbf{0.444}$$

$$\boldsymbol{Gini_{split}(Wind)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Weak}}{n} Gini(Weak) + \frac{n_{Strong}}{n} Gini(Strong)$$
$$= \frac{1}{3}(1 - P(Yes|Weak)^2 - P(No|Weak)^2) + \frac{2}{3}(1 - P(Yes|Strong)^2 - P(No|Strong)^2)$$
$$= \frac{1}{3}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) + \frac{2}{3}\left(1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2\right) = \mathbf{0}$$

Table 4 Gini split results for different features for the Hot branch

Feature	Gini
Weather	0.444
Humidity	0.444
Wind	0 (Chosen to be used at the intermediate node)

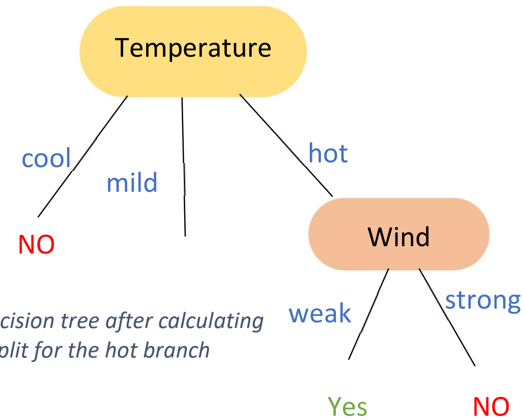

Figure 2 Decision tree after calculating Gini split for the hot branch

Table 5 The data that will be used to calculate the Gini split for the Mild branch

Weather (F1)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Rainy	Normal	Strong	Yes
Cloudy	High	Strong	No
Sunny	High	Strong	No
Cloudy	High	Weak	Yes

$$\boldsymbol{Gini_{split}(Weather)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Cloudy}}{n} Gini(Cloudy) + \frac{n_{Sunny}}{n} Gini(Sunny) + \frac{n_{Rainy}}{n} Gini(Rainy)$$
$$= \frac{2}{4}(1 - P(Yes|Cloudy)^2 - P(No|Cloudy)^2) + \frac{1}{4}(1 - P(Yes|Sunny)^2 - P(No|Sunny)^2)$$
$$+ \frac{1}{4}(1 - P(Yes|Rainy)^2 - P(No|Rainy)^2)$$
$$= \frac{2}{4}\left(1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right) + \frac{1}{4}\left(1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2\right) + \frac{1}{4}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) = \mathbf{0.25}$$

$$\boldsymbol{Gini_{split}(Humidity)} = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Normal}}{n} Gini(Normal) + \frac{n_{High}}{n} Gini(High)$$
$$= \frac{1}{4}(1 - P(Yes|Normal)^2 - P(No|Normal)^2) + \frac{3}{4}(1 - P(Yes|High)^2 - P(No|High)^2)$$
$$= \frac{1}{4}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) + \frac{3}{4}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = \mathbf{0.333}$$

$$Gini_{split}(Wind) = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Weak}}{n} Gini(Weak) + \frac{n_{Strong}}{n} Gini(Strong)$$

$$= \frac{1}{4}(1 - P(Yes|Weak)^2 - P(No|Weak)^2) + \frac{3}{4}(1 - P(Yes|Strong)^2 - P(No|Strong)^2)$$

$$= \frac{1}{4}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) + \frac{3}{4}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = \mathbf{0.333}$$

Table 6 Gini split results for different features for the Mild branch

Feature	Gini
Weather	0.25 (Chosen to be used at the intermediate node)
Humidity	0.333
Wind	0.333

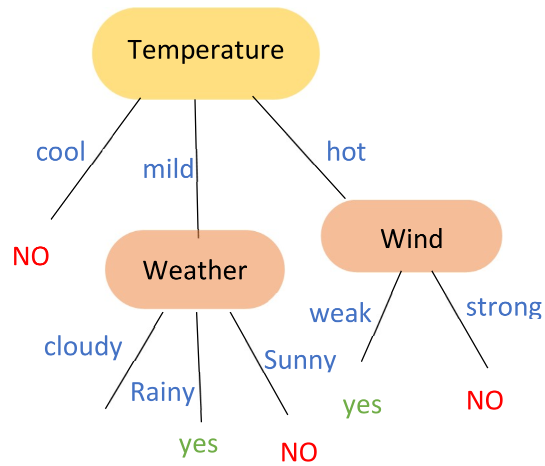

Figure 3 Decision tree after second step

- Finally, Gini split will be calculated for the remaining two features to decide which feature will be used at the intermediate node in the Cloudy branch.

Table 7 The data that will be used to calculate the Gini split for the Cloudy branch

Humidity (F3)	Wind (F4)	Hiking (Labels)
High	Strong	No
High	Weak	Yes

$$Gini_{split}(Humidity) = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{High}}{n} Gini(High) = \frac{2}{2}(1 - P(Yes|High)^2 - P(No|High)^2)$$

$$= \left(1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right) = \mathbf{0.5}$$

$$Gini_{split}(Wind) = \sum_{i=1}^{N_c} \frac{n_i}{n} Gini(i) = \frac{n_{Weak}}{n} Gini(Weak) + \frac{n_{Strong}}{n} Gini(Strong)$$

$$= \frac{1}{2}(1 - P(Yes|Weak)^2 - P(No|Weak)^2) + \frac{1}{2}(1 - P(Yes|Strong)^2 - P(No|Strong)^2)$$

$$= \frac{1}{2}\left(1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2\right) + \frac{1}{2}\left(1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2\right) = \mathbf{0}$$

Table 8 Gini split results for different features for the Cloudy branch

Feature	Gini
Humidity	0.5
wind	0 (Chosen to be used at the intermediate node)

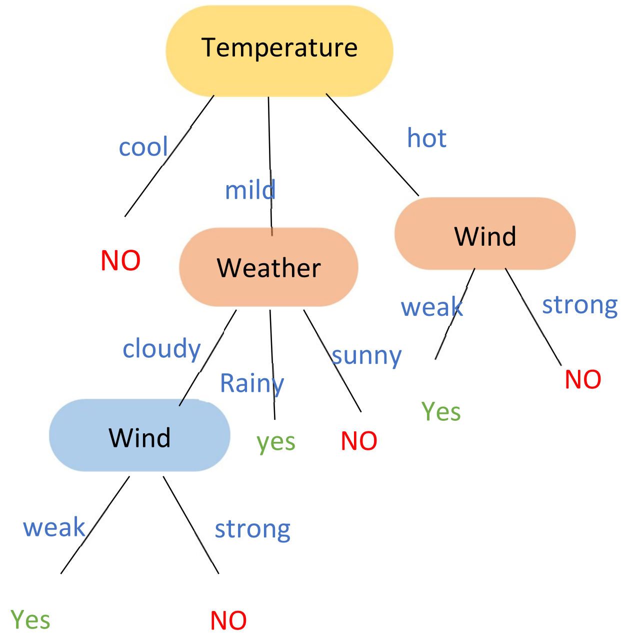

Figure 4 The final shape of decision tree

b)

Table 9 Data

Weather (F1)	Temperature (F2)	Humidity (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

- First, Information Gain will be calculated for the four features to decide which feature will be used at the root node.

$$
\begin{aligned}
\boldsymbol{Gain_{split}(S, Weather)} &= Entropy(S) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i) \\
&= Entropy(S) - \frac{n_{Cloudy}}{n} Entropy(Cloudy) - \frac{n_{Sunny}}{n} Entropy(Sunny) \\
&- \frac{n_{Rainy}}{n} Entropy(Rainy) \\
&= (-P(Yes)\log_2 P(Yes) - P(No)\log_2 P(No)) \\
&- \frac{3}{10}(-P(Yes|Cloudy)\log_2 P(Yes|Cloudy) - P(No|Cloudy)\log_2 P(No|Cloudy)) \\
&- \frac{4}{10}(-P(Yes|Sunny)\log_2 P(Yes|Sunny) - P(No|Sunny)\log_2 P(No|Sunny)) \\
&- \frac{3}{10}(-P(Yes|Rainy)\log_2 P(Yes|Rainy) - P(No|Rainy)\log_2 P(No|Rainy)) \\
&= \left(-\left(\frac{3}{10}\right)\log_2\left(\frac{3}{10}\right) - \left(\frac{7}{10}\right)\log_2\left(\frac{7}{10}\right)\right) - \frac{3}{10}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) \\
&- \frac{4}{10}\left(-\left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right)\right) - \frac{3}{10}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) \\
&= 0.8812 - 0.2754 - 0.3245 - 0.2754 = \mathbf{0.0059}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{Gain_{split}(S, Temperature)} &= Entropy(S) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i) \\
&= Entropy(S) - \frac{n_{Cool}}{n} Entropy(Cool) - \frac{n_{Hot}}{n} Entropy(Hot) - \frac{n_{Mild}}{n} Entropy(Mild) \\
&= 0.8812 - \frac{3}{10}(-P(Yes|Cool)\log_2 P(Yes|Cool) - P(No|Cool)\log_2 P(No|Cool)) \\
&- \frac{3}{10}(-P(Yes|Hot)\log_2 P(Yes|Hot) - P(No|Hot)\log_2 P(No|Hot)) \\
&- \frac{4}{10}(-P(Yes|Mild)\log_2 P(Yes|Mild) - P(No|Mild)\log_2 P(No|Mild)) \\
&= 0.8812 - \frac{3}{10}\left(-\left(\frac{0}{3}\right)\log_2\left(\frac{0}{3}\right) - \left(\frac{3}{3}\right)\log_2\left(\frac{3}{3}\right)\right) - \frac{3}{10}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) \\
&- \frac{4}{10}\left(-\left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right)\right) = 0.8812 - 0 - 0.2754 - 0.4 = \mathbf{0.2058}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{Gain_{split}(S, Humidity)} &= Entropy(S) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i) \\
&= Entropy(S) - \frac{n_{Normal}}{n} Entropy(Normal) - \frac{n_{High}}{n} Entropy(High) \\
&= 0.8812 \\
&- \frac{4}{10}(-P(Yes|Normal)\log_2 P(Yes|Normal) - P(No|Normal)\log_2 P(No|Normal)) \\
&- \frac{6}{10}(-P(Yes|High)\log_2 P(Yes|High) - P(No|High)\log_2 P(No|High)) \\
&= 0.8812 - \frac{4}{10}\left(-\left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right)\right) - \frac{6}{10}\left(-\left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right) - \left(\frac{4}{6}\right)\log_2\left(\frac{4}{6}\right)\right) \\
&= 0.8812 - 0.3245 - 0.5509 = \mathbf{0.0058}
\end{aligned}
$$

$$\boldsymbol{Gain_{split}(S, Wind)} = Entropy(S) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(S) - \frac{n_{Weak}}{n} Entropy(Weak) - \frac{n_{Strong}}{n} Entropy(Strong)$$

$$= 0.8812 - \frac{4}{10}(-P(Yes|Weak)\log_2 P(Yes|Weak) - P(No|Weak)\log_2 P(No|Weak))$$

$$- \frac{6}{10}(-P(Yes|Strong)\log_2 P(Yes|Strong) - P(No|Strong)\log_2 P(No|Strong))$$

$$= 0.8812 - \frac{4}{10}\left(-\left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right)\right) - \frac{6}{10}\left(-\left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right)\right)$$

$$= 0.8812 - 0.4 - 0.39 = \mathbf{0.0912}$$

Table 10 Information gain results for different features

Feature	Information Gain
Weather	0. 0059
Temperature	0.2058 (Chosen to be used at the root node)
Humidity	0.0058
Wind	0.0912

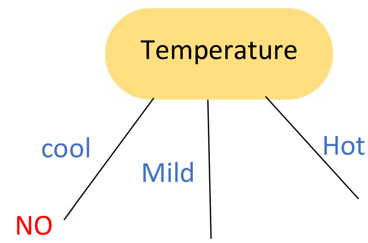

Figure 5 Decision tree after first step

- Second, Information gain will be calculated for the remaining three features to decide which feature will be used at the intermediate nodes in both Hot and Mild branches.

Table 11 The data that will be used to calculate the Information gain for the Hot branch

Weather (F1)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Sunny	High	Weak	Yes
Sunny	High	Strong	No
Sunny	High	Strong	No

$$\boldsymbol{Gain_{split}(Hot, Weather)} = Entropy(Hot) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i) = Entropy(Hot) - \frac{n_{Sunny}}{n} Entropy(Sunny)$$

$$= (-P(Yes|Hot)\log_2 P(Yes|Hot) - P(No|Hot)\log_2 P(No|Hot))$$

$$- \frac{3}{3}(-P(Yes|Sunny)\log_2 P(Yes|Sunny) - P(No|Sunny)\log_2 P(No|Sunny))$$

$$= \left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) - \frac{3}{3}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) = 0.918 - 0.918 = \mathbf{0}$$

$$\boldsymbol{Gain_{split}(Hot, Humidity)} = Entropy(Hot) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i) = 0.918 - \frac{n_{High}}{n} Entropy(High)$$
$$= 0.918 - \frac{3}{3}(-P(Yes|High)\log_2 P(Yes|High) - P(No|High)\log_2 P(No|High))$$
$$= 0.918 - \frac{3}{3}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right) = 0.918 - 0.918 = \mathbf{0}$$

$$\boldsymbol{Gain_{split}(Hot, Wind)} = Entropy(Hot) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$
$$= 0.918 - \frac{n_{Weak}}{n} Entropy(Weak) - \frac{n_{Strong}}{n} Entropy(Strong)$$
$$= 0.918 - \frac{1}{3}(-P(Yes|Weak)\log_2 P(Yes|Weak) - P(No|Weak)\log_2 P(No|Weak))$$
$$- \frac{2}{3}(-P(Yes|Strong)\log_2 P(Yes|Strong) - P(No|Strong)\log_2 P(No|Strong))$$
$$= 0.918 - \frac{1}{3}\left(-\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right)\right) - \frac{2}{3}\left(-\left(\frac{0}{2}\right)\log_2\left(\frac{0}{2}\right) - \left(\frac{2}{2}\right)\log_2\left(\frac{2}{2}\right)\right)$$
$$= 0.918 - 0 - 0 = \mathbf{0.918}$$

Table 12 Information gain results for different features for the Hot branch

Feature	Information Gain
Weather	0
Humidity	0
Wind	0.918 (Chosen to be used at the intermediate node)

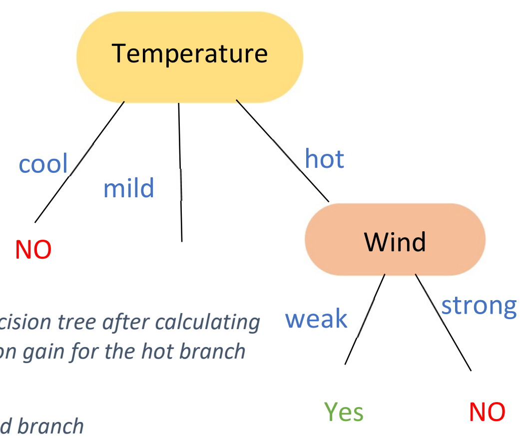

Figure 6 Decision tree after calculating Information gain for the hot branch

Table 13 The data that will be used to calculate the Information gain for the Mild branch

Weather (F1)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Rainy	Normal	Strong	Yes
Cloudy	High	Strong	No
Sunny	High	Strong	No
Cloudy	High	Weak	Yes

$$\boldsymbol{Gain_{split}(Mild, Weather)} = Entropy(Mild) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(Mild) - \frac{n_{Cloudy}}{n} Entropy(Cloudy) - \frac{n_{Sunny}}{n} Entropy(Sunny)$$
$$- \frac{n_{Rainy}}{n} Entropy(Rainy)$$
$$= (-P(Yes|Mild)\log_2 P(Yes|Mild) - P(No|Mild)\log_2 P(No|Mild))$$
$$- \frac{2}{4}(-P(Yes|Cloudy)\log_2 P(Yes|Cloudy) - P(No|Cloudy)\log_2 P(No|Cloudy))$$
$$- \frac{1}{4}(-P(Yes|Sunny)\log_2 P(Yes|Sunny) - P(No|Sunny)\log_2 P(No|Sunny))$$
$$- \frac{1}{4}(-P(Yes|Rainy)\log_2 P(Yes|Rainy) - P(No|Rainy)\log_2 P(No|Rainy))$$
$$= \left(-\left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right)\right) - \frac{2}{4}\left(-\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right)\right)$$
$$- \frac{1}{4}\left(-\left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right) - \left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right)\right) - \frac{1}{4}\left(-\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right)\right) = 1 - 0.5 - 0 - 0$$
$$= \mathbf{0.5}$$

$$\boldsymbol{Gain_{split}(Mild, Humidity)} = Entropy(Mild) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(Mild) - \frac{n_{Normal}}{n} Entropy(Normal) - \frac{n_{High}}{n} Entropy(High)$$
$$= 1 - \frac{1}{4}(-P(Yes|Normal)\log_2 P(Yes|Normal) - P(No|Normal)\log_2 P(No|Normal))$$
$$- \frac{3}{4}(-P(Yes|High)\log_2 P(Yes|High) - P(No|High)\log_2 P(No|High))$$
$$= 1 - \frac{1}{4}\left(-\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right)\right) - \frac{3}{4}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right)$$
$$= 1 - 0 - 0.6887 = \mathbf{0.3113}$$

$$\boldsymbol{Gain_{split}(Mild, Wind)} = Entropy(Mild) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(Mild) - \frac{n_{Weak}}{n} Entropy(Weak) - \frac{n_{Strong}}{n} Entropy(Strong)$$
$$= 1 - \frac{1}{4}(-P(Yes|Weak)\log_2 P(Yes|Weak) - P(No|Weak)\log_2 P(No|Weak))$$
$$- \frac{3}{4}(-P(Yes|Strong)\log_2 P(Yes|Strong) - P(No|Strong)\log_2 P(No|Strong))$$
$$= 1 - \frac{1}{4}\left(-\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right)\right) - \frac{3}{4}\left(-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right) - \left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right)\right)$$
$$= 1 - 0 - 0.6887 = \mathbf{0.3113}$$

Table 14 Information gain results for different features for the Mild branch

Feature	Information Gain
Weather	0.5 (Chosen to be used at the intermediate node)
Humidity	0.3113
Wind	0.3113

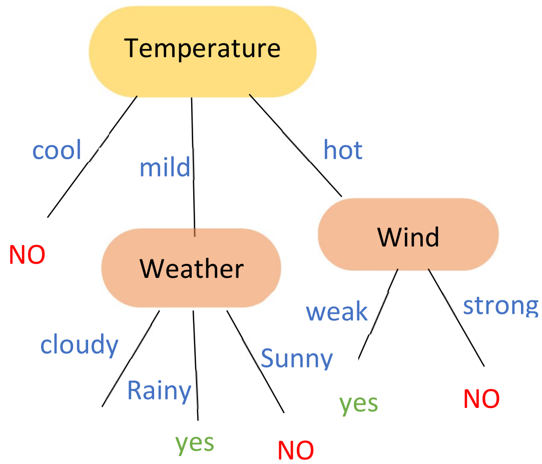

Figure 7 Decision tree after second step

- Finally, Information gain will be calculated for the remaining two features to decide which feature will be used at the intermediate node in the Cloudy branch.

Table 15 The data that will be used to calculate the Information gain for the Cloudy branch

Humidity (F3)	Wind (F4)	Hiking (Labels)
High	Strong	No
High	Weak	Yes

$$\boldsymbol{Gain_{split}(Cloudy, Humidity)} = Entropy(Cloudy) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(Cloudy) - \frac{n_{High}}{n} Entropy(High)$$

$$= (-P(Yes|Cloudy)\log_2 P(Yes|Cloudy) - P(No|Cloudy)\log_2 P(No|Cloudy))$$

$$- \frac{2}{2}(-P(Yes|High)\log_2 P(Yes|High) - P(No|High)\log_2 P(No|High))$$

$$= \left(-\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right)\right) - \frac{2}{2}\left(-\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right)\right) = 1 - 1 = \mathbf{0}$$

$$\boldsymbol{Gain_{split}(Cloudy, Wind)} = Entropy(Cloudy) - \sum_{i=1}^{N_c} \frac{n_i}{n} Entropy(i)$$

$$= Entropy(Cloudy) - \frac{n_{Weak}}{n} Entropy(Weak) - \frac{n_{Strong}}{n} Entropy(Strong)$$

$$= 1 - \frac{1}{2}(-P(Yes|Weak)\log_2 P(Yes|Weak) - P(No|Weak)\log_2 P(No|Weak))$$

$$- \frac{1}{2}(-P(Yes|Strong)\log_2 P(Yes|Strong) - P(No|Strong)\log_2 P(No|Strong))$$

$$= 1 - \frac{1}{2}\left(-\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right)\right) - \frac{1}{2}\left(-\left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right) - \left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right)\right) = 1 - 0 - 0$$

$$= \mathbf{1}$$

Table 16 Information gain results for different features for the Cloudy branch

Feature	Information Gain
Humidity	0
Wind	1 (Chosen to be used at the intermediate node)

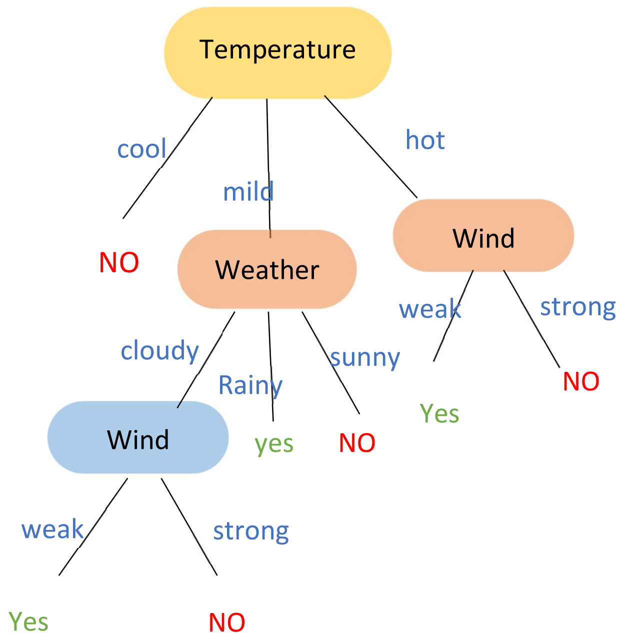

Figure 8 The final shape of decision tree

c)

The two measures (Gini index and information gain) managed to build the same decision tree, the following table summarizes the advantages and disadvantages of both measures:

Table 17 The advantages and disadvantages of Gini index and Information gain

	Gini Index	Information Gain
Advantages	Gini index calculations are faster than information gain calculations because information gain uses logarithms in its calculations [1].	• It can work with continuous and discrete features. • It tends to divide the data into large number of homogeneous groups. • This method chooses the features that has the largest gain to be used in the nodes [3].
Disadvantages	Gini index is greedy approach so the global minima of the solution may not be found. This greedy approach will also cause overfitting [2].	It tends to choose the features that have many distinct values like “customer ID”, as these features will produce pure nodes, but it will cause overfitting as it can only deal with the customers that it dealt with before [3].

Part 2:

Question (2):

```
def eval(model,x_test,y_test,model_name=''):
  print(model_name)
  y_pred=model.predict(x_test)
  acc=accuracy_score(y_test,y_pred)
  print("______________________________________________________________________________________")
  print("The Test accuracy :",acc)
  print("______________________________________________________________________________________")
  fig, ax = plt.subplots(figsize=(20, 10))
  ConfusionMatrixDisplay.from_estimator(model, x_test, y_test,ax=ax)
  plt.show()

  print("______________________________________________________________________________________")
  print(classification_report(y_test,y_pred))
  print("______________________________________________________________________________________")
```

```
DT = DecisionTreeClassifier(random_state=0)
DT.fit(X_train, y_train)
```

```
eval(DT,X_test,y_test,model_name='Decision Tree')
```

	precision	recall	f1-score	support
0	0.94	0.95	0.95	363
1	0.87	0.88	0.88	364
2	0.88	0.95	0.92	364
3	0.89	0.94	0.91	336
4	0.91	0.97	0.94	364
5	0.94	0.84	0.89	335
6	0.95	0.95	0.95	336
7	0.95	0.85	0.90	364
8	0.94	0.94	0.94	336
9	0.95	0.92	0.94	336
accuracy			0.92	3498
macro avg	0.92	0.92	0.92	3498
weighted avg	0.92	0.92	0.92	3498

Figure 9 Decision Tree Classification Report

```
Decision Tree
____________________________________________
The Test accuracy : 0.9208118925100057
```

Figure 10 Decision Tree Test Accuracy

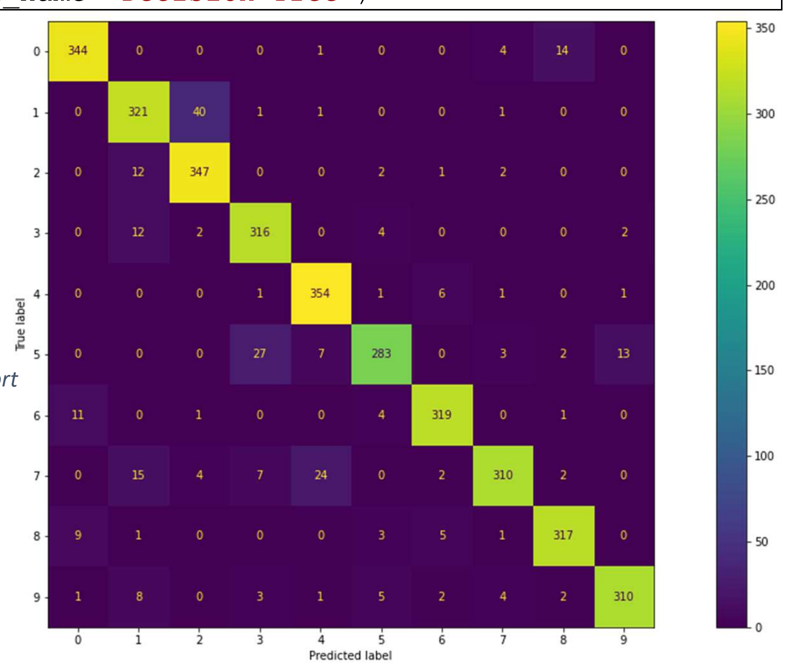

Figure 11 Decision Tree Confusion Matrix

Question (3):

A)

SVM with bagging:

```
estimator_SVM = BaggingClassifier(base_estimator=SVC(),random_state=0)
estimator_SVM.fit(X_train, y_train)
eval(estimator_SVM,X_test,y_test,model_name='SVM with BaggingClassifier')
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	363
1	0.96	0.96	0.96	364
2	0.96	0.99	0.98	364
3	0.99	0.99	0.99	336
4	1.00	0.99	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	1.00	0.98	336
9	0.97	0.98	0.98	336
accuracy			0.98	3498
macro avg	0.98	0.98	0.98	3498
weighted avg	0.98	0.98	0.98	3498

Figure 12 SVM with bagging classification report

```
SVM with BaggingClassifier
________________________________________
The Test accuracy : 0.9817038307604345
```

Figure 13 SVM with bagging test accuracy

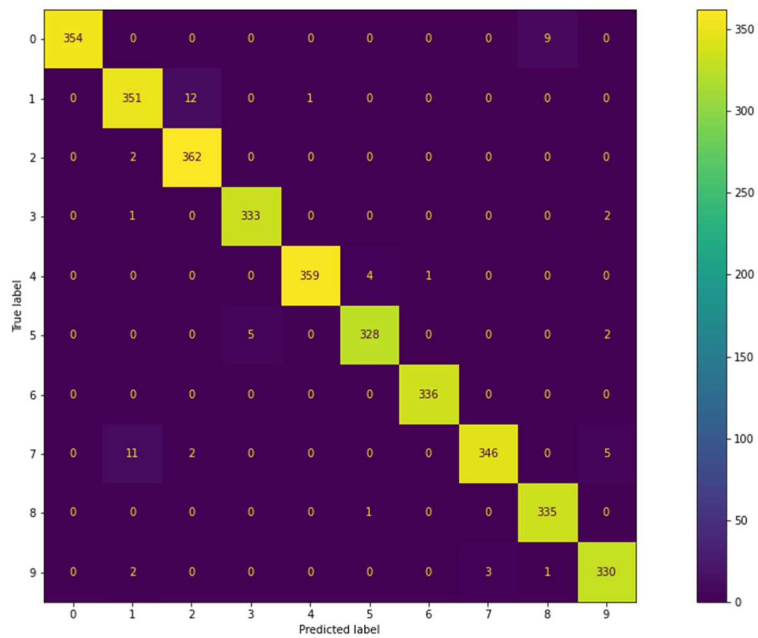

Figure 14 SVM with bagging Confusion Matrix

Decision Tree with bagging

```
estimator_DT = BaggingClassifier(base_estimator=DecisionTreeClassifier()
,random_state=0)
estimator_DT.fit(X_train, y_train)
eval(estimator_DT,X_test,y_test,model_name='Decision Tree with BaggingCl
assifier')
```

```
_____________________________________________________
              precision    recall  f1-score   support

           0       0.98      0.95      0.97       363
           1       0.91      0.93      0.92       364
           2       0.93      0.97      0.95       364
           3       0.95      0.97      0.96       336
           4       0.92      0.98      0.95       364
           5       0.95      0.88      0.91       335
           6       0.99      0.96      0.98       336
           7       0.96      0.90      0.93       364
           8       0.94      0.98      0.96       336
           9       0.95      0.95      0.95       336

    accuracy                           0.95      3498
   macro avg       0.95      0.95      0.95      3498
weighted avg       0.95      0.95      0.95      3498
```

Figure 15 Decision Tree with bagging Classification Report

```
Decision Tree with BaggingClassifier
_____________________________________
The Test accuracy : 0.9479702687249857
```

Figure 16 Decision Tree with bagging test accuracy

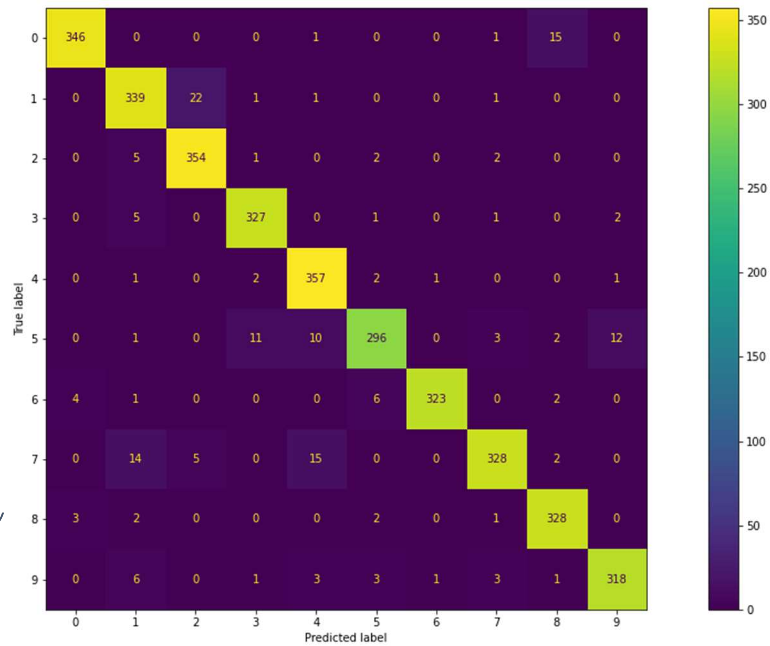

Figure 17 Decision Tree with bagging confusion matrix

B)

```
acc=[]
iter=[10,50,100,150, 200]
for i in iter:
  estimator_DT = BaggingClassifier(base_estimator=DecisionTreeClassifier
(),n_estimators=i,random_state=0)
  estimator_DT.fit(X_train, y_train)
  eval(estimator_DT,X_test,y_test,model_name='Decision Tree with Bagging
 Classifier with number of estimator={}'.format(i))
  y_pred = estimator_DT.predict(X_test)
  acc.append(accuracy_score(y_test,y_pred))
```

Decision Tree with Bagging Classifier with number of estimators =150

	precision	recall	f1-score	support
0	0.99	0.96	0.97	363
1	0.88	0.93	0.90	364
2	0.93	0.98	0.95	364
3	0.97	0.98	0.97	336
4	0.96	0.98	0.97	364
5	0.98	0.91	0.94	335
6	0.99	0.98	0.98	336
7	0.98	0.89	0.93	364
8	0.93	0.98	0.96	336
9	0.95	0.96	0.95	336
accuracy			0.95	3498
macro avg	0.96	0.95	0.95	3498
weighted avg	0.96	0.95	0.95	3498

Figure 18 Decision Tree with Bagging Classifier at 150 estimators' classification report

The Test accuracy : 0.9542595769010863

Figure 19 Decision Tree with Bagging Classifier at 150 estimators test Accuracy

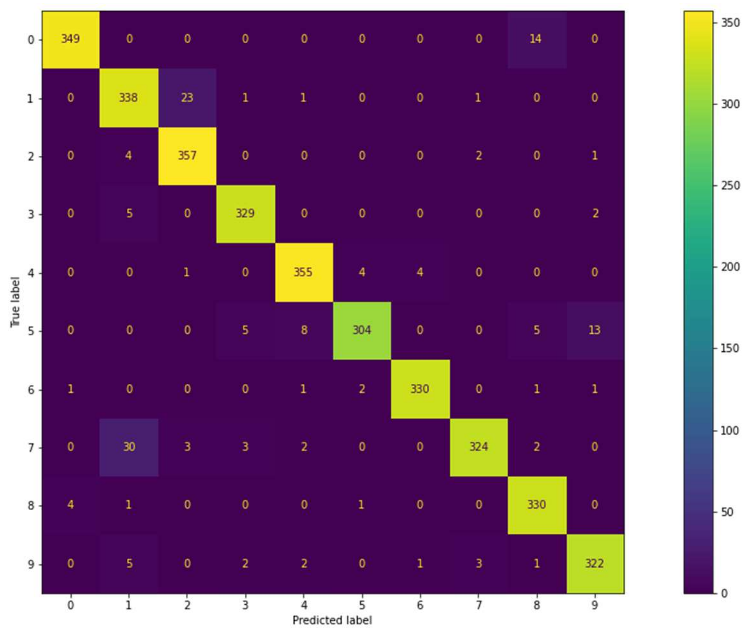

Figure 20 Decision Tree with Bagging Classifier at 150 estimators' confusion matrix

```
plt.figure(figsize=(20, 10))
f=sns.lineplot(x=iter,y=acc)
plt.title('Find the best number of estimators in Decision Tree with Bagg
ing Classifier')
# Set x-axis label
plt.xlabel('number of estimators')
# Set y-axis label
plt.ylabel('Accuracy')

max_acc=max(acc)

maxInd=np.argmax(acc)
print(maxInd)
best_N_estimator=iter[maxInd]
print(best_N_estimator)
bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k")
arrowprops=dict(arrowstyle="->")
kw = dict(xycoords='data',textcoords="axes fraction",
          arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
f.annotate("The best number of estimators", xy=(best_N_estimator, max_ac
c), xytext=(0.84,0.76), **kw)
plt.show()
```

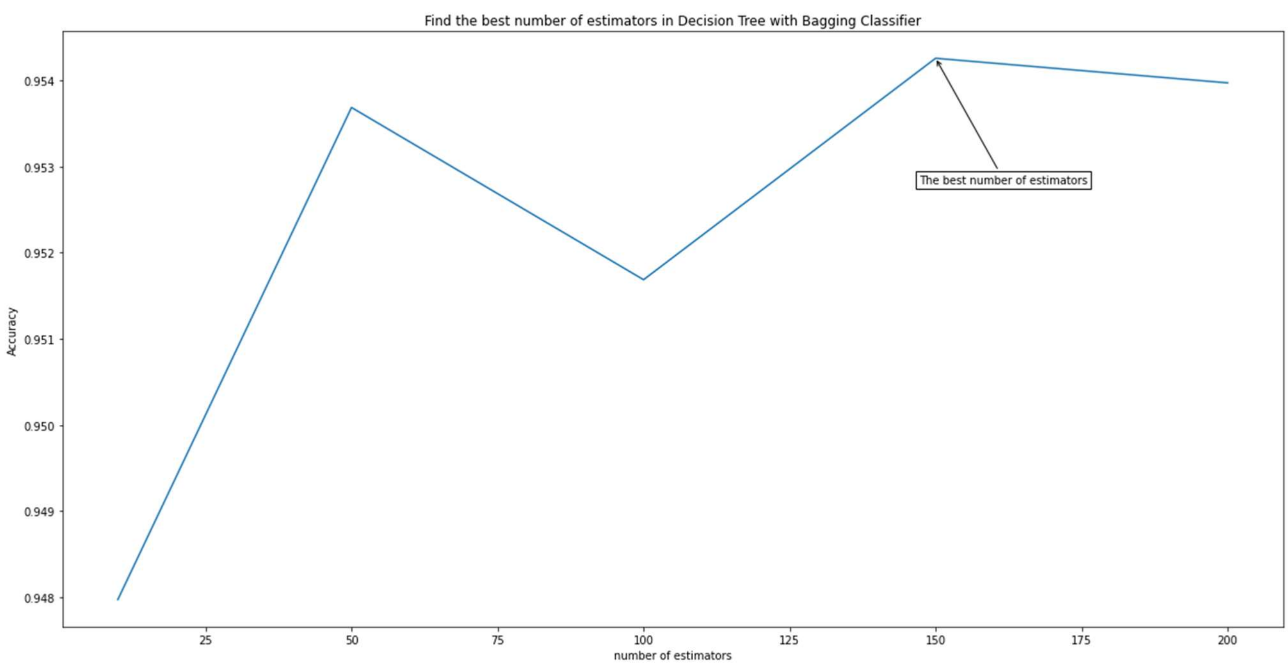

Figure 21 Plot for the best number of estimators in decision tree with bagging classifier

N.B: From the previous accuracies and plot, the best number of estimators is 150.

Question (4):

A)

The best estimator:

```
acc=[]
iters=[50, 100, 150, 200]
for i in iters:
  estimator_B = GradientBoostingClassifier(n_estimators=i, random_state=
0)
  estimator_B.fit(X_train, y_train)
  eval(estimator_B,X_test,y_test,model_name='Gradient Boosting with numb
er of estimator={}'.format(i))
  y_pred = estimator_B.predict(X_test)
  acc.append(accuracy_score(y_test,y_pred))
```

```
plt.figure(figsize=(20, 10))
f=sns.lineplot(x=iters,y=acc)
plt.title('Find the best number of estimators in Gradient Boosting')
# Set x-axis label
plt.xlabel('number of estimators')
# Set y-axis label
plt.ylabel('Accuracy')

max_acc=max(acc)

maxInd=np.argmax(acc)
print(maxInd)
best_N_estimator=iters[maxInd]
print(best_N_estimator)
bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k")
arrowprops=dict(arrowstyle="->")
kw = dict(xycoords='data',textcoords="axes fraction",
          arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
f.annotate("The best number of estimators", xy=(best_N_estimator, max_ac
c), xytext=(0.84,0.76), **kw)
plt.show()
```

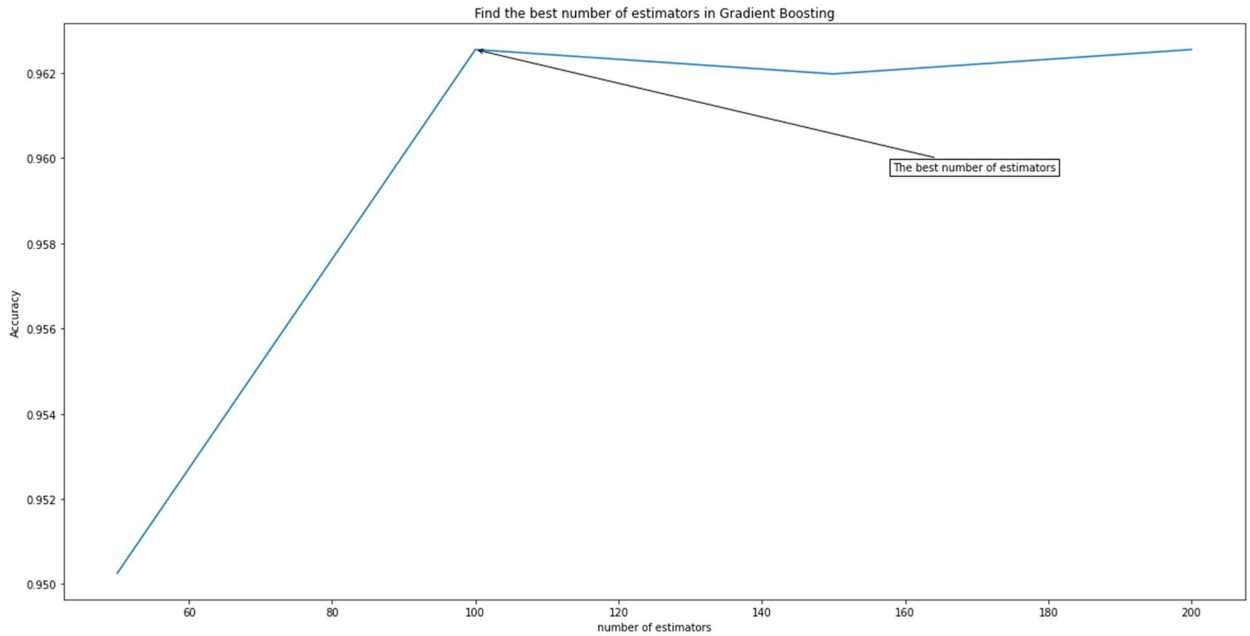

Figure 22 Plot for the best number of estimators in gradient boosting

N.B: From the previous accuracies and plot, the best number of estimators is 100.

Gradient Boosting with the best number of estimators (100):

```
estimator_B = GradientBoostingClassifier(n_estimators=best_N_estimator,
random_state=0)
start = timeit.default_timer()
estimator_B.fit(X_train, y_train)
stop = timeit.default_timer()
print('Train Time: ', stop - start)
start = timeit.default_timer()
y_pred = estimator_B.predict(X_test)
stop = timeit.default_timer()
print('prediction Time: ', stop - start)
eval(estimator_B,X_test,y_test,model_name='bestGradient Boosting with th
e best number of estimator')
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	363
1	0.91	0.93	0.92	364
2	0.94	0.99	0.96	364
3	0.97	0.99	0.98	336
4	1.00	1.00	1.00	364
5	0.99	0.93	0.96	335
6	1.00	0.99	1.00	336
7	0.99	0.90	0.94	364
8	0.93	1.00	0.96	336
9	0.91	0.96	0.93	336
accuracy			0.96	3498
macro avg	0.96	0.96	0.96	3498
weighted avg	0.96	0.96	0.96	3498

Figure 24 Gradient Boosting with 100 estimators classification report

The Test accuracy : 0.9625500285877644

Figure 23 Gradient Boosting with 100 estimators accuracy

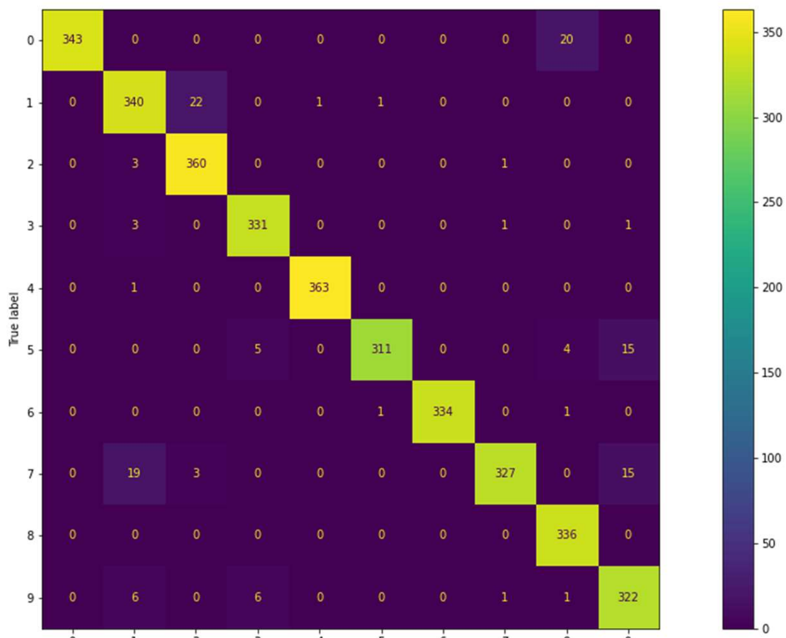

Figure 25 Gradient Boosting with 100 estimators confusion matrix

The Best Learning Rate:

```
acc=[]
LR=[0.1, 0.3, 0.6, 0.9]
for i in LR:
  estimator_B = GradientBoostingClassifier(n_estimators=best_N_estimator
,learning_rate=i, random_state=0)
  estimator_B.fit(X_train, y_train)
  eval(estimator_B,X_test,y_test,model_name='Gradient Boosting with lear
ning rate={}'.format(i))
  y_pred = estimator_B.predict(X_test)
  acc.append(accuracy_score(y_test,y_pred))
```

```
plt.figure(figsize=(20, 10))
f=sns.lineplot(x=LR,y=acc)
plt.title('Find the best Learning rate in Gradient Boosting')
# Set x-axis label
plt.xlabel('Learning rate')
# Set y-axis label
plt.ylabel('Accuracy')

max_acc=max(acc)

maxInd=np.argmax(acc)
print(maxInd)
best_LR=LR[maxInd]
print(best_LR)
bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k")
arrowprops=dict(arrowstyle="->")
kw = dict(xycoords='data',textcoords="axes fraction",
          arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
f.annotate("The best value for Learning rate", xy=(best_LR, max_acc), xy
text=(0.84,0.76), **kw)
plt.show()
```

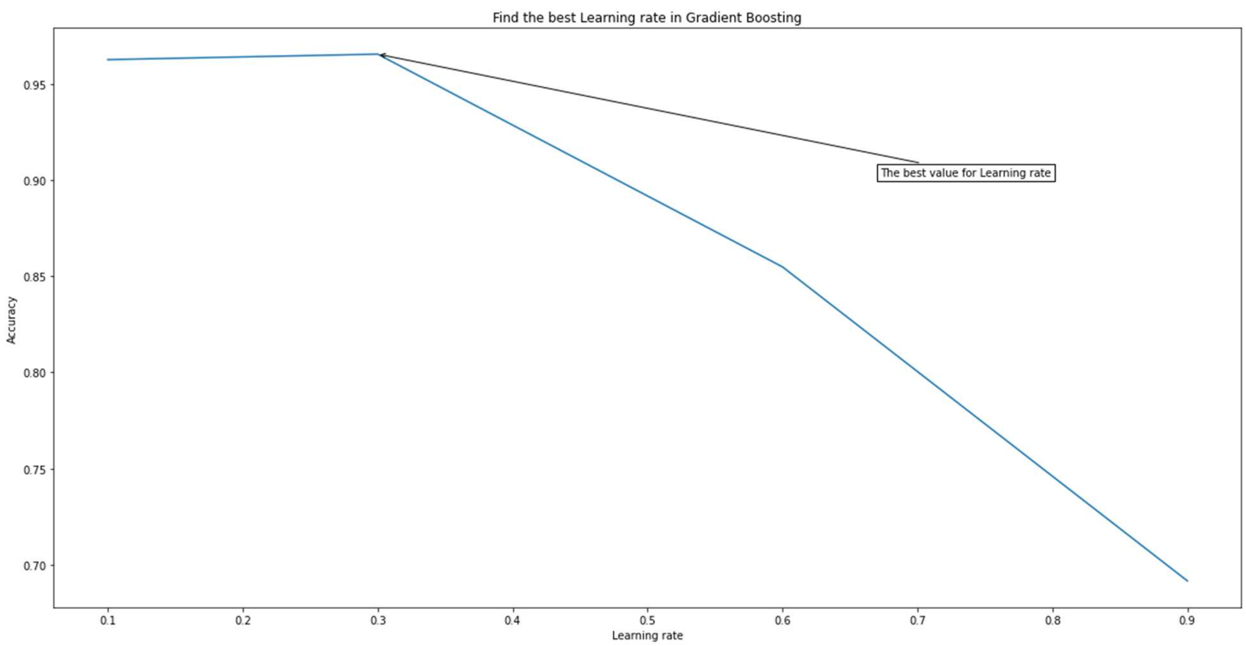

Figure 26 Plot for the best learning rate in gradient boosting

N.B: From the previous accuracies and plot, the best learning rate is 0.3.

Gradient Boosting with the best learning rate (0.3) and number of estimators (100):

```
estimator_B = GradientBoostingClassifier(n_estimators=best_N_estimator,l
earning_rate=best_LR, random_state=0)

start = timeit.default_timer()
estimator_B.fit(X_train, y_train)
stop = timeit.default_timer()
print('Train Time: ', stop - start)
start = timeit.default_timer()
y_pred = estimator_B.predict(X_test)
stop = timeit.default_timer()
print('prediction Time: ', stop - start)
eval(estimator_B,X_test,y_test,model_name='Gradient Boosting with the le
arning rate and best number of estimaters')
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.91      0.95      0.93       364
           2       0.95      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      1.00      1.00       364
           5       0.99      0.93      0.96       335
           6       1.00      0.99      1.00       336
           7       0.99      0.90      0.94       364
           8       0.93      1.00      0.96       336
           9       0.93      0.97      0.95       336

    accuracy                           0.97      3498
   macro avg       0.97      0.97      0.97      3498
weighted avg       0.97      0.97      0.97      3498
```

Figure 28 Gradient Boosting with best parameters classification report

```
The Test accuracy : 0.9654088050314465
```

Figure 27 Gradient Boosting with best parameters accuracy

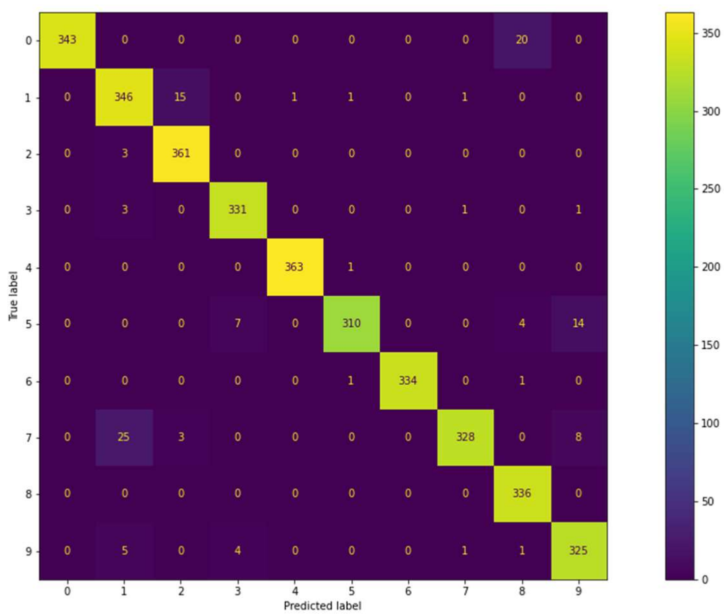

Figure 29 Gradient Boosting with best parameters confusion matrix

B)

XG-Boost with the same learning rate (0.3) and number of estimators (100):

```
xgb_Model = xgb.XGBClassifier(n_estimators=best_N_estimator,learning_rat
e=best_LR)
start = timeit.default_timer()
xgb_Model.fit(X_train, y_train)
stop = timeit.default_timer()
print('Train Time: ', stop - start)
start = timeit.default_timer()

y_pred = xgb_Model.predict(X_test)
stop = timeit.default_timer()
print('prediction Time: ', stop - start)
eval(xgb_Model,X_test,y_test,model_name='XGBoost classifier')
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	363
1	0.90	0.94	0.92	364
2	0.95	0.99	0.97	364
3	0.97	0.99	0.98	336
4	1.00	1.00	1.00	364
5	0.99	0.96	0.98	335
6	1.00	1.00	1.00	336
7	0.99	0.89	0.93	364
8	0.94	1.00	0.97	336
9	0.95	0.97	0.96	336
accuracy			0.97	3498
macro avg	0.97	0.97	0.97	3498
weighted avg	0.97	0.97	0.97	3498

Figure 30 XG-boost with the best parameters classification report

The Test accuracy : 0.9668381932532876

Figure 31 XG-boost with the best parameters accuracy

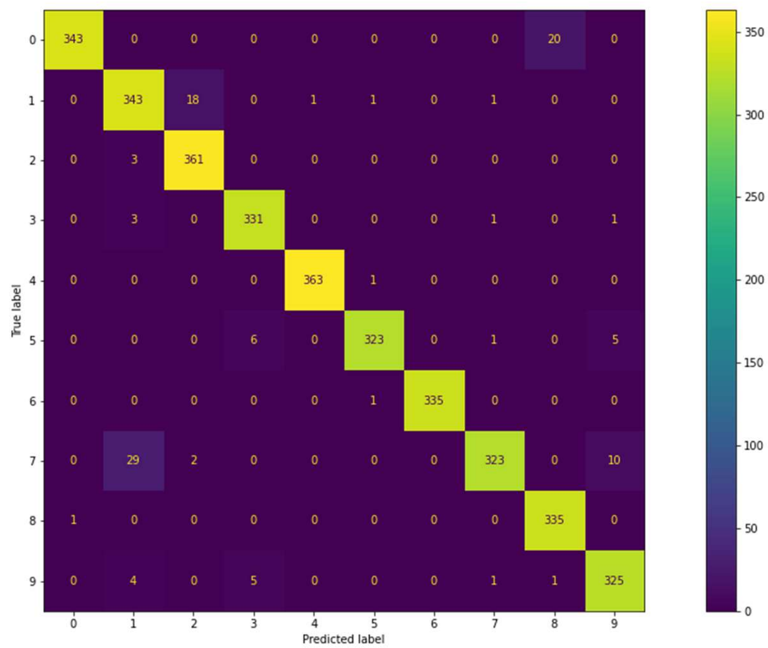

Figure 32 XG-boost with the best parameters confusion matrix

C)

In this problem each metric is good to compare performance because data almost balanced (not significant difference between classes sizes) but if you need more information about the performance in each class the confusion matrix will be better than the accuracy and if you need to easy compare with just one number the accuracy is better than confusion matrix.

The accuracies are very close in both models (XGBoost gives 96.68% accuracy and Gradient Boosting gives 96.54% accuracy), but in XGBoost the training time is 3.336 seconds and in Gradient Boosting 19.91 seconds which is greater than 6 times, it's noted that the XGBoost is faster than gradient boost because the Gradient Boosting is considering the possible loss for all possible splits to create a new branch and XGBoost solve this problem by looking at the distribution of each feature on the data and used this distribution to reduce the complexity of the tree [4].

Using Bagging approach with SVM base estimator gave the highest accuracy among all other models (98.17%). Regarding using Decision tree as base estimator in Bagging and Boosting approaches, the XGBoost gave the highest accuracy (96.68%). Boosting approach models (XGBoost and Gradient Boosting with best parameters) gave higher accuracy (96.68% and 96.54% respectively) than the Bagging approach with Decision tree as base estimator model (Bagging using 150 estimators (95.42%)).

References:

[1] P. Aznar, “Decision Trees: Gini vs Entropy ⋆ Quantdare,” *Quantdare*, Dec. 02, 2020. https://quantdare.com/decision-trees-gini-vs-entropy/

[2] J. Lawless, “Under the Hood: Using Gini impurity to your advantage in Decision Tree Classifiers,” *Medium*, Dec. 30, 2020. https://towardsdatascience.com/under-the-hood-using-gini-impurity-to-your-advantage-in-decision-tree-classifiers-9be030a650d5

[3] “Information gain in decision trees,” *Wikipedia*, Jun. 22, 2022. https://en.wikipedia.org/wiki/Information_gain_in_decision_trees#Advantages (accessed Jul. 12, 2022).

[4] “XGBoost Classifier vs Gradient Boosting Classifier | Data Science and Machine Learning,” *www.kaggle.com*. https://www.kaggle.com/getting-started/108470