Report Assignment 4

(Numerical Questions & Programming)

Applied Machine Learning ELG5255[EG]

Group Number: G_26

Prepared by:

Sawsan Awad (300327224)

Sondos Ali (300327219)

Toka Mostafa (300327284)

Problems: Part 1: Numerical Questions

Use Let's assume that TAs would go hiking every weekend, and we would make final decisions (i.e., Yes/No) according to weather, temperature, humidity, and wind. Please create a decision tree to predict our decisions based on Table 1.

1. **a) Please build a decision tree by using Gini Index (i.e.,**

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

,where NC is the number of classes).

b) Please build a decision tree by using Information Gain (i.e.,

$$IG(T, a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

, More information about IG).

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

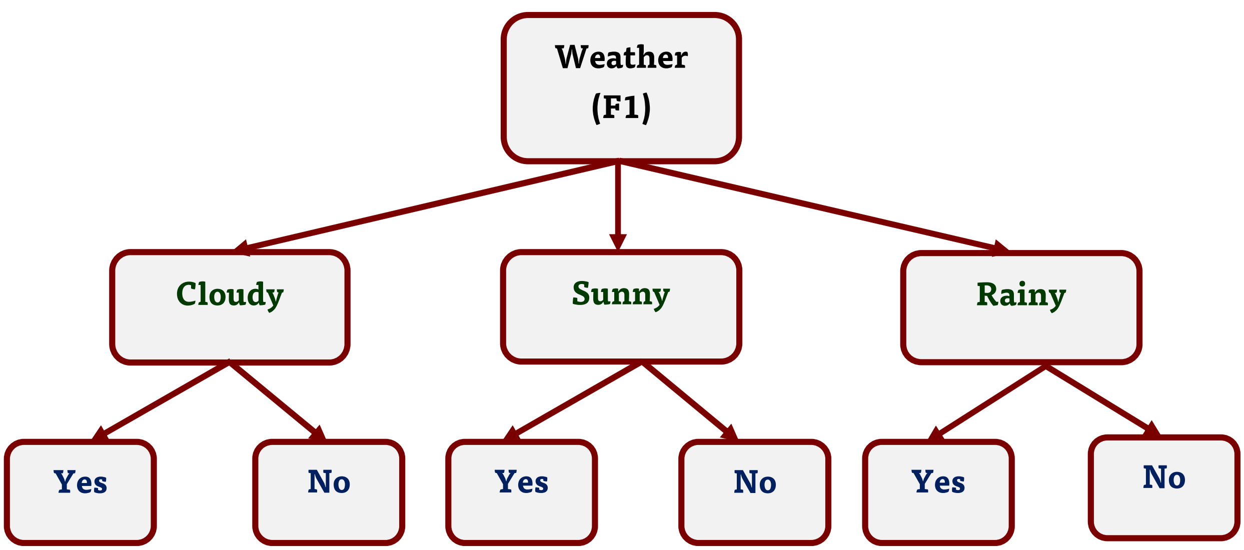
Weather (F1)	Temperature (F2)	Humidty (F3)	Wind (F4)	Hiking (Labels)
Cloudy	Cool	Normal	Weak	No
Sunny	Hot	High	Weak	Yes
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Rainy	Cool	Normal	Strong	No
Cloudy	Mild	High	Weak	Yes
Sunny	Hot	High	Strong	No
Rainy	Cool	Normal	Weak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

Solution for (a: Gini Index):

1. **Step (1): Calculate the Gini Index for Weather (F1):**

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Weather (F1)	Hiking (Labels)
Cloudy	No
Sunny	Yes
Rainy	Yes
Cloudy	No
Sunny	No
Rainy	No
Cloudy	Yes
Sunny	No
Rainy	No
Sunny	No



1.1. Calculate the Gini Index for Cloudy:

$$P(F1 = Cloudy) = \frac{3}{10}$$

$$If(F1 = Cloudy \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Cloudy \,\&\, Hiking = No) = \frac{2}{3}$$

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

1.2. Calculate the Gini Index for Sunny:

$$P(F1 = Sunny) = \frac{4}{10}$$

$$If(F1 = Sunny \,\&\, Hiking = Yes) = \frac{1}{4}$$

$$If(F1 = Sunny \,\&\, Hiking = No) = \frac{3}{4}$$

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

1.3. Calculate the Gini Index for Rainy:

$$P(F1 = Rainy) = \frac{3}{10}$$

$$If(F1 = Rainy \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Rainy \,\&\, Hiking = No) = \frac{2}{3}$$

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

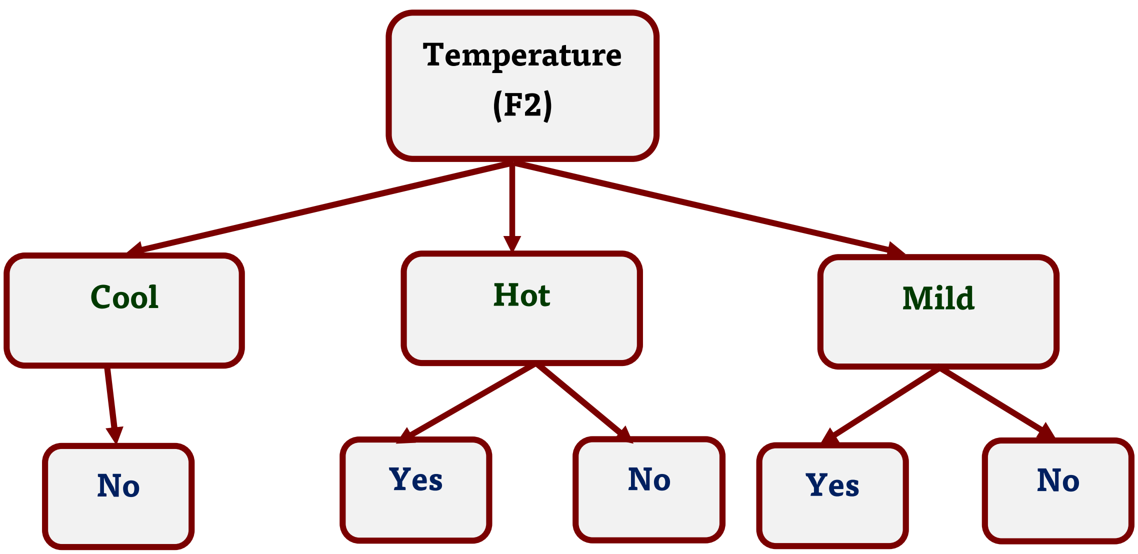
$$Gini\ Index\ for\ Weather(F1) = \left(p(F1 = Cloudy) * Gini\ Index(1)\right) + \left(p(F1 = Sunny) * Gini\ Index(2)\right) + \left(p(F1 = Rainy) * Gini\ Index(3)\right)$$

$$= \left(\frac{3}{10} * 0.44\right) + \left(\frac{4}{10} * 0.375\right) + \left(\frac{3}{10} * 0.44\right) = \boxed{0.41667}$$

2. Step (2): Calculate the Gini Index for Temprature (F2):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Temperature (F2)	Hiking (Labels)
Cool	No
Hot	Yes
Mild	Yes
Mild	No
Mild	No
Cool	No
Mild	Yes
Hot	No
Cool	No
Hot	No



2.1. Calculate the Gini Index for Cool:

$$P(F1 = Cool) = \frac{3}{10}$$

$$If(F1 = Cool \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Cool \,\&\, Hiking = No) = \frac{3}{3} = 1$$

$$Gini\ Index(1) = 1 - ((0)^2 + (1)^2) = 0$$

2.2. Calculate the Gini Index for Hot:

$$P(F1 = Hot) = \frac{3}{10}$$

$$If(F1 = Hot \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Hot \,\&\, Hiking = No) = \frac{2}{3}$$

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

2.3. Calculate the Gini Index for Mild:

$$P(F1 = Mild) = \frac{4}{10}$$

$$If(F1 = Mild \,\&\, Hiking = Yes) = \frac{2}{4}$$

$$If(F1 = Mild \,\&\, Hiking = No) = \frac{2}{4}$$

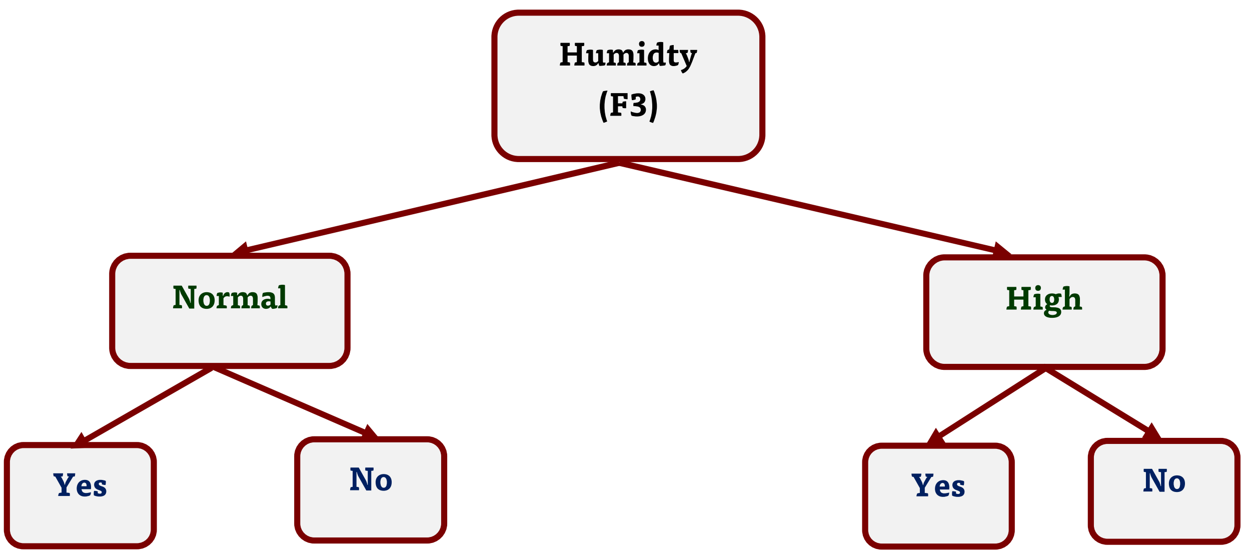
$$Gini\ Index(3) = 1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right) = 0.5$$

$$Gini\ Index\ for\ Temperature(F2)$$
$$= \big(p(F1 = Cool) * Gini\ Index(1)\big) + \big(p(F1 = Hot) * Gini\ Index(2)\big) + \big(p(F1 = Mild) * Gini\ Index(3)\big)$$
$$= \left(\frac{3}{10} * 0\right) + \left(\frac{3}{10} * 0.44\right) + \left(\frac{4}{10} * 0.5\right) \boxed{= 0.333}$$

3. Step (3): Calculate the Gini Index for Humidty (F3):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Humidty (F3)	Hiking (Labels)
Normal	No
High	Yes
Normal	Yes
High	No
High	No
Normal	No
High	Yes
High	No
Normal	No
High	No



3.1. Calculate the Gini Index for Normal:

$$P(F1 = Normal) = \frac{4}{10}$$

$$If(F1 = Normal \,\&\, Hiking = Yes) = \frac{1}{4}$$

$$If(F1 = Normal \,\&\, Hiking = No) = \frac{3}{4}$$

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

3.2. Calculate the Gini Index for High:

$$P(F1 = High) = \frac{6}{10}$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{2}{6}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{4}{6}$$

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

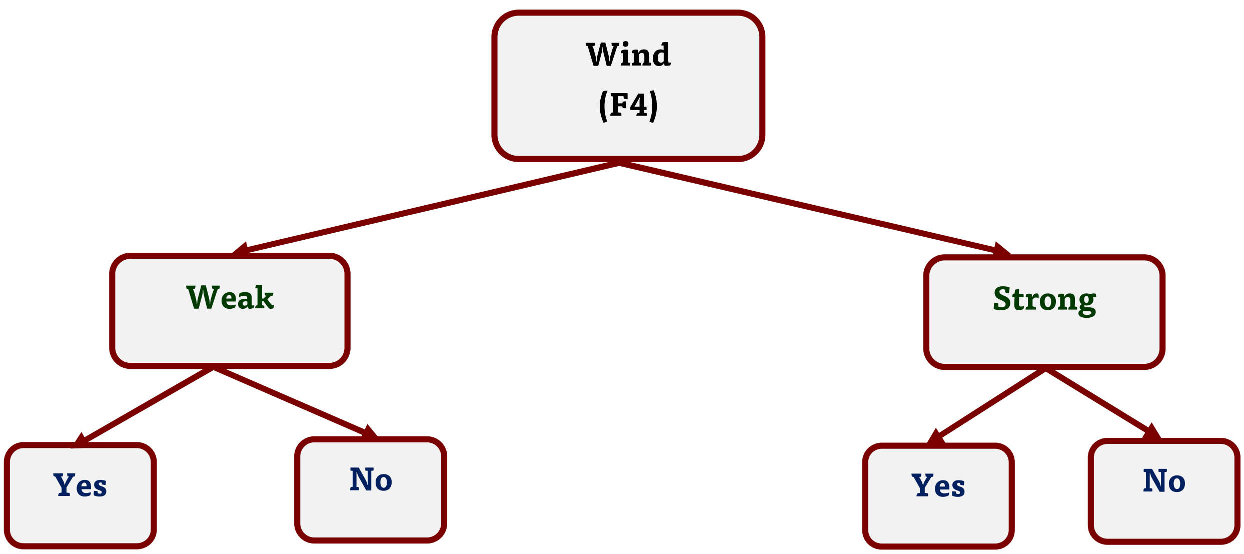
$$Gini\ Index\ for\ Humidty(F3) = \big(p(F1 = Normal) * Gini\ Index(1)\big) + \big(p(F1 = High) * Gini\ Index(2)\big)$$

$$= \left(\frac{4}{10} * 0.375\right) + \left(\frac{6}{10} * 0.44\right) = \boxed{0.41667}$$

4. **Step (4): Calculate the Gini Index for Wind (F4):**

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Wind (F4)	Hiking (Labels)
Weak	No
Weak	Yes
Strong	Yes
Strong	No
Strong	No
Strong	No
Weak	Yes
Strong	No
Weak	No
Strong	No



4.1. Calculate the Gini Index for Weak:

$$P(F1 = Weak) = \frac{4}{10}$$

$$If(F1 = Weak \,\&\, Hiking = Yes) = \frac{2}{4}$$

$$If(F1 = Weak \,\&\, Hiking = No) = \frac{2}{4}$$

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

4.2. Calculate the Gini Index for Strong:

$$P(F1 = Strong) = \frac{6}{10}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = \frac{1}{6}$$

$$If(F1 = Strong \,\&\, Hiking = No) = \frac{5}{6}$$

$$Gini\ Index(2) = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) = 0.2778$$

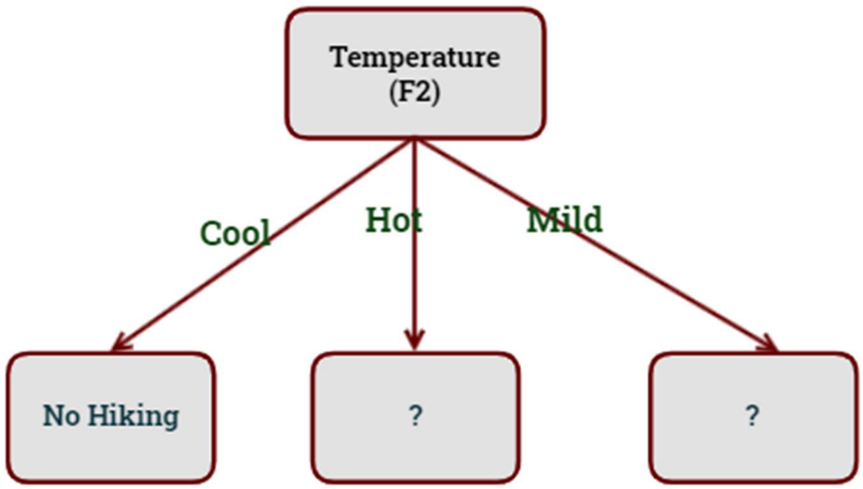
$$Gini\ Index\ for\ Wind(F4) = \big(p(F1 = Weak) * Gini\ Index(1)\big) + \big(p(F1 = Strong) * Gini\ Index(2)\big)$$

$$= \left(\frac{4}{10} * 0.5\right) + \left(\frac{6}{10} * 0.2778\right) \boxed{= 0.3667}$$

- ## Gini Index Features:

Features	Gini Index
Weather (F1)	0.41667
Temperature (F2)	0.3333
Humidty (F3)	0.41667
Wind (F4)	0.3667

- **From the above table, we observe that 'Temperature' 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.**
- **The decision tree will be:**



- **We will calculate the Gini Index for the 'Hot & Mild' branches of Temperature (because 'Cool' category ended with result 'No Hiking') as follows:**

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

5. Step (5): Calculate the Gini Index for each feature with the 2 category in Temperature (Hot & Mild):

5.1. Calculate the Gini Index for Weather (F1) with the first category in Temperature (Hot):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Weather (F1)	Temperature (F2)	Hiking (Labels)
Sunny	Hot	Yes
Sunny	Hot	No
Sunny	Hot	No

5.1.1. Calculate the Gini Index for Sunny:

$$P(F1 = Sunny) = 1$$

$$If(F1 = Sunny \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Sunny \,\&\, Hiking = No) = \frac{2}{3}$$

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

$$\begin{aligned} Gini\ Index\ for\ Weather(F1) \\ &= \big(p(F1 = Cloudy) * Gini\ Index(1)\big) \\ &+ \big(p(F1 = Sunny) * Gini\ Index(2)\big) \\ &+ \big(p(F1 = Rainy) * Gini\ Index(3)\big) \end{aligned}$$

$$= 0 + 0.44 + 0 \boxed{= 0.44}$$

5.2. Calculate the Gini Index for Humidty (F3) with the first category in Temperature (Hot):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Temperature (F2)	Humidty (F3)	Hiking (Labels)
Hot	High	Yes
Hot	High	No
Hot	High	No

5.2.1. Calculate the Gini Index for High:

$$P(F1 = High) = 1$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{2}{3}$$

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

$$\begin{aligned} Gini\ Index\ for\ Humidty(F3) \\ &= \big(p(F1 = High) * Gini\ Index(1)\big) \\ &+ \big(p(F1 = Normal) * Gini\ Index(2)\big) \\ &= 0 + 0.44 \boxed{= 0.44} \end{aligned}$$

5.3. Calculate the Gini Index for Wind (F4) with the first category in Temperature (Hot):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Temperature (F2)	Wind (F4)	Hiking (Labels)
Hot	Weak	Yes
Hot	Strong	No
Hot	Strong	No

5.3.1. Calculate the Gini Index for Weak:

$$P(F1 = Weak) = \frac{1}{3}$$

$$If(F1 = High \,\&\, Hiking = Yes) = 1$$

$$If(F1 = High \,\&\, Hiking = No) = 0$$

$$Gini\ Index(1) = 1 - 1 = 0$$

5.3.2. Calculate the Gini Index for Strong:

$$P(F1 = Strong) = \frac{2}{3}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Strong \,\&\, Hiking = No) = 1$$

$$Gini\ Index(2) = 1 - 1 = 0$$

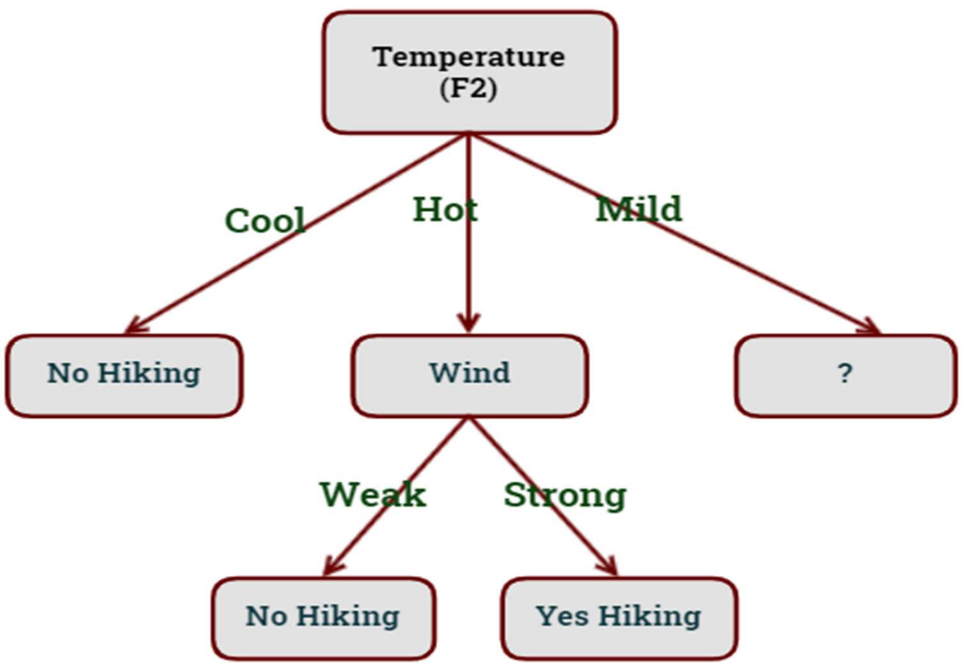
$$Gini\ Index\ for\ Wind(F4) = \big(p(F1 = Weak) * Gini\ Index(1)\big) + \big(p(F1 = Strong) * Gini\ Index(2)\big)$$

$$= \frac{1}{3} * 0 + \frac{2}{3} * 0 \boxed{= 0}$$

- **Gini Index Features based on (Hot Category):**

Features	Gini Index
Weather (F1)	0.44
Humidty (F3)	0.44
Wind (F4)	0

- **From the above table, we observe that 'Wind' has the lowest Gini Index and hence it will be chosen as the next branch to Hot category for how decision tree works.**
- **We will repeat the same procedure to determine the sub-nodes or branches of the decision tree for Mild category.**
- **The decision tree will be :**



5.4. Calculate the Gini Index for Weather (F1) with the second category in Temperature (Mild):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Weather (F1)	Temperature (F2)	Hiking (Labels)
Rainy	Mild	Yes
Cloudy	Mild	No
Sunny	Mild	No
Cloudy	Mild	Yes

5.4.1. Calculate the Gini Index for Sunny:

$$P(F1 = Sunny) = \frac{1}{4}$$

$$If(F1 = Sunny \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Sunny \,\&\, Hiking = No) = 1$$

$$Gini\ Index(1) = 1 - 1 = 0$$

5.4.2. Calculate the Gini Index for Rainy:

$$P(F1 = Rainy) = \frac{1}{4}$$

$$If(F1 = Rainy \& Hiking = Yes) = 1$$

$$If(F1 = Rainy \& Hiking = No) = 0$$

$$Gini\ Index(2) = 1 - 1 = 0$$

5.4.3. Calculate the Gini Index for Cloudy:

$$P(F1 = Coudy) = \frac{2}{4}$$

$$If(F1 = Coudy\&\ Hiking = Yes) = \frac{1}{2}$$

$$If(F1 = Coudy\&\ Hiking = No) = \frac{1}{2}$$

$$Gini\ Index(3) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) = 0.5$$

$$Gini\ Index\ for\ Weather(F1) = \left(p(F1 = Cloudy) * Gini\ Index(1)\right) + \left(p(F1 = Sunny) * Gini\ Index(2)\right) + \left(p(F1 = Rainy) * Gini\ Index(3)\right)$$

$$= \left(\frac{1}{4} * 0\right) + \left(\frac{1}{4} * 0\right) + \left(\frac{2}{4} * 0.5\right) \boxed{= 0.25}$$

5.5. Calculate the Gini Index for Humidty (F3) with the second category in Temperature (Mild):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Temperature (F2)	Humidty (F3)	Hiking (Labels)
Mild	Normal	Yes
Mild	High	No
Mild	High	No
Mild	High	Yes

5.5.1. Calculate the Gini Index for High:

$$P(F1 = High) = \frac{3}{4}$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{2}{3}$$

$$Gini\ Index(1) = 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right) \boxed{= 0.44}$$

5.5.2. Calculate the Gini Index for Normal:

$$P(F1 = Normal) = \frac{1}{4}$$

$$If(F1 = Normal \& \ Hiking = Yes) = 1$$

$$If(F1 = Normal \& \ Hiking = No) = 0$$

$$Gini\ Index(2) = 1 - 1 = 0$$

$$Gini\ Index\ for\ Humidty(F3) = \left(p(F1 = High) * Gini\ Index(1)\right) + \left(p(F1 = Normal) * Gini\ Index(2)\right)$$

$$= 0 + \frac{3}{4} * 0.44 \boxed{= 0.33}$$

5.6. Calculate the Gini Index for Wind (F4) with the second category in Temperature (Mild):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Temperature (F2)	Wind (F4)	Hiking (Labels)
Mild	Strong	Yes
Mild	Strong	No
Mild	Strong	No
Mild	Weak	Yes

5.6.1. Calculate the Gini Index for Weak:

$$P(F1 = Weak) = \frac{1}{4}$$

$$If(F1 = High \,\&\, Hiking = Yes) = 1$$

$$If(F1 = High \,\&\, Hiking = No) = 0$$

$$Gini\ Index(1) = 1 - 1 = 0$$

5.6.2. Calculate the Gini Index for Strong:

$$P(F1 = Strong) = \frac{3}{4}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = \frac{1}{3}$$

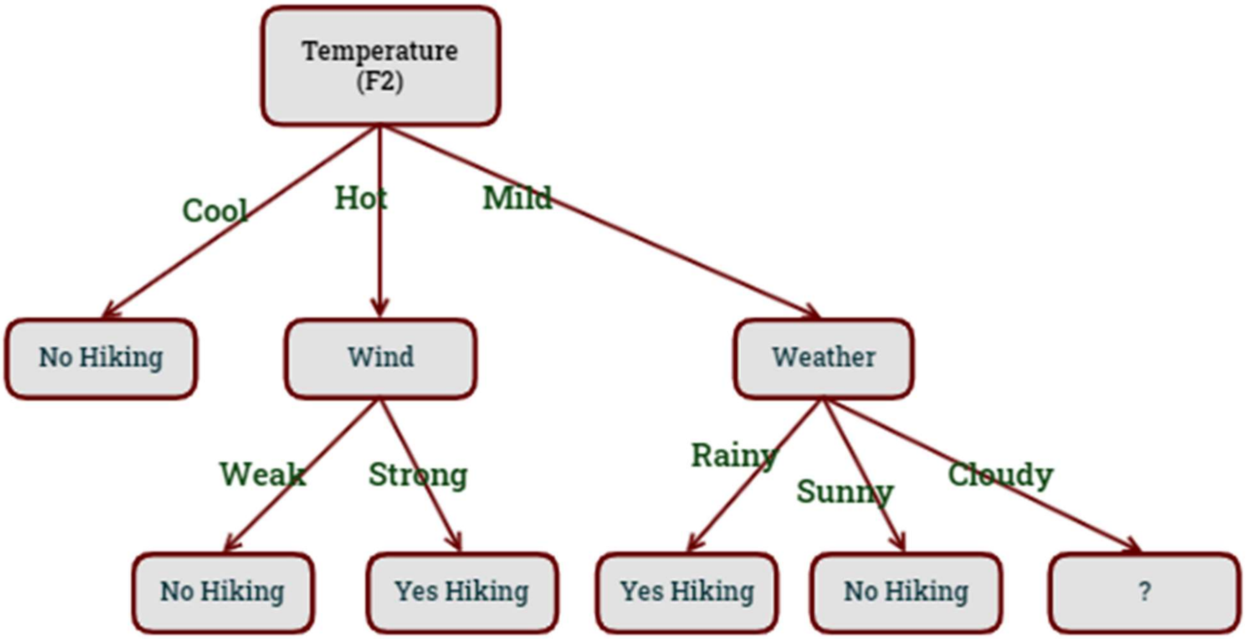
$$If(F1 = Strong \,\&\, Hiking = No) = \frac{2}{3}$$

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

$$Gini\ Index\ for\ Wind(F4)$$
$$= \big(p(F1 = Weak) * Gini\ Index(1)\big) + \big(p(F1 = Strong) * Gini\ Index(2)\big)$$
$$= 0 + \frac{3}{4} * 0.44 \boxed{= 0.33}$$

- **Gini Index Features based on (Mild Category):**

Features	Gini Index
Weather (F1)	0.25
Humidty (F3)	0.33
Wind (F4)	0.33



6. **Step (6): Calculate the Gini Index for each feature with the Cloudy category in Weather feature:**

6.1. Calculate the Gini Index for Humidty (F3):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Weather (F1)	Temperature (F2)	Humidty (F3)	Hiking (Labels)
Cloudy	Mild	High	No
Cloudy	Mild	High	Yes

6.1.1. Calculate the Gini Index for High:

$P(F1 = High) = 1$

$$\boldsymbol{If(F1 = High \,\&\, Hiking = Yes) = \frac{1}{2}}$$

$$\boldsymbol{If(F1 = High \,\&\, Hiking = No) = \frac{1}{2}}$$

$$\boldsymbol{Gini\ Index(1) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right) = 0.5}$$

$$\boldsymbol{Gini\ Index\ for\ Humidty(F3)}$$
$$\boldsymbol{= \big(p(F1 = High) * Gini\ Index(1)\big) + \big(p(F1 = Normal) * Gini\ Index(2)\big)}$$
$$\boldsymbol{= 0.44 + 0 \boxed{= 0.44}}$$

6.2. Calculate the Gini Index for Wind (F4):

$$GINI = 1 - \sum_{i}^{NC} (pi)^2$$

Weather (F1)	Temperature (F2)	Wind (F4)	Hiking (Labels)
Cloudy	Mild	Strong	No
Cloudy	Mild	Weak	Yes

6.2.1. Calculate the Gini Index for Weak:

$$P(F1 = Weak) = \frac{1}{2}$$

$$If(F1 = Weak \,\&\, Hiking = Yes) = 1$$

$$If(F1 = Weak \,\&\, Hiking = No) = 0$$

$$Gini\ Index(1) = 1 - 1 = 0$$

6.2.2. Calculate the Gini Index for Strong:

$$P(F1 = Strong) = \frac{1}{2}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Strong \,\&\, Hiking = No) = 1$$

$$Gini\ Index(2) = 1 - 1 = 0$$

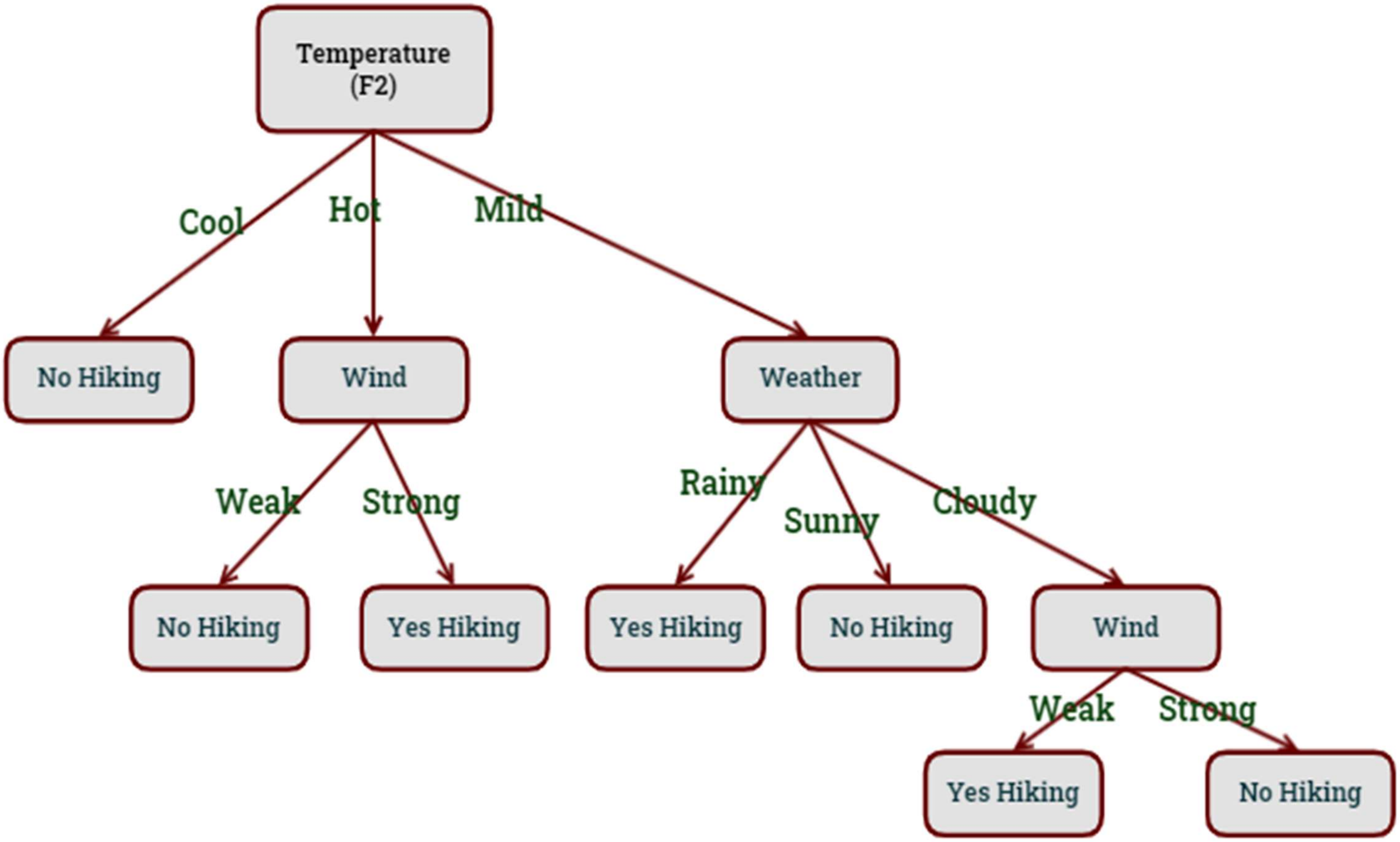
$$Gini\ Index\ for\ Wind(F4)$$
$$= \left(p(F1 = Weak) * Gini\ Index(1)\right) + \left(p(F1 = Strong) * Gini\ Index(2)\right)$$
$$= 0 + 0 \boxed{= 0}$$

- **Gini Index Features based on (Cloudy Category):**

Features	Gini Index
Humidty (F3)	0.44
Wind (F4)	0

- **From the above table, we observe that 'Wind' has the lowest Gini Index and hence it will be chosen as the next branch to Cloudy category for how decision tree works.**

- **Final Decision Tree according to Gini Index:**



Solution for (b. Gain Information):

1. **Step (1): Calculate the Information Gain for Weather (F1):**

$$IG(T, a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T) \log_2 p(j|T)$$

Weather (F1)	Hiking (Labels)
Cloudy	No
Sunny	Yes
Rainy	Yes
Cloudy	No
Sunny	No
Rainy	No
Cloudy	Yes
Sunny	No
Rainy	No
Sunny	No

- **Calculate the Entropy of Parent:**

$$\text{Entropy(Hiking)} = -\left(\frac{3}{10}\log_2\frac{3}{10} + \frac{7}{10}\log_2\frac{7}{10}\right)\boxed{= 0.8813}$$

1.1. Calculate the Entropy for Cloudy:

$$P(F1 = Cloudy) = \frac{3}{10}$$

$$If(F1 = Cloudy \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Cloudy \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(Cloudy) = -\frac{3}{10}\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.2755$$

1.2. Calculate the Entropy for Sunny:

$$P(F1 = Sunny) = \frac{4}{10}$$

$$If(F1 = Sunny \,\&\, Hiking = Yes) = \frac{1}{4}$$

$$If(F1 = Sunny \,\&\, Hiking = No) = \frac{3}{4}$$

$$Entropy(Sunny) = -\frac{4}{10}\left(\frac{1}{4}log_2\frac{1}{4} + \frac{3}{4}log_2\frac{3}{4}\right) = 0.3245$$

1.3. Calculate the Entropy for Rainy:

$$P(F1 = Rainy) = \frac{3}{10}$$

$$If(F1 = Rainy \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Rainy \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(Rainy) = -\frac{3}{10}\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.2755$$

1.4. Calculate the Entropy for Weather(F1):

$$Entropy(F1) = Entropy(Cloudy) + Entropy(Sunny) + Entropy(Rainy)$$
$$= 0.2755 + 0.3245 + 0.2755 = 0.8755$$

$$GAIN_{Info}\ for\ Weather(F1) = Entropy(Parent) - Entropy(F1)$$
$$= 0.8813 - 0.8755 \boxed{= 0.005802}$$

2. **Step (2): Calculate the Information Gain for Temperature (F2):**

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T)\log_2 p(j|T)$$

Temperature (F2)	Hiking (Labels)
Cool	No
Hot	Yes
Mild	Yes
Mild	No
Mild	No
Cool	No
Mild	Yes
Hot	No
Cool	No
Hot	No

2.1. Calculate the Entropy for Cool:

$$P(F1 = Cool) = \frac{3}{10}$$

$$If(F1 = Cool \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Cool \,\&\, Hiking = No) = \frac{3}{3} = 1$$

$$Entropy(Cool) = -\frac{3}{10}(log_2 1) = 0$$

2.2. Calculate the Entropy for Hot:

$$P(F1 = Hot) = \frac{3}{10}$$

$$If(F1 = Hot \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Hot \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(Hot) = -\frac{3}{10}\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.2755$$

2.3. Calculate the Entropy for Mild:

$$P(F1 = Mild) = \frac{4}{10}$$

$$If(F1 = Mild \,\&\, Hiking = Yes) = \frac{2}{4}$$

$$If(F1 = Mild \,\&\, Hiking = No) = \frac{2}{4}$$

$$Entropy(Mild) = -\frac{4}{10}\left(\frac{2}{4}log_2\frac{2}{4} + \frac{2}{4}log_2\frac{2}{4}\right) = 0.4$$

2.4. Calculate the Entropy for Temperature(F2):

$$Entropy(F1) = Entropy(Cool) + Entropy(Hot) + Entropy(Mild)$$
$$= 0 + 0.2755 + 0.4 = 0.6755$$

$$GAIN_{Info}\ for\ Temperature(F2) = Entropy(Parent) - Entropy(F2)$$
$$= 0.8813 - 0.6755 \boxed{= 0.205802}$$

3. **Step (3): Calculate the Information Gain for Humidty (F3):**

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T) \log_2 p(j|T)$$

Humidty (F3)	Hiking (Labels)
Normal	No
High	Yes
Normal	Yes
High	No
High	No
Normal	No
High	Yes
High	No
Normal	No
High	No

3.1. Calculate the Entropy for Normal:

$$P(F1 = Normal) = \frac{4}{10}$$

$$If(F1 = Normal \,\&\, Hiking = Yes) = \frac{1}{4}$$

$$If(F1 = Normal \,\&\, Hiking = No) = \frac{3}{4}$$

$$Entropy(Normal) = -\frac{4}{10}\left(\frac{1}{4}log_2\frac{1}{4} + \frac{3}{4}log_2\frac{3}{4}\right) = 0.3245$$

3.2. Calculate the Entropy for High:

$$P(F1 = High) = \frac{6}{10}$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{2}{6}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{4}{6}$$

$$Entropy(High) = -\frac{6}{10}\left(\frac{2}{6}log_2\frac{2}{6} + \frac{4}{6}log_2\frac{4}{6}\right) = 0.5509$$

3.3. Calculate the Entropy for Humidty (F3):

$$Entropy(F3) = Entropy(Normal) + Entropy(High)$$

$$= 0 + 0.3245 + 0.5509 = 0.8755$$

$$GAIN_{Info}\ for\ Humidty\ (F3) = Entropy(Parent) - Entropy(F3)$$

$$= 0.8813 - 0.8755 \boxed{= 0.005802}$$

4. **Step (4): Calculate the Information Gain for Wind (F4):**

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T) \log_2 p(j|T)$$

Wind (F4)	Hiking (Labels)
Weak	No
Weak	Yes
Strong	Yes
Strong	No
Strong	No
Strong	No
Weak	Yes
Strong	No
Weak	No
Strong	No

4.1. Calculate the Entropy for Weak:

$$P(F1 = Weak) = \frac{4}{10}$$

$$If(F1 = Weak \,\&\, Hiking = Yes) = \frac{2}{4}$$

$$If(F1 = Weak \,\&\, Hiking = No) = \frac{2}{4}$$

$$Entropy(Weak) = -\frac{4}{10}\left(\frac{2}{4}log_2\frac{2}{4} + \frac{2}{4}log_2\frac{2}{4}\right) = 0.4$$

4.2. Calculate the Entropy for Strong:

$$P(F1 = Strong) = \frac{6}{10}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = \frac{1}{6}$$

$$If(F1 = Strong \,\&\, Hiking = No) = \frac{5}{6}$$

$$Entropy(Strong) = -\frac{6}{10}\left(\frac{1}{6}log_2\frac{1}{6} + \frac{5}{6}log_2\frac{5}{6}\right) = 0.3901$$

4.3. Calculate the Entropy for Wind (F4):

$$Entropy(F4) = Entropy(Weak) + Entropy(Strong)$$

$$= 0.4 + 0.3901 = 0.7901$$

$$GAIN_{Info}\ for\ Wind\ (F4) = Entropy(Parent) - Entropy(F4)$$

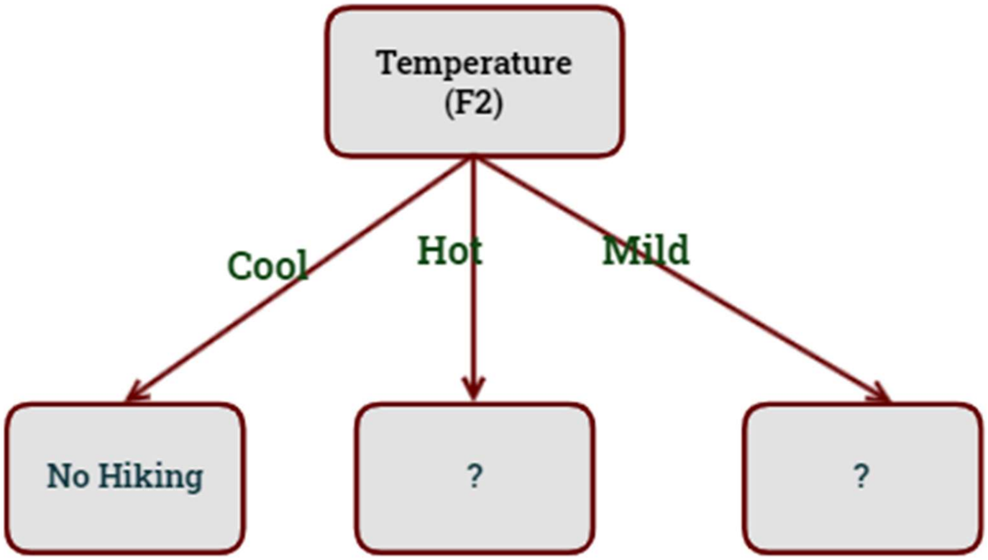
$$= 0.8813 - 0.7901 = \boxed{0.091277}$$

- **Gain Informatin Features:**

Features	Gini Index
Weather (F1)	0.005802
Temperature (F2)	0.205802
Humidty (F3)	0.005802
Wind (F4)	0.091277

- **From the above table, we observe that 'Temperature' has the highest Gain Information 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.**
- **The decision tree will be:**



- **We will calculate the Gain Information for the 'Hot & Mild' branches of Temperature (because 'Cool' category ended with result 'No Hiking') as follows:**

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

5. **Step (5): Calculate the Information Gain for each feature with the 2 category in Temperature (Hot & Mild):**

5.1. Calculate the Information Gain for Weather (F1) with the first category in Temperature (Hot):

$$IG(T, a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_j p(j|T) \log_2 p(j|T)$$

Weather (F1)	Temperature (F2)	Hiking (Labels)
Sunny	Hot	Yes
Sunny	Hot	No
Sunny	Hot	No

- **Calculate the Entropy of Parent:**

$$\text{Entropy(Hiking)} = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \boxed{= 0.918296}$$

5.1.1. Calculate the Entropy for Sunny:

$$P(F1 = Sunny) = 1$$

$$If(F1 = Sunny \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Sunny \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(Sunny) = -\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.918296$$

$$GAIN_{Info}\ for\ Weather(F1) = Entropy(Parent) - Entropy(F1)$$

$$= 0.918296 - 0.918296 \boxed{= 0}$$

5.2. Calculate the Information Gain for Humidty (F3) with the first category in Temperature (Hot):

$$IG(T, a) = \text{Entropy}(\text{T}) - \text{Entropy}(\text{T}|\text{a})$$

Calculate the Entropy:

$$Entropy(T) = -\sum_j p(j|T)\log_2 p(j|T)$$

Temperature (F2)	Humidty (F3)	Hiking (Labels)
Hot	High	Yes
Hot	High	No
Hot	High	No

5.2.1. Calculate the Entropy for High:

$$P(F1 = High) = 1$$

$$If(F1 = High\ \&\ Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = High\ \&\ Hiking = No) = \frac{2}{3}$$

$$Entropy(High) = -\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.918296$$

$$GAIN_{Info}\ for\ Humidty(F3) = Entropy(Parent) - Entropy(F3)$$

$$= 0.918296 - 0.918296 \boxed{= 0}$$

5.3. Calculate the Information Gain for Wind (F4) with the first category in Temperature (Hot):

$$IG(T, a) = \text{Entropy}(T) - \text{Entropy}(T|a)$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T) \log_2 p(j|T)$$

Temperature (F2)	Wind (F4)	Hiking (Labels)
Hot	Weak	Yes
Hot	Strong	No
Hot	Strong	No

5.3.1. Calculate the Entropy for Weak:

$$P(F1 = Weak) = \frac{1}{3}$$

$$If(F1 = High \,\&\, Hiking = Yes) = 1$$

$$If(F1 = High \,\&\, Hiking = No) = 0$$

$$Entropy(Weak) = -\frac{1}{3}(log_2 1) = 0$$

5.3.2. Calculate the Entropy for Strong:

$$P(F1 = Strong) = \frac{2}{3}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Strong \,\&\, Hiking = No) = 1$$

$$Entropy(Strong) = -\frac{2}{3}(log_2 1) = 0$$

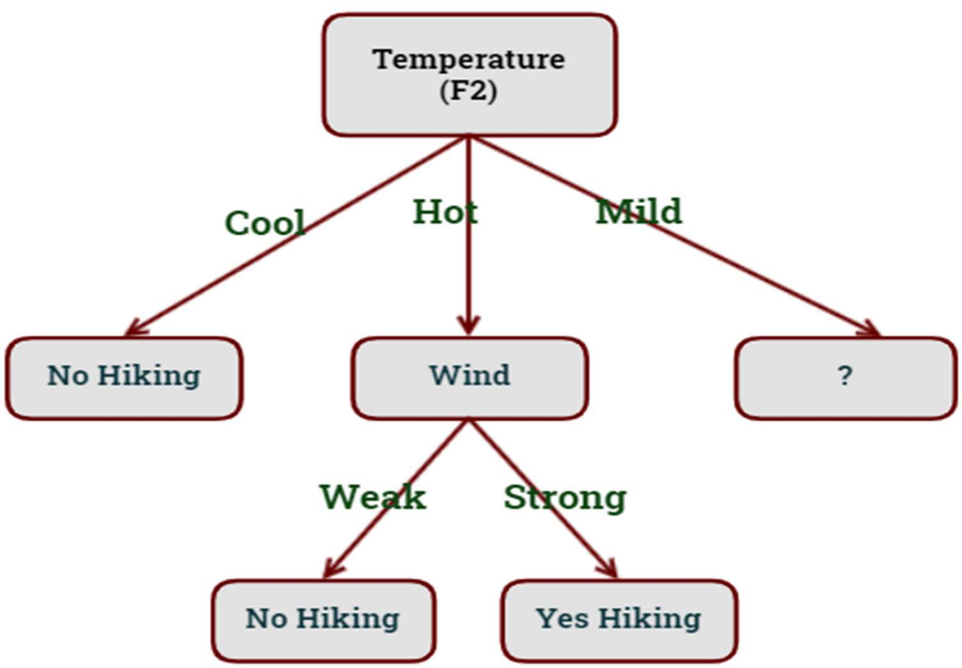
$$GAIN_{Info}\ for\ Wind(F4) = Entropy(Parent) - Entropy(F4)$$

$$= 0.918296 - 0 \boxed{= 0.918296}$$

- **Gain Informatin Features based on (Hot Category):**

Features	Gini Index
Weather (F1)	0
Humidty (F3)	0
Wind (F4)	0.918296

- **From the above table, we observe that 'Wind' has the highest Gain Information and hence it will be chosen as the next branch to Hot category for how decision tree works.**
- **We will repeat the same procedure to determine the sub-nodes or branches of the decision tree for Mild category.**
- **The decision tree will be:**



5.4. Calculate the Information Gain for Weather (F1) with the second category in Temperature (Mild):

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_j p(j|T)\log_2 p(j|T)$$

Weather (F1)	Temperature (F2)	Hiking (Labels)
Rainy	Mild	Yes
Cloudy	Mild	No
Sunny	Mild	No
Cloudy	Mild	Yes

- **Calculate the Entropy of Parent:**

$$\text{Entropy(Hiking)} = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right)\boxed{= 1}$$

5.4.1. Calculate the Entropy for Sunny:

$$P(F1 = Sunny) = \frac{1}{4}$$

$$If(F1 = Sunny \;\&\; Hiking = Yes) = 0$$

$$If(F1 = Sunny \;\&\; Hiking = No) = 1$$

$$Entropy(Sunny) = -\frac{1}{4}(log_2 1) = 0$$

5.4.2. Calculate the Entropy for Rainy:

$$P(F1 = Rainy) = \frac{1}{4}$$

$$If(F1 = Rainy \& \; Hiking = Yes) = 1$$

$$If(F1 = Rainy \& \; Hiking = No) = 0$$

$$Entropy(Rainy) = -\frac{1}{4}(log_2 1) = 0$$

5.4.3. Calculate the Entropy for Cloudy:

$$P(F1 = Coudy) = \frac{2}{4}$$

$$If(F1 = Coudy \& \; Hiking = Yes) = \frac{1}{2}$$

$$If(F1 = Coudy \& \; Hiking = No) = \frac{1}{2}$$

$$Entropy(Cloudy) = -\frac{2}{4}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 0.5$$

5.4.4. Calculate the Entropy for Weather(F1):

$$\begin{aligned} Entropy(F1) &= Entropy(Cloudy) + Entropy(Sunny) + Entropy(Rainy) \\ &= 0 + 0 + 0.5 = 0.5 \end{aligned}$$

$$\begin{aligned} GAIN_{Info} \; for \; Weather(F1) &= Entropy(Parent) - Entropy(F1) \\ &= 1 - 0.5 = \boxed{0.5} \end{aligned}$$

5.5. Calculate the Information Gain for Humidty (F3) with the second category in Temperature (Mild):

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_j p(j|T) \log_2 p(j|T)$$

Temperature (F2)	Humidty (F3)	Hiking (Labels)
Mild	Normal	Yes
Mild	High	No
Mild	High	No
Mild	High	Yes

5.5.1. Calculate the Entropy for High:

$$P(F1 = High) = \frac{3}{4}$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(High) = -\frac{3}{4}\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.688722$$

5.5.2. Calculate the Entropy for Normal:

$$P(F1 = Normal) = \frac{1}{4}$$

$$If(F1 = Normal \& \ Hiking = Yes) = 1$$

$$If(F1 = Normal \& \ Hiking = No) = 0$$

$$Entropy(Normal) = -\frac{1}{4}(log_2 1) = 0$$

5.5.3. Calculate the Entropy for Humidty (F3):

$$Entropy(F3) = Entropy(Normal) + Entropy(High)$$

$$= 0 + 0.688722 = 0.688722$$

$$GAIN_{Info} \ for \ Humidty(F3) = Entropy(Parent) - Entropy(F3)$$

$$= 1 - 0.688722 \boxed{= 0.311278}$$

5.6. Calculate the Information Gain for Wind (F4) with the second category in Temperature (Mild):

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T)\log_2 p(j|T)$$

Temperature (F2)	Wind (F4)	Hiking (Labels)
Mild	Strong	Yes
Mild	Strong	No
Mild	Strong	No
Mild	Weak	Yes

5.6.1. Calculate the Entropy for Weak:

$$P(F1 = Weak) = \frac{1}{4}$$

$$If(F1 = High \,\&\, Hiking = Yes) = 1$$

$$If(F1 = High \,\&\, Hiking = No) = 0$$

$$Entropy(Weak) = -\frac{1}{4}(log_2 1) = 0$$

5.6.2. Calculate the Entropy for Strong:

$$P(F1 = Strong) = \frac{3}{4}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = \frac{1}{3}$$

$$If(F1 = Strong \,\&\, Hiking = No) = \frac{2}{3}$$

$$Entropy(Strong) = -\frac{3}{4}\left(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}\right) = 0.688722$$

5.6.3. Calculate the Entropy for Wind (F4):

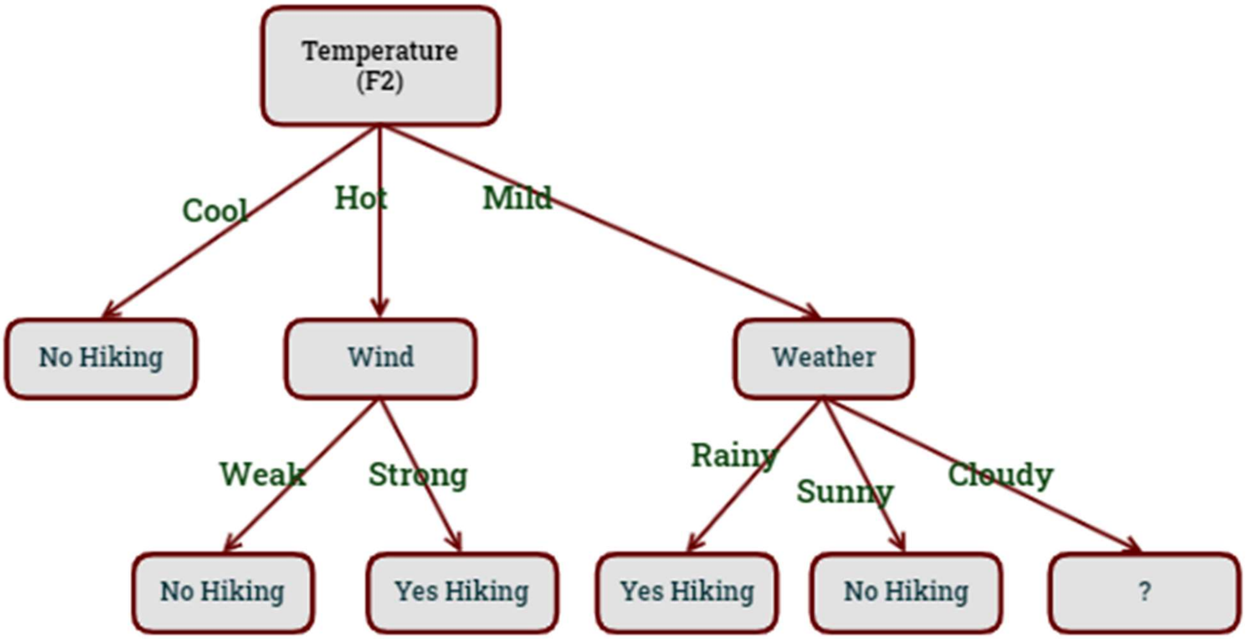
$$Entropy(F4) = Entropy(Weak) + Entropy(Strong)$$
$$= 0 + 0.688722 = 0.688722$$

$$GAIN_{Info}\ for\ Wind(F4) = Entropy(Parent) - Entropy(F4)$$
$$= 1 - 0.688722 \boxed{= 0.311278}$$

- **Gain Informatin Features based on (Mild Category):**

Features	Gini Index
Weather (F1)	0.5
Humidty (F3)	0.311278
Wind (F4)	0.311278

- **From the above table, we observe that 'Weather' has the highest Gain Information and hence it will be chosen as the next branch to Mild category 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 Gain Information for the 'Cloudy' branch of Weather (because 'Rainy' category ended with result 'Yes Hiking' & 'Sunny' category ended with result 'No Hiking') as follows:**
- **The decision tree will be:**



6. **Step (6): Calculate the Information Gain for each feature with the Cloudy category in Weather feature:**

   6.1. Calculate the Information Gain for Humidty (F3):

$$IG(T,a) = \text{Entropy}(\text{T}) - \text{Entropy}(\text{T}|\text{a})$$

Calculate the Entropy:

$$Entropy(T) = -\sum_{j} p(j|T)\log_2 p(j|T)$$

Weather (F1)	Temperature (F2)	Humidty (F3)	Hiking (Labels)
Cloudy	Mild	High	No
Cloudy	Mild	High	Yes

- **Calculate the Entropy of Parent:**

$$\textbf{Entropy(Hiking)} = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

6.1.1. Calculate the Entropy for High:

$$P(F1 = High) = 1$$

$$If(F1 = High \,\&\, Hiking = Yes) = \frac{1}{2}$$

$$If(F1 = High \,\&\, Hiking = No) = \frac{1}{2}$$

$$Entropy(High) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$GAIN_{Info}\ for\ Humidty(F3) = Entropy(Parent) - Entropy(F3)$$

$$= 1 - 1 \boxed{= 0}$$

6.2. Calculate the Information Gain for Wind (F4):

$$IG(T,a) = \text{Entropy(T)} - \text{Entropy(T|a)}$$

Calculate the Entropy:

$$Entropy(T) = -\sum_j p(j|T)\log_2 p(j|T)$$

Weather (F1)	Temperature (F2)	Wind (F4)	Hiking (Labels)
Cloudy	Mild	Strong	No
Cloudy	Mild	Weak	Yes

6.2.1. Calculate the Entropy for Weak:

$$P(F1 = Weak) = \frac{1}{2}$$

$$If(F1 = Weak \,\&\, Hiking = Yes) = 1$$

$$If(F1 = Weak \,\&\, Hiking = No) = 0$$

$$Entropy(Weak) = -\frac{1}{2}(log_2 1) = 0$$

6.2.2. Calculate the Entropy for Strong:

$$P(F1 = Strong) = \frac{1}{2}$$

$$If(F1 = Strong \,\&\, Hiking = Yes) = 0$$

$$If(F1 = Strong \,\&\, Hiking = No) = 1$$

$$Entropy(Strong) = -\frac{1}{2}(log_2 1) = 0$$

6.2.3. Calculate the Entropy for Wind (F4):

$$Entropy(F4) = \ Entropy(Weak) + Entropy(Strong)$$
$$= \ 0 + 0 = 0$$

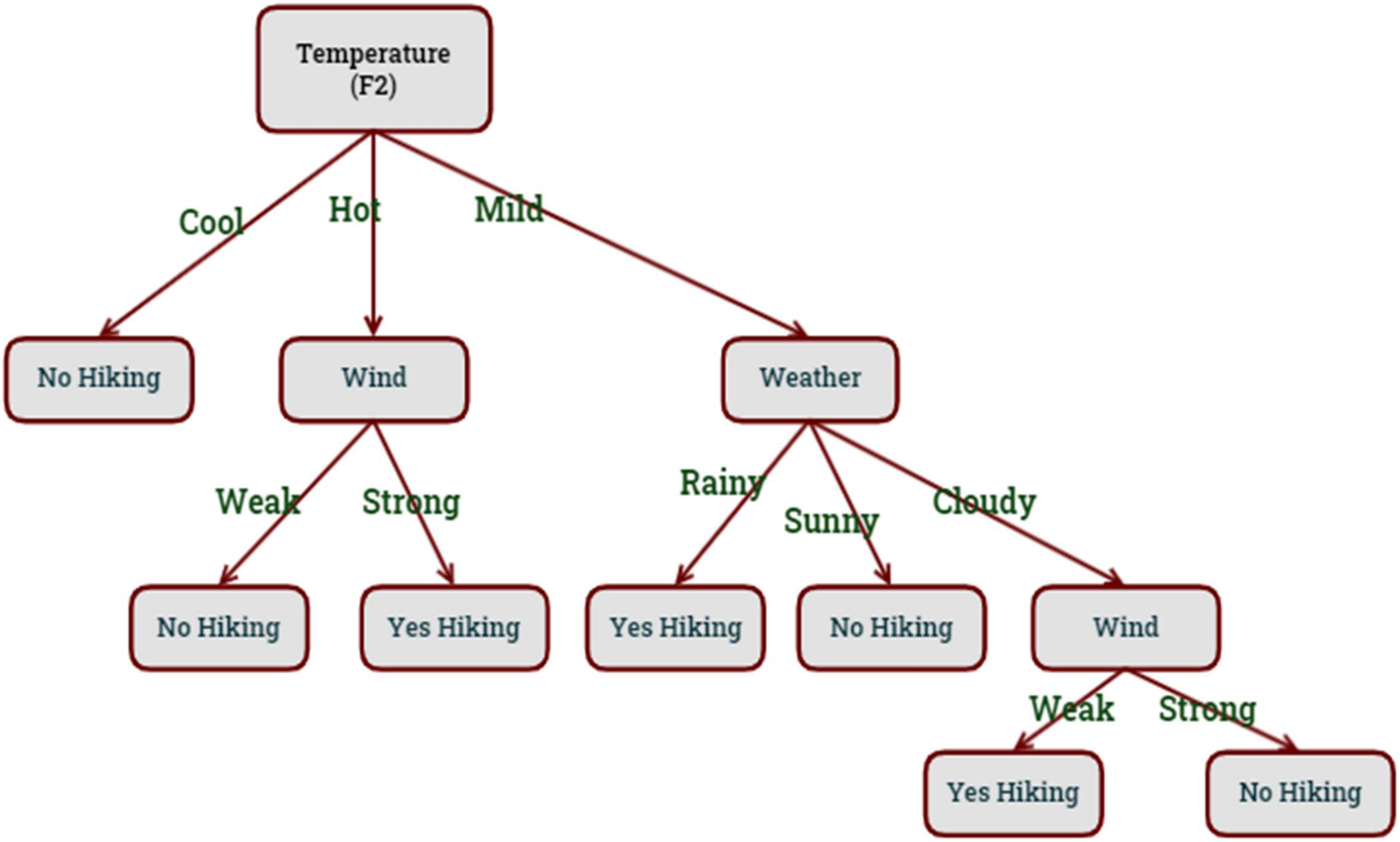
$$GAIN_{Info} \ for \ Wind(F4) = Entropy(Parent) - Entropy(F4)$$
$$= \ 1 - \ 0 = 1$$

- **Gain Informatin Features based on (Cloudy Category):**

Features	Gini Index
Humidty (F3)	0
Wind (F4)	1

- **From the above table, we observe that 'Wind' has the highest Gain Information and hence it will be chosen as the next branch to Cloudy category for how decision tree works.**

- **Final Decision Tree according to Gain Information:**



Solution for (c. Comparison between Gini Index & Information Gain):

1. **Information Gain:** **Entropy plays an important role in measuring the information gain. However, "Information gain is based on the information theory". It is used for determining the best features/attributes that render maximum information about a class. It follows the concept of entropy while aiming at decreasing the level of entropy, beginning from the root node to the leaf nodes. Information gain computes the difference between entropy before and after split and specifies the impurity in class elements.**

   ➢ **Information Gain = Entropy before splitting - Entropy after splitting**

   ✓ **Generally, it is not preferred as it involves 'log' function that results in the computational complexity.**

   - **Moreover;**
     - ✓ **Information gain is non-negative.**
     - ✓ **Information Gain is symmetric such that switching of the split variable and target variable, the same amount of information gain is obtained. (Source)**
     - ✓ **Information gain determines the reduction of the uncertainty after splitting the dataset on a particular feature such that if the value of information gain increases, that feature is most useful for classification.**
     - ✓ **The feature having the highest value of information gain is accounted for as the best feature to be chosen for split.**

2. **Gini Index:** **computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of gini coefficient. It works on categorical variables, provides outcomes either be "successful" or "failure" and hence conducts binary splitting only. The degree of gini index varies from 0 to 1,**
   - **Where 0 depicts that all the elements be allied to a certain class, or only one class exists there.**

- **The gini index of value as 1 signifies that all the elements are randomly zdistributed across various classes, and**
- **A value of 0.5 denotes the elements are uniformly distributed into some classes.**

Gini Index Vs. Information Gain:

1. **Gini index is measured by subtracting the sum of squared probabilities of each class from one, in opposite of it, information gain is obtained by multiplying the probability of the class by log (base= 2) of that class probability.**
2. **Gini index favours larger partitions (distributions) and is very easy to implement whereas information gain supports smaller partitions (distributions) with various distinct values, i.e there is a need to perform an experiment with data and splitting criterion.**
3. **While working on categorical data variables, gini index gives results either in "success" or "failure" and performs binary splitting only, in contrast to this, information gain measures the entropy differences before and after splitting and depicts the impurity in class variables.**

✓ **So Gini Index, unlike information gain, isn't computationally intensive as it doesn't involve the logarithm function used to calculate entropy in information gain. This is why Gini Index is preferred over Information gain.**

Part 2: Programming

We followed some defined steps to obtain the aimed results:

2.1. Importing important libraries:

- **NumPy library:** it provides a lot of supporting functions that make working with ndarray very easy.
- **Pandas library:** it helps us to analyze and understand data better.
- **Matplotlib.pyplot library:** used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.
- **Seaborn library:** is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.
- **from sklearn.linear_model import LogisticRegression:** is a classification algorithm rather than regression algorithm. Based on a given set of independent variables, it is used to estimate discrete value (0 or 1, yes/no, true/false).
- **SVM:** is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.
- **Decision Tree:** is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.
- **from sklearn.metrics import classification_report, accuracy_score:**
    - **Classification_report:** is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model, and it will return accuracy.

- **The accuracy_score:** is function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.

- **Other libraries will be shown their importance in the code.**

Importing the libraries

```
[ ] #Essential libraries
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    %matplotlib inline

    #Machine learning
    from sklearn.svm import SVC
    from sklearn.ensemble import VotingClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.metrics import accuracy_score
    from sklearn.ensemble import BaggingClassifier
    from sklearn.ensemble import GradientBoostingClassifier
    from xgboost import XGBClassifier

    #Evaluation
    from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
```

2.2. Importing dataset:

- **First,** we use pd.read to read the 2 datasets.
- **Second,** we use .head() function to display the first five rows of the data frame by default.

Importing the dataset

```
[ ] train = pd.read_csv("/content/pendigits-tra.csv")
    test = pd.read_csv("/content/pendigits-tes.csv")
    train.head()
```

	47	100	27	81	57	37	26	0	0.1	23	56	53	100.1	90	40	98	8
0	0	89	27	100	42	75	29	45	15	15	37	0	69	2	100	6	2
1	0	57	31	68	72	90	100	100	76	75	50	51	28	25	16	0	1
2	0	100	7	92	5	68	19	45	86	34	100	45	74	23	67	0	4
3	0	67	49	83	100	100	81	80	60	60	40	40	33	20	47	0	1
4	100	100	88	99	49	74	17	47	0	16	37	0	73	16	20	20	6

```
[ ] test.head()
```

	88	92	2	99	16	66	94	37	70	0	0.1	24	42	65	100	100.1	8
0	80	100	18	98	60	66	100	29	42	0	0	23	42	61	56	98	8
1	0	94	9	57	20	19	7	0	20	36	70	68	100	100	18	92	8
2	95	82	71	100	27	77	77	73	100	80	93	42	56	13	0	0	9
3	68	100	6	88	47	75	87	82	85	56	100	29	75	6	0	0	9
4	70	100	100	97	70	81	45	65	30	49	20	33	0	16	0	0	1

2.3. Spiliting the data into features and target:

- **First,** **we split the 2 datasets by a unique function called .iloc[], which is used to select a value that belongs to a particular row or column from a set of values of a data frame or dataset.**
  - **For example, from our code, we gave it [:,:-1] in x_train & x_test, which means that retrieve all rows and all columns except the last column.**
- **Second, we use .iloc[:,-1].values with y_train & y_test to select all rows and only the last column.**

splitting the dataset into features and target

```
[ ] x_train = train.iloc[:,:-1].values
    y_train = train.iloc[:,-1].values
    x_test = test.iloc[:,:-1].values
    y_test = test.iloc[:,-1].values
    print(x_train)
    print(x_test)
    print(y_train)
    print(y_test)

[[  0  89  27 ...   2 100   6]
 [  0  57  31 ...  25  16   0]
 [  0 100   7 ...  23  67   0]
 ...
 [100  98  60 ...   0   0   5]
 [ 59  65  91 ...   1 100   0]
 [  0  78  29 ...  36 100  40]]
[[ 80 100  18 ...  61  56  98]
 [  0  94   9 ... 100  18  92]
 [ 95  82  71 ...  13   0   0]
 ...
 [ 56 100  27 ...  93  38  93]
 [ 19 100   0 ...  97  10  81]
 [ 38 100  37 ...  26  65   0]]
[2 1 4 ... 5 1 7]
[8 8 9 ... 0 0 4]
```

2.4. Evaluation Function:

1. **Decision tree:**
   - ✓ The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data).
   - ✓ In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. Based on the comparison, we follow the branch corresponding to that value and jump to the next node.

- **Implementation of Decision Tree:**
  - **First,** importing DecisionTreeClassifier from sklearn.
  - **Second,** trained the model using (.fit) passed to it x_train, y_train.
  - **Third,** testing the model using (.predict) passed to it x_test.
  - **Fourth,** compute the accuracy using (accuracy_score) passed to it y_test, y_predict.
  - **Finally,** display the confusion matrix.

✓ **Accuracy of decision tree: 92%.**

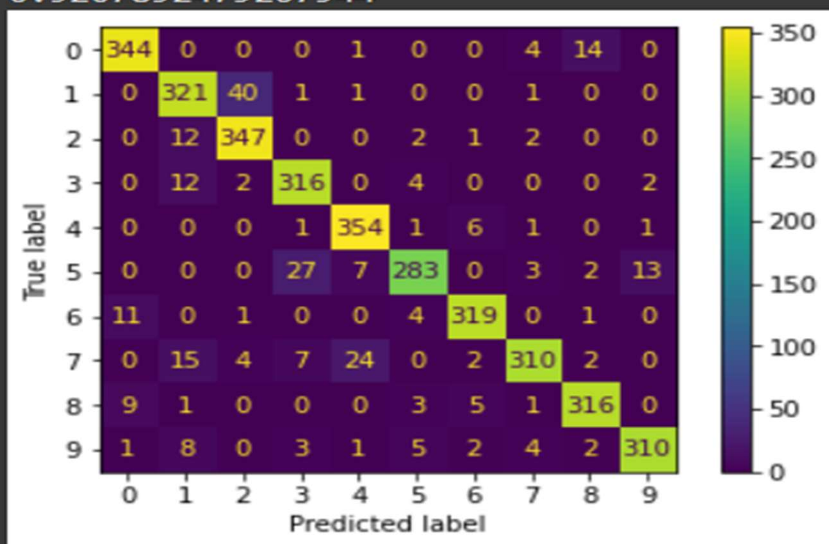
Evaluation function

```
[ ] def evaluation(y_test,y_pred,clf):
      acc = accuracy_score(y_test,y_pred)
      cm = confusion_matrix(y_test,y_pred)
      disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
      disp.plot()
      return acc
```

Decision tree

```
[ ]  dt = DecisionTreeClassifier(random_state =0)
     dt.fit(x_train,y_train)
     y_dt = dt.predict(x_test)
     acc_dt = evaluation(y_test,y_dt,dt)
     print(acc_dt)
```

```
0.9207892479267944
```



2. **Bagging:**
    - ✓ **Is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset.**
    - ✓ **In Each base classifier is trained in parallel with a training set.**

2.1. We used Bagging on two modes:

- **SVM**
- **Decision Tree**

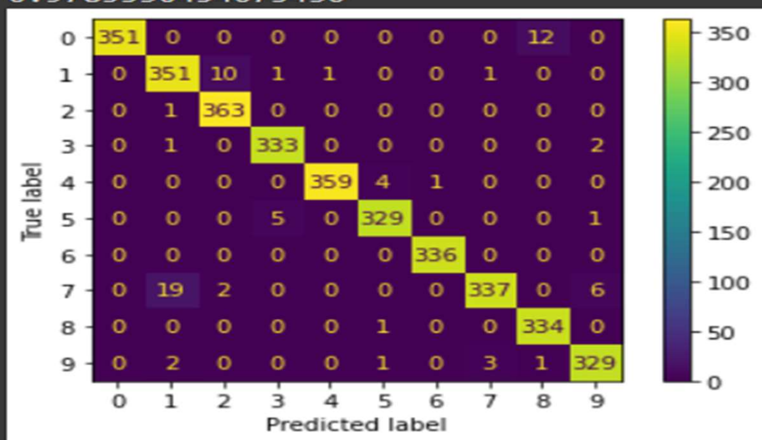
We trained the two models on the training set and made predictions on the testing set.

✓ **SVM accuracy: 0.9785530454675436**

A) Bagging for SVM

```
[ ]  bagging_svm = BaggingClassifier(SVC(), random_state=0)
     Bag_svm=bagging_svm.fit(x_train, y_train)
     y_pred_svm = Bag_svm.predict(x_test)
     acc_svm = evaluation(y_test,y_pred_svm,Bag_svm)
     print(acc_svm)
```

```
0.9785530454675436
```

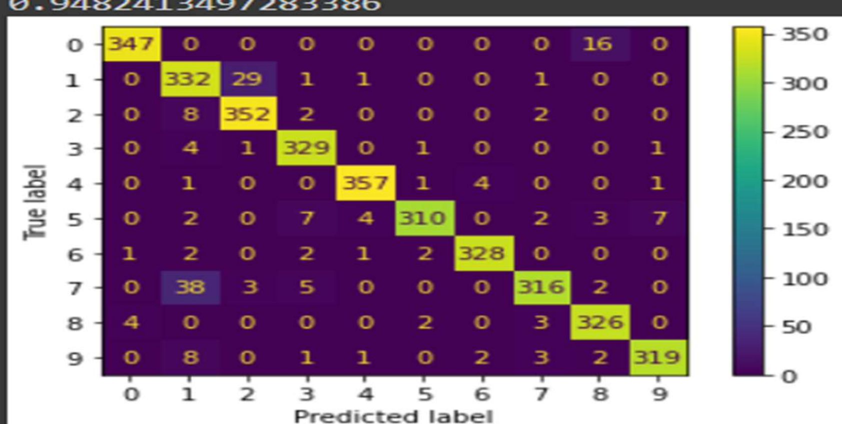


- ✓ **Decision Tree accuracy: 0.9482413497283386**

Bagging for Decision Tree

```
[ ]  bagging_clf_DT = BaggingClassifier(
         DecisionTreeClassifier(), random_state=0)
     Bag_DecisionTree=bagging_clf_DT.fit(x_train, y_train)
     y_pred_DT = Bag_DecisionTree.predict(x_test)
     acc_DT = evaluation(y_test,y_pred_DT,Bag_DecisionTree)
     print(acc_DT)
```

```
0.9482413497283386
```



2.5. Utilize majority voting:

- **We did majority voting (Soft or hard) by using Voting Classier from Sklearn to get the final decision**
- **We trained the Voting Classier model on the training set and made predictions on the testing set.**
  - **Hard Voting: the predicted output class is the class with the highest majority of votes.**
  - **Soft Voting: the output class is the prediction based on the average of probability given to that class.**

✓ **The accuracy of Hard voting: 0.9602516442665141**
✓ **The accuracy of Soft voting: 0.976551329711181**

utilize majority voting

```
[ ] # group / ensemble of models
    estimator = []
    estimator.append(('SVC', bagging_svm))
    estimator.append(('DTC', bagging_clf_DT))

    # Voting Classifier with hard voting
    vot_hard = VotingClassifier(estimators = estimator, voting ='hard')
    vot_hard.fit(x_train, y_train)
    y_pred_svm = vot_hard.predict(x_test)

    # using accuracy_score metric to predict accuracy
    score = accuracy_score(y_test, y_pred_svm)
    print("Hard Voting Score " , score)

    # Voting Classifier with soft voting
    vot_soft = VotingClassifier(estimators = estimator, voting ='soft')
    vot_soft.fit(x_train, y_train)
    y_pred_DT = vot_soft.predict(x_test)

    # using accuracy_score
    score = accuracy_score(y_test, y_pred_DT)
    print("Soft Voting Score ", score)
```

```
Hard Voting Score  0.9602516442665141
Soft Voting Score  0.976551329711181
```

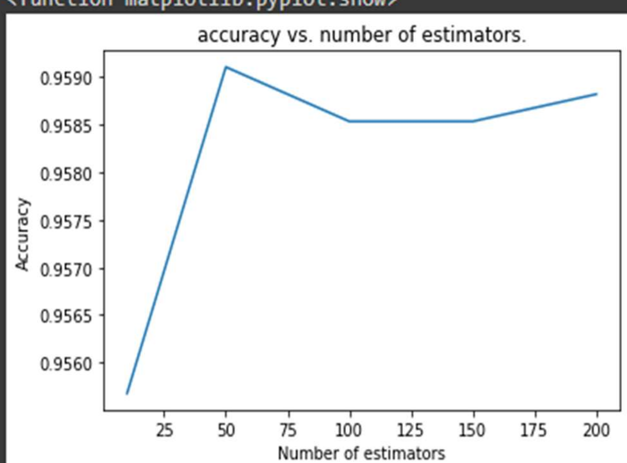
- **We tried five different values [10,50,100,150,200] taking the Decision Tree base estimator, trained the model on the training set, and made predictions on the testing set.**
- **We used accuracy_score from sklearn to print the accuracies of the five estimators.**

b)

```
Accuracy=[]
N_Est=[10,50,100,150,200]
for i in N_Est:

  tree = bagging_clf_DT
  estimator = BaggingClassifier(tree, n_estimators=i, random_state=2022)
  estimator.fit(x_train, y_train)
  y_pred = estimator.predict(x_test)
  report = accuracy_score(y_test, y_pred)
  Accuracy.append(report)
print(Accuracy)
plt.plot(N_Est,Accuracy)
plt.title('accuracy vs. number of estimators')
plt.xlabel("Number of estimators")
plt.ylabel("Accuracy")
plt.show
```

```
[0.9556762939662568, 0.9591078066914498, 0.9585358879039176, 0.9585358879039176, 0.9588218472976837]
<function matplotlib.pyplot.show>
```



From the previous figure ,the best number of estimators = 50

✓ **We Found that the best number of estimators as taking Decision Tree base estimator = 50**

3. **Boosting:**
    - ✓ **Boosting is a special type of Ensemble Learning technique that works by combining several weak learners (predictors with poor accuracy) into a strong learner (a model with strong accuracy). This works by each model paying attention to its predecessor's mistakes.**
    - ✓ **In Each base classifier is trained in parallel with a training set.**

3.1. The three most popular boosting methods are:

1. **Adaptive Boosting**
2. **Gradient Boosting**
3. **Xgboost**

3.1.1. Gradient boosting classifier

- **In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it fits a new predictor to the residual errors made by the previous predictor.**
- **To make initial predictions on the data, the algorithm will get the log of the odds of the target feature. This is usually the number of True values (values equal to 1) divided by the number of False values (values equal to 0).**
- **Once it has done this, it builds a new Decision Tree that tries to predict the residuals that were previously calculated.**

- **Implementation of Gradient Boosting Classifier:**
    - **First,** **importing GradientBoostingClassifier from sklearn.**
    - **Second,** **set four numbers for a number of estimators which are [10,50,100,200], and four numbers for learning rate. After that, apply for loop to find the best combination between these two hyperparameters.**
    - **Third,** **we trained the model using (.fit) and passed to it the x_train, y_train.**

- **Fourth,** we test the model using (.predict) and passed to it the x_test.
- **Finally,** we apply the accuracy score to figure out the accuracy of each combination.

Gradient boosting

```
[ ] accuracies = []
    for i in [10,50,100,200]:
      for j in [0.1,0.3,0.5,0.7]:
        gradient = GradientBoostingClassifier(n_estimators=i,learning_rate=j)
        gradient.fit(x_train,y_train)
        y_gb = gradient.predict(x_test)
        acc_gb = accuracy_score(y_test,y_gb)
        accuracies.append(acc_gb)
        print("Accuracy of gradient boosting of no. of estimator" ,i, "and learning rate" ,j,": ",acc_gb)
    print("argmax: ", np.argmax(accuracies))
```

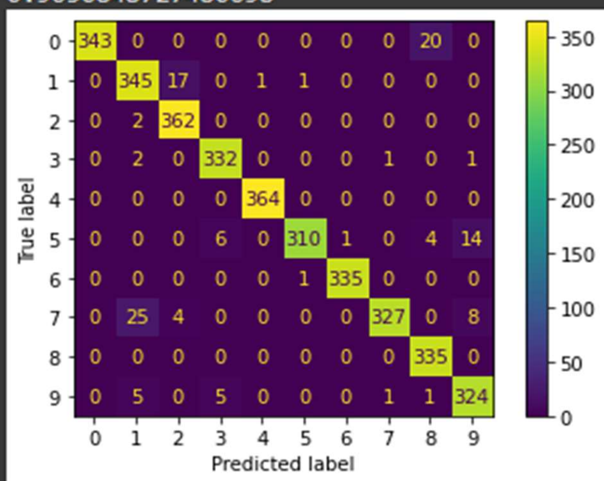
```
Accuracy of gradient boosting of no. of estimator 10 and learning rate 0.1 :  0.872748069774092
Accuracy of gradient boosting of no. of estimator 10 and learning rate 0.3 :  0.929368029739777
Accuracy of gradient boosting of no. of estimator 10 and learning rate 0.5 :  0.9433800400343151
Accuracy of gradient boosting of no. of estimator 10 and learning rate 0.7 :  0.9433800400343151
Accuracy of gradient boosting of no. of estimator 50 and learning rate 0.1 :  0.9499571060909351
Accuracy of gradient boosting of no. of estimator 50 and learning rate 0.3 :  0.9636831569917071
Accuracy of gradient boosting of no. of estimator 50 and learning rate 0.5 :  0.7354875607663712
Accuracy of gradient boosting of no. of estimator 50 and learning rate 0.7 :  0.9559622533600228
Accuracy of gradient boosting of no. of estimator 100 and learning rate 0.1 :  0.9619674006291107
Accuracy of gradient boosting of no. of estimator 100 and learning rate 0.3 :  0.9645410351730054
Accuracy of gradient boosting of no. of estimator 100 and learning rate 0.5 :  0.7372033171289677
Accuracy of gradient boosting of no. of estimator 100 and learning rate 0.7 :  0.9573920503288533
Accuracy of gradient boosting of no. of estimator 200 and learning rate 0.1 :  0.9656848727480698
Accuracy of gradient boosting of no. of estimator 200 and learning rate 0.3 :  0.9651129539605376
Accuracy of gradient boosting of no. of estimator 200 and learning rate 0.5 :  0.694309408064055
Accuracy of gradient boosting of no. of estimator 200 and learning rate 0.7 :  0.1332570774949957
argmax:  12
```

- **The results showed that the number of estimator 200 and the learning rate of 0.1 is the best combination of these two hyperparameters.**
- **So, we build our model using these hyperparameters (no. of estimator = 200, learning rate =0.1).**
- **After that, we apply the accuracy score and confusion matrix of it.**

✓ **Accuracy of gradient boosting classifier: 96.6%**

```
[ ] gradient = GradientBoostingClassifier(n_estimators=200,learning_rate=0.1)
    gradient.fit(x_train,y_train)
    y_gb = gradient.predict(x_test)
    acc_gb = evaluation(y_test,y_gb,gradient)
    print(acc_gb)
```

```
0.9656848727480698
```



3.1.2. XGBoost:

- **XGBoost is an optimized Gradient Boosting Machine Learning library. It is originally written in C++ but has API in several other languages. The core XGBoost algorithm is parallelizable i.e., it does parallelization within a single tree. There are some of the cons of using XGBoost:**
  1. **It is one of the most powerful algorithms with high speed and performance.**
  2. **It can harness all the processing power of modern multicore computers.**
  3. **It is feasible to train on large datasets.**
  4. **Consistently outperforms all single algorithm methods.**

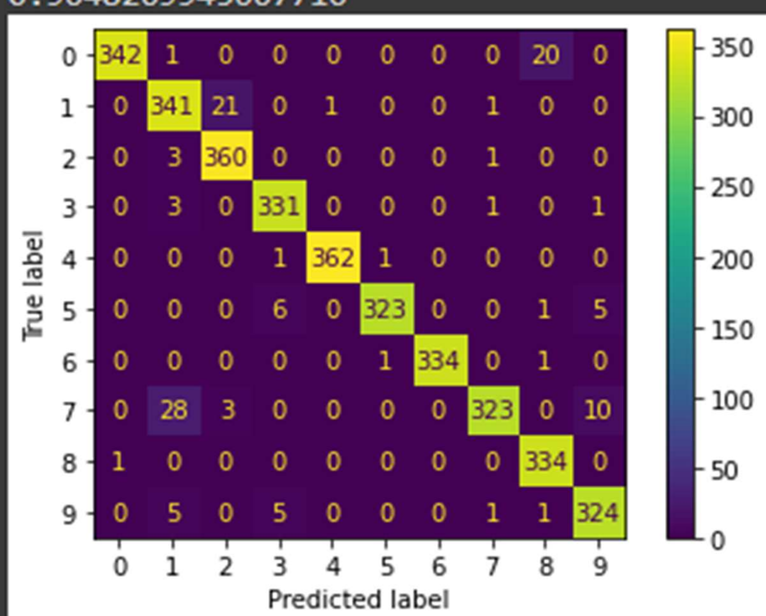
- **Implementation of XGBoost classifier:**
    - **First,** importing XGBClassifier from xgboost.
    - **Second,** building the model using the same hyperparameters that we used in the gradient boosting which are no. of estimator is 200, and learning rate is 0.1.
    - **Third,** we trained the model using (.fit) passed to it x_train, x_test.
    - **Fourth,** we tested the model using (.predict) passed to it x_test.
    - **Finally,** we apply the accuracy score and confusion matrix of it.

✓ **Accuracy of XGBoost: 96.5%**

XGBoost

```
xg = XGBClassifier(n_estimators=200,learning_rate=0.1)
xg.fit(x_train, y_train)
y_xg = xg.predict(x_test)
acc_xg = evaluation(y_test,y_xg,xg)
print(acc_xg)
```

```
0.9648269945667716
```



- **Comparison between gradient boosting and XGBoost:**

The two models produced a close accuracy score of 96.57% for gradient boosting and 96.48% for XGBoost. So, we can say the best is XGBoost because the time consumption in XGBoost is less than the time consumption in gradient boosting. Moreover, the XGBoost performs regularization which avoids the overfitting of the training set (the regularization we used in the model is 1, the default).

Conclusion

We had done bagging with SVM and Decision tree algorithms, to reduce decrease the variance and avoid overfitting. The accuracy we gathered from these two algorithms is 97.9% for SVM and 94.8% for decision tree. Then, we used a soft voting classifier and a hard voting classifier in seeking to improve the accuracy of our models, which the soft voting classifier was based on predicting the output by using the average of probability while the hard voting classifier was based on predicting the output by using the majority of votes. The accuracy we gathered from these two algorithms is 96% for the soft voting classifier and 97.6% for the hard voting classifier. After that, we try to build boosting algorithms based on combining several weak learners with poor accuracy to build a strong learner with higher accuracy. First, we tried boosting algorithm with a gradient boosting algorithm which technique fitting a new predictor to the residual errors made by the previous predictor. then, we made a hyperparameter tuning to find the best combinations between the learning rate and no of the estimator and we found the best combination is no. of the estimator is 200 and the learning rate is 0.1. the accuracy we gathered is 96.6%. After that, we have done the XGBoost algorithm which is an optimizer gradient boosting machine learning and we used the same hyperparameters that we used in the gradient boosting classifier. The accuracy we gathered is 96.5% the same as the gradient boosting algorithm but XGBoost succeeds the gradient boosting algorithm by regularization. Finally, the bagging algorithm perform well than boosting algorithms.