ELG 5255: Applied Machine Learning

Assignment 4

Group23_HW4

Part1: Calculations

1(a): build a decision tree by using Gini Index.

There are possible output variables **Yes** and **No**.

The data has **7 instances of No** and **3 instances of Yes.**

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

Step 1: Calculate the Total Gini index using this $Gini = 1 - \sum_{i=1}^{N_c}(p_i)^2$ **formula**

Gini(S) = 1 $- \left(\left(\frac{3}{10}\right)^2 + \left(\frac{7}{10}\right)^2\right)$ **= 0.42**

Step 2: Calculate the Gini index for feature 1 (Weather).

It has **3 possible outcomes**, **3 instances of Cloudy**, **4 instances of Sunny,** and **3 instances of Rainy.**

-Now we will calculate **Gini(Cloudy).**

Gini(Cloudy) = 1 $- \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right)$ **= 0.444**

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

-Now we will calculate **Gini(Sunny).**

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

-Now we will calculate **Gini(Rainy).**

Gini(Rainy) = $1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)$ **= 0.444**

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

Gini(Sunny)

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

Gini(Rainy)

-Now we will calculate the total Gini Index score for feature 1 **Gini(Weather).**

Gini(Weather) = 0.444 * $\left(\frac{3}{10}\right)$ **+ 0.375 *** $\left(\frac{4}{10}\right)$ **+ 0.444 *** $\left(\frac{3}{10}\right)$ **= 0.416**

Step 3: Calculate the Gini index for feature 2 (Temperature).

It has **3 possible outcomes**, **3 instances of Cool**, **3 instances of Hot,** and **4 instances of Mild.**

-Now we will calculate **Gini(Cool).**

Gini(Cool) = $1 - \left(\left(\frac{0}{3}\right)^2 + \left(\frac{3}{3}\right)^2\right)$ **= 0**

-Now we will calculate **Gini(Hot).**

Gini(Hot) = $1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right)$ **= 0.444**

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

Gini(Cool)

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

Gini(Hot)

-Now we will calculate **Gini(Mild).**

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

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

-Now we will calculate the total Gini Index score for feature 2 **Gini(Temperature).**

Gini(Temperature) = 0 * $\left(\frac{3}{10}\right)$ **+ 0.444 *** $\left(\frac{3}{10}\right)$ **+ 0.5 *** $\left(\frac{4}{10}\right)$ **= 0.333**

Step 4: Calculate the Gini index for feature 3 (Humidity).

It has **2 possible outcomes**, **4 instances of Normal**, **6 instances of High.**

-Now we will calculate **Gini(Normal).**

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

-Now we will calculate **Gini(High).**

Gini(High) = $1 - \left(\left(\frac{2}{6}\right)^2 + \left(\frac{4}{6}\right)^2\right)$ **= 0.444**

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

Gini(Normal)

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

Gini(High)

-Now we will calculate the total Gini Index score for feature 3 **Gini(Humidty).**

Gini(Humidty) = 0.375 * $\left(\frac{4}{10}\right)$ **+ 0.444 *** $\left(\frac{6}{10}\right)$ **= 0.416**

Step 5: Calculate the Gini index for feature 4 (Wind).

It has **2 possible outcomes**, **4 instances of Weak**, **6 instances of Strong.**

-Now we will calculate **Gini(Weak).**

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

-Now we will calculate **Gini(Strong).**

Gini(Strong) = $1-\left(\left(\frac{1}{6}\right)^2+\left(\frac{5}{6}\right)^2\right)$ **= 0.277**

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

Gini(Weak)

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

Gini(Strong)

-Now we will calculate the total Gini Index score for feature 4 **Gini(Wind).**

Gini(Wind) = 0.5 * $\left(\frac{4}{10}\right)$ **+ 0.277 *** $\left(\frac{6}{10}\right)$ **= 0.366**

Step 6: Now we will choose the root node based on the minimum value of Gini Index.

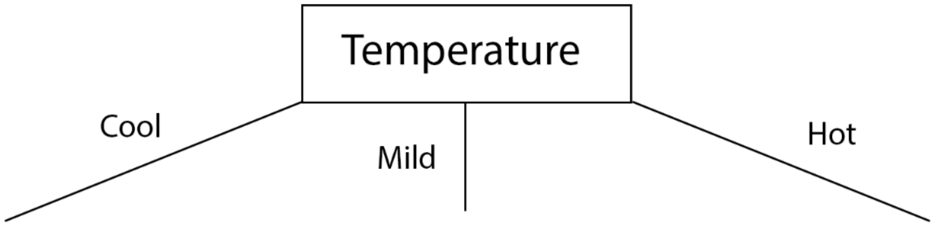
Gini(Weather) = 0.416

Gini(Temperature) = 0.333

Gini(Humidty) = 0.416

Gini(Wind) = 0.366

We found that the **Gini(Temperature)** was had the **minimum** value of Gini Index score, so we will take the temperature feature as the root node.



Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Cloudy	Cool	Normal	Weak	No
Rainy	Cool	Normal	Strong	No
Rainy	Cool	Normal	Weak	No

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

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

Step 7: now we will see the Gini index score with the other features.

-When Temperature = Cool

Gini(Temperature = Cool | Weather = Cloudy) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right)$ **= 0**

Gini(Temperature = Cool | Weather = Rainy) = 1 $- \left(\left(\frac{2}{2}\right)^2 + \left(\frac{0}{2}\right)^2\right)$ **= 0**

- **Total = Gini(Temperature = Cool | Weather) = 0 *** $\left(\frac{1}{3}\right)$ **+ 0 *** $\left(\frac{2}{3}\right)$ **= 0**

-When Temperature = Mild

Gini(Temperature = Mild | Weather = Rainy) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right)$ **= 0**

Gini(Temperature = Mild | Weather = Cloudy) = 1 $- \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$ **= 0.5**

Gini(Temperature = Mild | Weather = Sunny) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right)$ **= 0**

- **Total = Gini(Temperature = Mild | Weather) = 0 *** $\left(\frac{1}{4}\right)$ **+ 0.5 *** $\left(\frac{2}{4}\right)$ **+ 0 *** $\left(\frac{1}{4}\right)$ **= 0.25**

-When Temperature = Hot

Gini(Temperature = Hot | Weather = Sunny) = 1 $- \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right)$ **= 0.444**

- **Total = Gini(Temperature = Hot | Weather) = 0.444 *** $\left(\frac{3}{3}\right)$ **= 0.444**

So that's mean the when the temperature is Cool the Decision will be No.

Now we will compute the temperature with the other feature which is **Humidity**.

-When Temperature = Mild

Gini(Temperature = Mild | Humidity = Normal) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2 \right)$ **= 0**

Gini(Temperature = Mild | Humidity = High) = 1 $- \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right)$ **= 0.444**

- **Total = Gini(Temperature = Mild | Humidity) = 0 *** $\left(\frac{1}{4}\right)$ **+ 0.444 *** $\left(\frac{3}{4}\right)$ **= 0.333**

-When Temperature = Hot

Gini(Temperature = Hot | Humidity = High) = 1 $- \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right)$ **= 0.444**

- **Total = Gini(Temperature = Hot | Humidity) = 0.444 *** $\left(\frac{3}{3}\right)$ **= 0.444**

Now we will compute the temperature with the other feature which is **Wind**.

-When Temperature = Mild

Gini(Temperature = Mild | Wind = Strong) = 1 $- \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2 \right)$ **= 0.444**

Gini(Temperature = Mild | Wind = Weak) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2 \right)$ **= 0**

- **Total = Gini(Temperature = Mild | Wind) = 0.444 *** $\left(\frac{3}{4}\right)$ **+ 0 *** $\left(\frac{1}{4}\right)$ **= 0.333**

-When Temperature = Hot

Gini(Temperature = Hot | Wind = Strong) = 1 $- \left(\left(\frac{2}{2}\right)^2 + \left(\frac{0}{2}\right)^2 \right)$ **= 0**

Gini(Temperature = Hot | Wind = Weak) = 1 $- \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2 \right)$ **= 0**

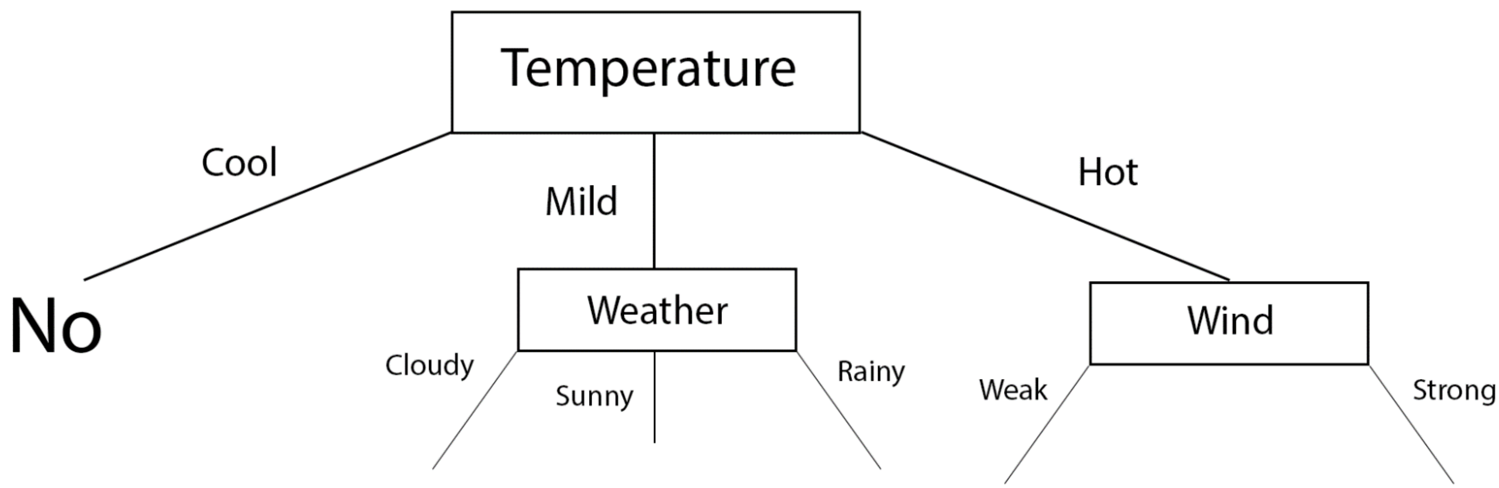
- **Total = Gini(Temperature = Hot | Wind) = 0 *** $\left(\frac{2}{3}\right)$ **+ 0 *** $\left(\frac{1}{3}\right)$ **= 0**

Step 8: From these scores we can start to build the tree.

1- When the **temperature is Mild** we will see which feature has the minimum value of Gini index score.
   Gini(Temperature = Mild | Weather) = 0.25
   Gini(Temperature = Mild | Humidity) = 0.333
   Gini(Temperature = Mild | Wind) = 0.333
   so, we will take feature (**Weather**) with the temperate = **Mild**.

2- When the **temperature is Hot** we will see which feature has the minimum value of Gini index score.
   Gini(Temperature = Hot | Weather) = 0.444
   Gini(Temperature = Hot | Humidity) = 0.444
   Gini(Temperature = Hot | Wind) = 0
   so, we will take feature (**Wind**) with the temperate when it's = **Hot**.

So, until now we have something like this.



Step 9: Complete the Tree.

So now we will see what is the result of each possibility.

We notice from the table, when the temperature is **Mild**,

And the weather is **Rainy** we found that the **only** hiking option is **YES.**

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

And the same for weather is **Sunny** we found that the **only** hiking option is **NO.**

But we found that there are **and impurity when weather is Cloudy** (one time hiking option was **NO** and the other time it was **Yes**)

So now we will calculate the Gini Index score for…

Gini(Temperature = Mild | Weather = Cloudy | Humidity) = $1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$ = 0.5

- **Total = Gini(Temperature = Mild | Weather = Cloudy | Humidity) = 0.5 * $\left(\frac{2}{2}\right)$ = 0.5**

Gini(Temperature = Mild | Weather = Cloudy | Wind = Strong) = $1 - \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right)$ = 0

Gini(Temperature = Mild | Weather = Cloudy | Wind = Weak) = $1 - \left(\left(\frac{1}{1}\right)^2 + \left(\frac{0}{1}\right)^2\right)$ = 0

- **Total = Gini(Temperature = Mild | Weather = Cloudy | Wind) = 0 * $\left(\frac{1}{2}\right)$ + 0 * $\left(\frac{1}{2}\right)$ = 0**

We found that **Gini(Temperature = Mild | Weather = Cloudy | Wind)** has the minimum Gini index score.

Gini(Temperature = Mild | Weather = Cloudy | Humidity) = 0.5

Gini(Temperature = Mild | Weather = Cloudy | Wind) = 0

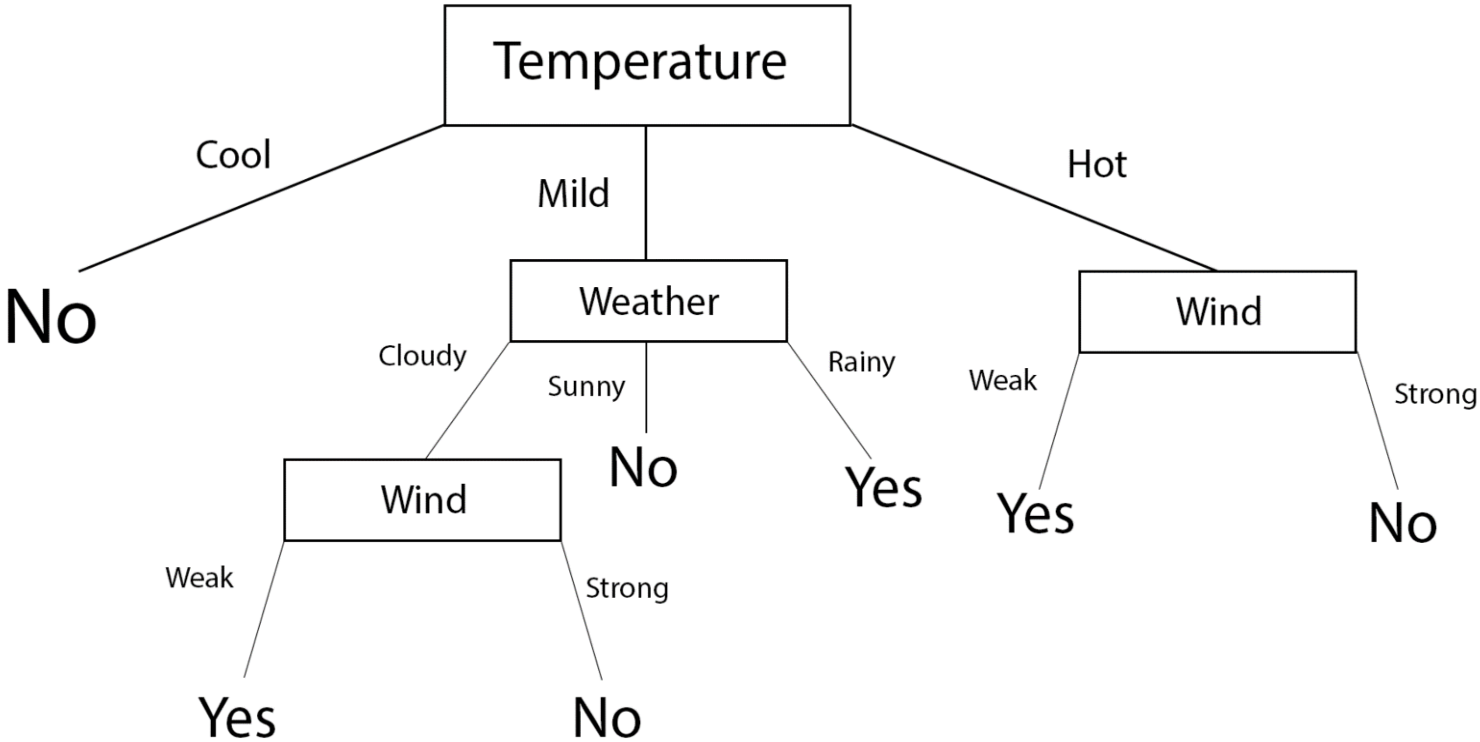
So now when the **Temperature = Mild** and the **Weather = Cloudy,** we will check for **Wind** Value if it was = **strong** it will be **No,** and if it was = **Weak** it will be **Yes, based on that table on the below. (There is no Impurity).**

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

So now when the **Temperature = Hot**, we will check for **Wind** Value if it was = **strong** it will be **No,** and if it was = **Weak** it will be **Yes, based on that table on the below (There is no Impurity).**

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

So we have reached to the leaf node **(Final Decision)** on each branch, and we got something like this…



1(b): build a decision tree by using 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

Step 1: Calculate the Entropy(S) using this formula… $Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$

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

P(Yes) = $\frac{3}{10}$

Entropy(S) = $-\frac{7}{10} * \log_2 \frac{7}{10} - \frac{3}{10} * \log_2 \frac{3}{10}$ = 0.88129

Step 2: Calculate the Information Gain Score for each feature using this formula…

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

- **Weather**

3 Cloudy -> 2 No, 1 Yes.

4 Sunny -> 3 No, 1 Yes.

3 Rainy -> 2 No, 1 Yes.

Gain(S, Weather) = 0.88129 $- \frac{3}{10} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} * \log_2 \frac{1}{3}\right) - \frac{4}{10} * \left(-\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} * \log_2 \frac{1}{4}\right) - \frac{3}{10} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} * \log_2 \frac{1}{3}\right)$ = 0.4982

- **Temperature**

3 Cool -> 3 No, 0 Yes.

3 Hot -> 2 No, 1 Yes.

4 Mild -> 2 No, 2 Yes.

Gain(S, Temperature) = 0.88129 $- \frac{3}{10} * \left(-\frac{3}{3} \log_2 \frac{3}{3} - \frac{0}{3} * \log_2 \frac{0}{3}\right) - \frac{3}{10} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} * \log_2 \frac{1}{3}\right) - \frac{4}{10} * \left(-\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} * \log_2 \frac{2}{4}\right)$ = 0.82279

- **Humidity**

4 Normal -> 3 No, 1 Yes.

6 High -> 4 No, 2 Yes.

Gain(S, Humidity) = 0.88129 $-\frac{4}{10}*\left(-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}*\log_2\frac{1}{4}\right)-\frac{6}{10}*\left(-\frac{4}{6}\log_2\frac{4}{6}-\frac{2}{6}*\log_2\frac{2}{6}\right)$ **= 0.00580**

- **Wind**

4 Weak -> 2 No, 2 Yes.

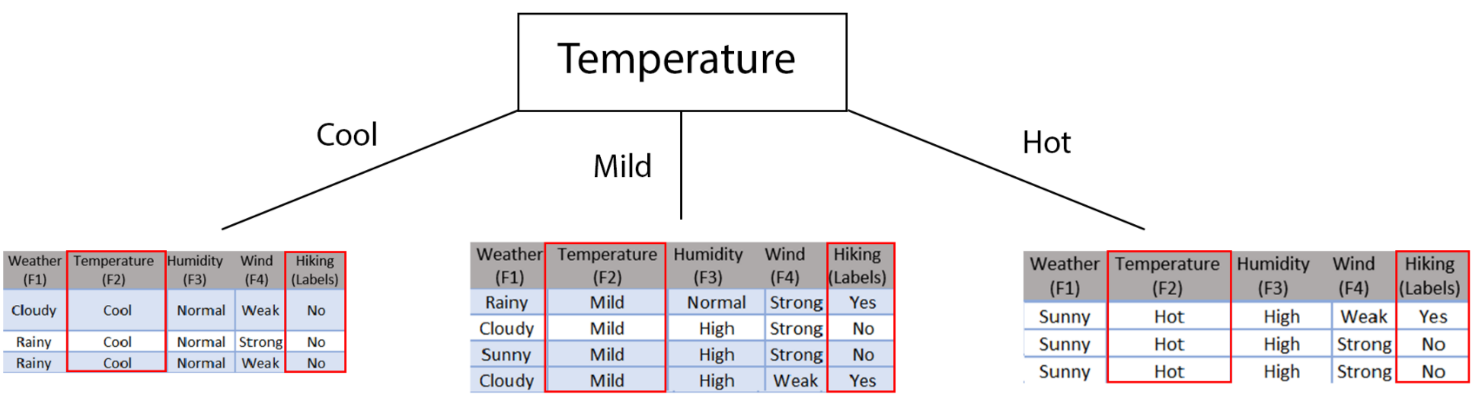
6 Strong -> 5 No, 1 Yes.

Gain(S, Wind) = 0.88129 $-\frac{4}{10}*\left(-\frac{2}{4}\log_2\frac{2}{4}-\frac{2}{4}*\log_2\frac{2}{4}\right)-\frac{6}{10}*\left(-\frac{5}{6}\log_2\frac{5}{6}-\frac{1}{6}*\log_2\frac{1}{6}\right)$ **= 0.09127**

Step 3: Determine which feature will be the root node, and after that calculate the IG Score for the other features.

We found that the **Temperature has the highest value of information**, so it will be the root node.

Gain(S, Temperature) = 0.82279



Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Labels)
Cloudy	Cool	Normal	Weak	No
Rainy	Cool	Normal	Strong	No
Rainy	Cool	Normal	Weak	No

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

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

from the above figure we will found that when **Temperature = Cool**, the **decision will be No.**

now we will calculate the other feature when **Temperature = Mild,** but first we will calculate the **new Entropy.**

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

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

- **Weather**

1 Rainy -> 0 No, 1 Yes.

2 Cloudy -> 1 No, 1 Yes.

1 Sunny -> 1 No, 0 Yes.

Gain(S, Weather) = 1 $-\frac{1}{4}*\left(-\frac{1}{1}\log_2\frac{1}{1}\right)-\frac{2}{4}*\left(-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}*\log_2\frac{1}{2}\right)-\frac{1}{4}*\left(-\frac{1}{1}\log_2\frac{1}{1}\right)$ **= 0.5**

- **Humidity**

1 Normal -> 0 No, 1 Yes.

3 High -> 2 No, 1 Yes.

Gain(S, Humidity) = 1 $-\frac{1}{4}*\left(-\frac{1}{1}\log_2\frac{1}{1}\right)-\frac{3}{4}*\left(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}*\log_2\frac{1}{3}\right)$ **= 0.311278**

- **Wind**

3 Strong -> 2 No, 1 Yes.

1 Weak -> 0 No, 1 Yes.

Gain(S, Wind) = 1 $-\frac{3}{4}*\left(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{1}{3}\right)-\frac{1}{4}*\left(-\frac{1}{1}\log_2\frac{1}{1}\right)$ **= 0.311278**

We found that the **Weather has the highest value of information**, When **Temperature = Mild.**

Gain(S, Weather) = 0.5

now we will calculate the other feature when **Temperature = Hot,** but first we will calculate the **new Entropy.**

Entropy(S) = $-\frac{1}{3} * \log_2 \frac{1}{3} - \frac{2}{3} * \log_2 \frac{2}{3}$ **= 0.9182**

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

- **Weather**

3 Sunny -> 2 No, 1 Yes.

Gain(S, Weather) = 0.9182 $-\frac{3}{3} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}\right)$ **= 0**

- **Humidity**

3 High -> 2 No, 1 Yes.

Gain(S, Humidity) = 0.9182 $-\frac{3}{3} * \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3}\right)$ **= 0**

- **Wind**

2 Strong -> 2 No, 0 Yes.

1 Weak -> 0 No, 1 Yes.

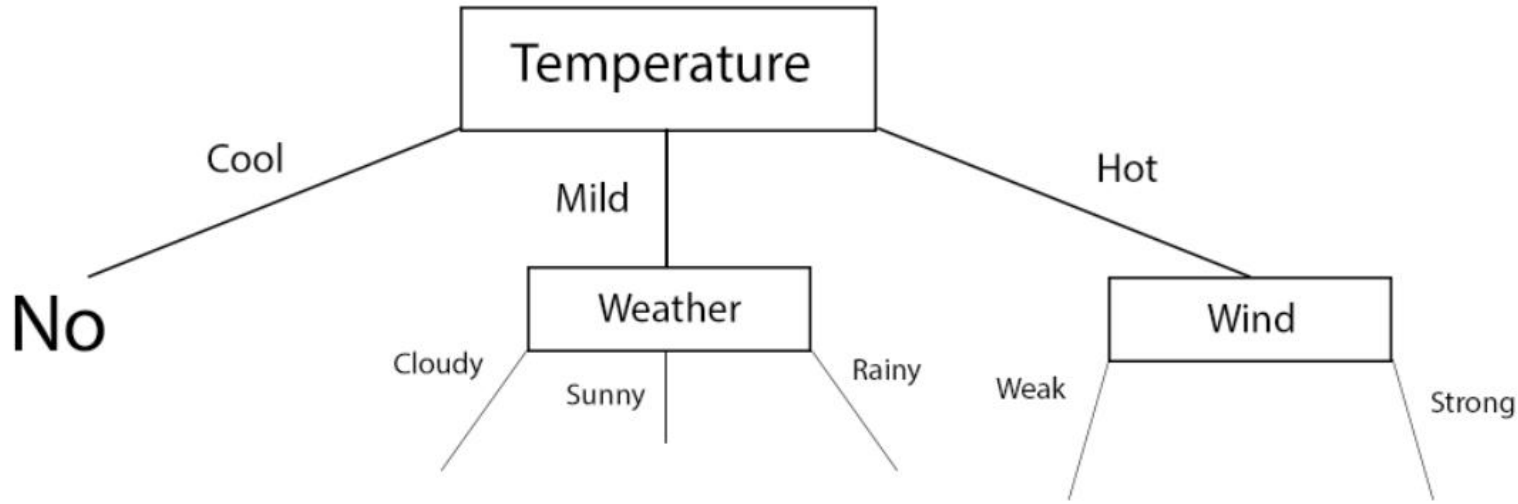
Gain(S, Wind) = 0.9182 $-\frac{2}{3} * \left(-\frac{2}{2} \log_2 \frac{2}{2}\right) - \frac{1}{3} * \left(-\frac{1}{1} \log_2 \frac{1}{1}\right)$ **= 0.9182**

We found that the **Wind has the highest value of information**, When **Temperature = Hot.**

Gain(S, Wind) = 0.9182

Step 4: continue constructing the tree...

Now we have something like this...



So now we will see what is the result of each possibility.

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

We notice from the table, when the temperature is **Mild,**

And the weather is **Rainy** we found that the **only** hiking option is **YES.**

And the same for weather is **Sunny** we found that the **only** hiking option is **NO.**

But we found that there are **and impurity when weather is Cloudy** (one time hiking option was **NO** and the other time it was **Yes**)

Now we will calculate the other feature when **Temperature = Mild** and **Weather = Cloudy,** but first we will calculate the **new Entropy.**

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

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

- **Humidity**

2 High -> 1 No, 1 Yes.

Gain(S, Humidity) = 1 $-\frac{2}{2} * \left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)$ **= 0**

- **Wind**

1 Strong -> 1 No, 0 Yes.

1 Weak -> 0 No, 1 Yes.

Gain(S, Wind) = 1 $-\frac{1}{2} * \left(-\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{1}{2} * \left(-\frac{1}{1}\log_2\frac{1}{1}\right)$ **= 1**

We found that the **Wind has the highest value of information**, When **Temperature = Mild** and **Weather = Cloudy.**

Gain(S, Wind) = 1

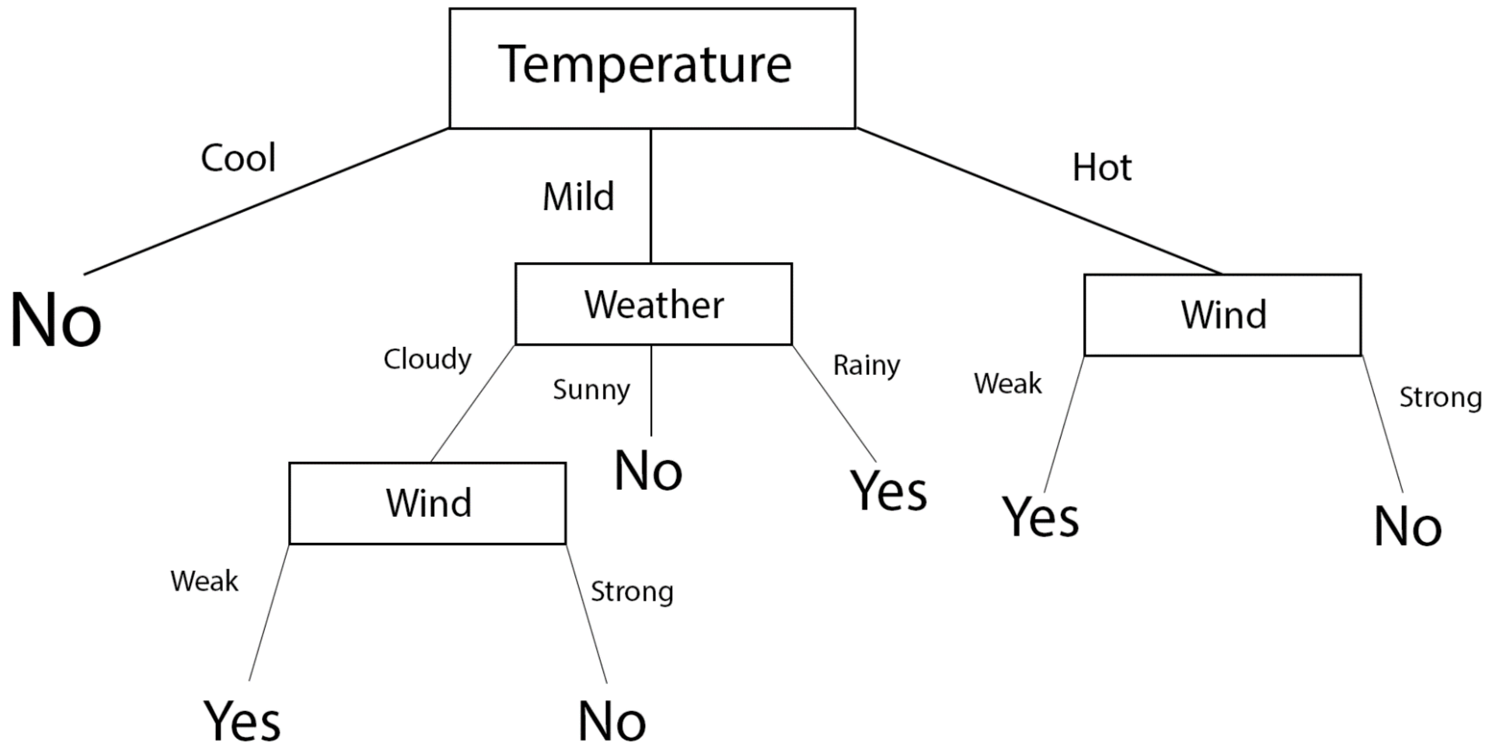
So now when the **Temperature = Mild** and the **Weather = Cloudy,** we will check for **Wind** Value if it was = **strong** it will be **No,** and if it was = **Weak** it will be **Yes, based on that table on the below. (There is no Impurity).**

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

So now when the **Temperature = Hot**, we will check for **Wind** Value if it was = **strong** it will be **No,** and if it was = **Weak** it will be **Yes, based on that table on the below (There is no Impurity).**

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

So we have reached to the leaf node **(Final Decision)** on each branch, and we got something like this…



1(c): We have seen that both ways gave us the same Tree at the end, but talk about each method from the perspective of computational power, the Gini index will win because…

- Gini index it only goes up to 0.5 and then it starts decreasing, hence it requires less computational power.
- But The range of Entropy (information gain) lies in between 0 to 1 and the range.

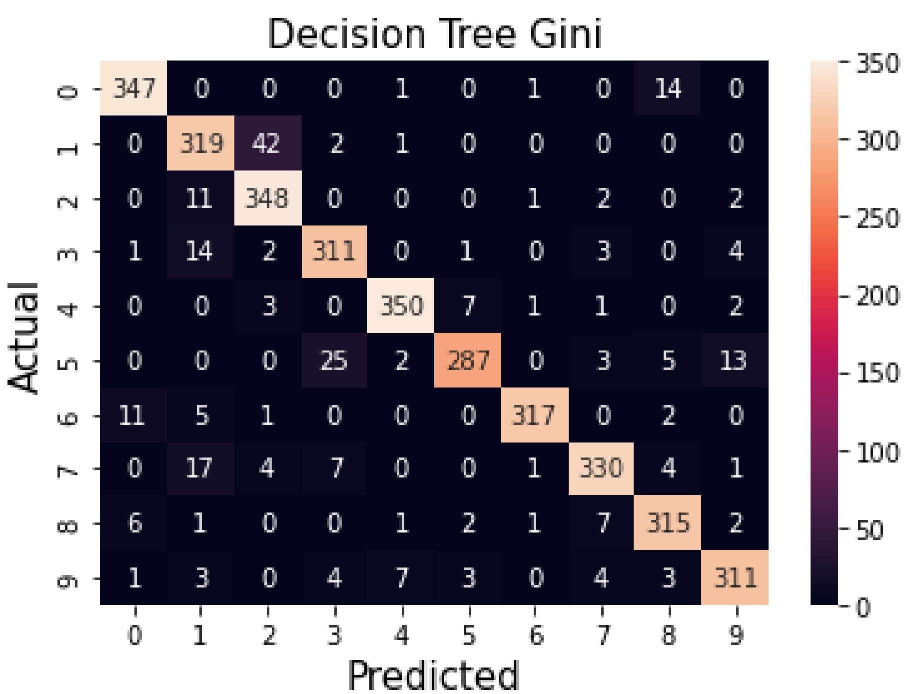
Hence, we can conclude that Gini Impurity is better as compared to entropy (information gain) for selecting the best features.

Part2: Programming

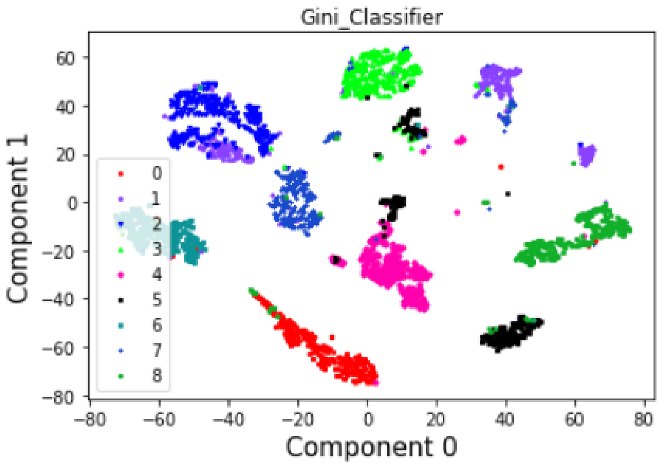
2(a): we have tried both Decision tree with **Gini Index** method and **Entropy** method, and we found that Gini Index gave better accuracy…

And after that we have plot the different **predicted classes** with different colors.

```
In [33]: print(Gini_report)

              precision    recall  f1-score   support

           0       0.95      0.96      0.95       363
           1       0.86      0.88      0.87       364
           2       0.87      0.96      0.91       364
           3       0.89      0.93      0.91       336
           4       0.97      0.96      0.96       364
           5       0.96      0.86      0.90       335
           6       0.98      0.94      0.96       336
           7       0.94      0.91      0.92       364
           8       0.92      0.94      0.93       335
           9       0.93      0.93      0.93       336

    accuracy                           0.93      3497
   macro avg       0.93      0.92      0.93      3497
weighted avg       0.93      0.93      0.93      3497
```



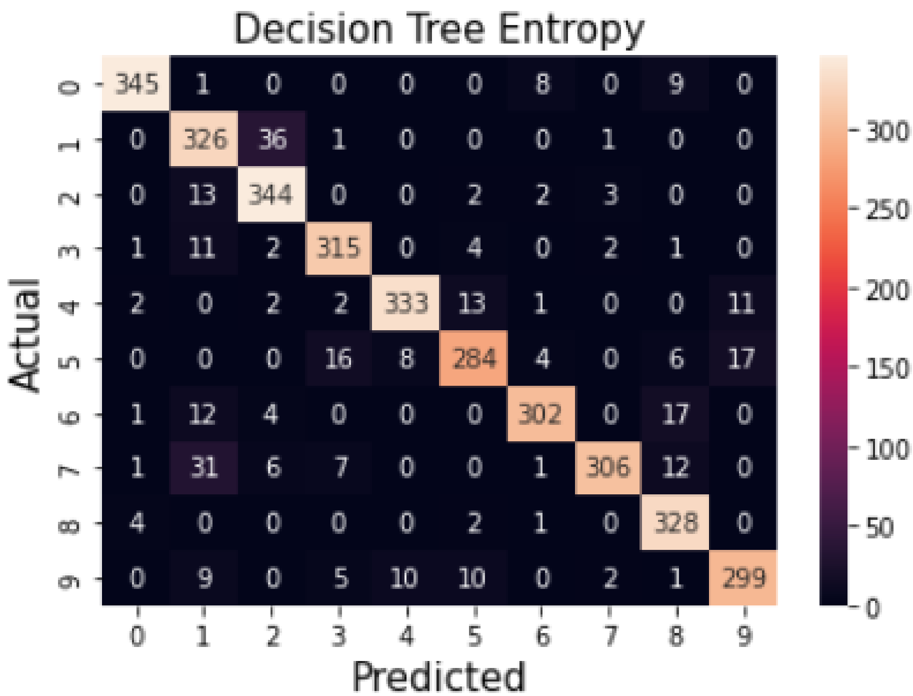
```
In [15]: # Plot Gini
XTsne = pd.concat([pd.DataFrame(XTsne), pd.DataFrame(Gini_Ypred)],axis=1 , ignore_index = True).astype(float)
GiniLs = GetListOfClasses(9, XTsne,  pd.DataFrame(XTsne).columns[2])
Labels = ['0','1','2','3','4','5','6','7','8']
PlotDataPoints(9, GiniLs, 'Component 0', 'Component 1' ,Labels ,5, 'Gini_Classifier').show()
XTsne = T_SNE(XTest)
```



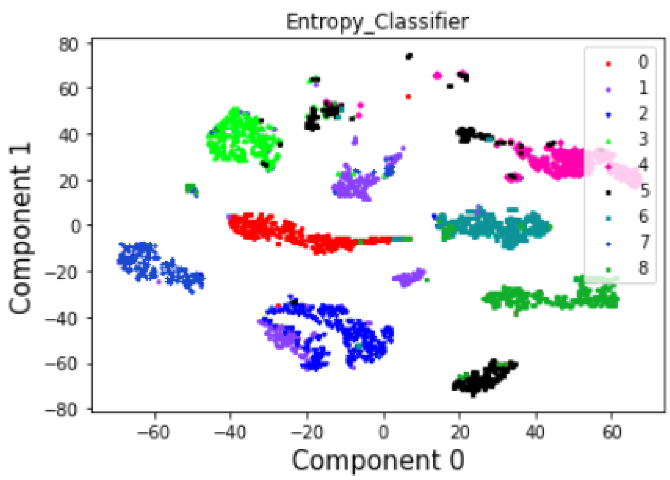
- **Entropy**

```
In [34]: print(Ent_report)
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       363
           1       0.81      0.90      0.85       364
           2       0.87      0.95      0.91       364
           3       0.91      0.94      0.92       336
           4       0.95      0.91      0.93       364
           5       0.90      0.85      0.87       335
           6       0.95      0.90      0.92       336
           7       0.97      0.84      0.90       364
           8       0.88      0.98      0.93       335
           9       0.91      0.89      0.90       336

    accuracy                           0.91      3497
   macro avg       0.91      0.91      0.91      3497
weighted avg       0.91      0.91      0.91      3497
```



In [17]:
```
# Plot Entropy
XTsne = pd.concat([pd.DataFrame(XTsne), pd.DataFrame(Ent_Ypred)],axis=1 , ignore_index = True).astype(float)
EntLs = GetListOfClasses(9, XTsne,  pd.DataFrame(XTsne).columns[2])
PlotDataPoints(9, EntLs, 'Component 0', 'Component 1' ,Labels ,5, 'Entropy_Classifier').show()
```



3(a): here we applied **BaggingClassifier(base_estimator=SVC(), n_estimators=(we have tries 1 and 2))**

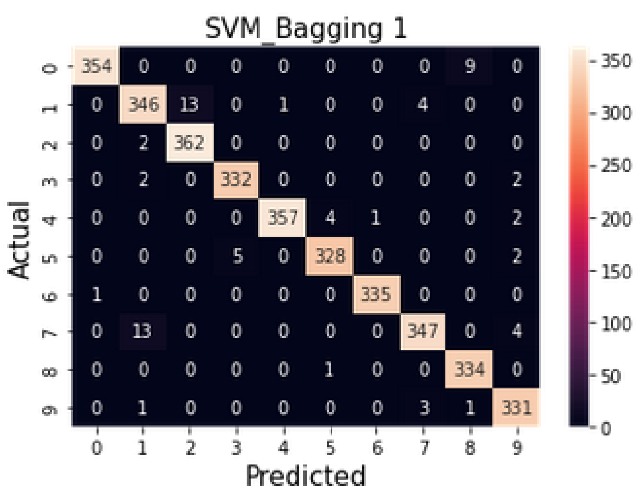
so, the frist time the **n_estimators was = 1** and the second time, **n_estimators was = 2.**

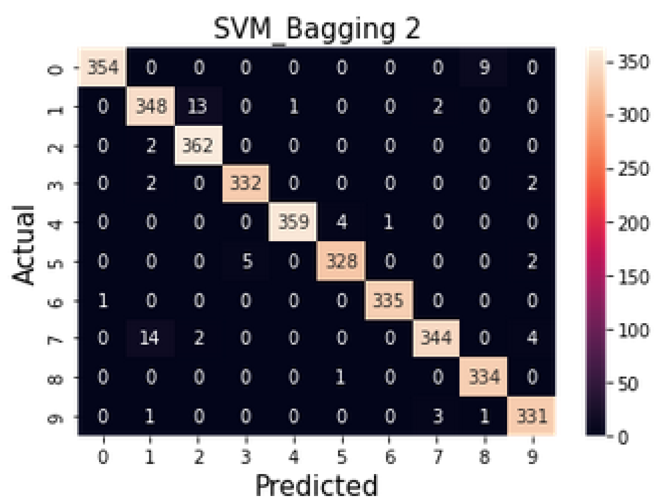
SVM Bagging: 1

	precision	recall	f1-score	support
0	1.00	0.98	0.99	363
1	0.95	0.95	0.95	364
2	0.97	0.99	0.98	364
3	0.99	0.99	0.99	336
4	1.00	0.98	0.99	364
5	0.98	0.98	0.98	335
6	1.00	1.00	1.00	336
7	0.98	0.95	0.97	364
8	0.97	1.00	0.98	335
9	0.97	0.99	0.98	336
accuracy			0.98	3497
macro avg	0.98	0.98	0.98	3497
weighted avg	0.98	0.98	0.98	3497

SVM Bagging: 2

	precision	recall	f1-score	support
0	1.00	0.98	0.99	363
1	0.95	0.96	0.95	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	0.96	364
8	0.97	1.00	0.98	335
9	0.98	0.99	0.98	336
accuracy			0.98	3497
macro avg	0.98	0.98	0.98	3497
weighted avg	0.98	0.98	0.98	3497

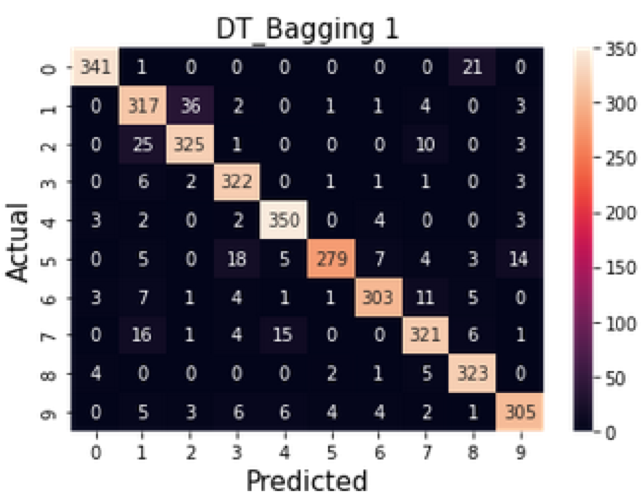


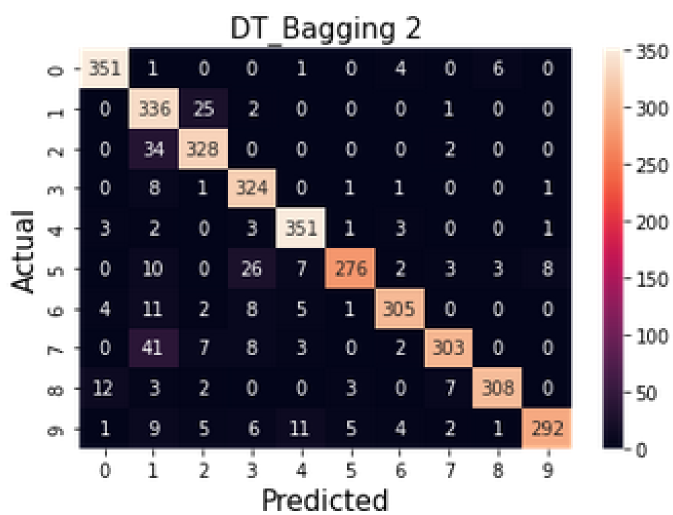


- here we did the same thing but we have changed the base estimator to … **BaggingClassifier(base_estimator= DecisionTreeClassifier(), n_estimators=(we have tries 1 and 2))**
  so, the frist time the **n_estimators was = 1** and the second time, **n_estimators was = 2.**

```
            Decision Tree Bagging:  1
           precision    recall  f1-score   support

           0       0.97      0.94      0.96       363
           1       0.83      0.87      0.85       364
           2       0.88      0.89      0.89       364
           3       0.90      0.96      0.93       336
           4       0.93      0.96      0.94       364
           5       0.97      0.83      0.90       335
           6       0.94      0.90      0.92       336
           7       0.90      0.88      0.89       364
           8       0.90      0.96      0.93       335
           9       0.92      0.91      0.91       336

    accuracy                           0.91      3497
   macro avg       0.91      0.91      0.91      3497
weighted avg       0.91      0.91      0.91      3497
```

```
            Decision Tree Bagging:  2
           precision    recall  f1-score   support

           0       0.95      0.97      0.96       363
           1       0.74      0.92      0.82       364
           2       0.89      0.90      0.89       364
           3       0.86      0.96      0.91       336
           4       0.93      0.96      0.95       364
           5       0.96      0.82      0.89       335
           6       0.95      0.91      0.93       336
           7       0.95      0.83      0.89       364
           8       0.97      0.92      0.94       335
           9       0.97      0.87      0.92       336

    accuracy                           0.91      3497
   macro avg       0.92      0.91      0.91      3497
weighted avg       0.91      0.91      0.91      3497
```



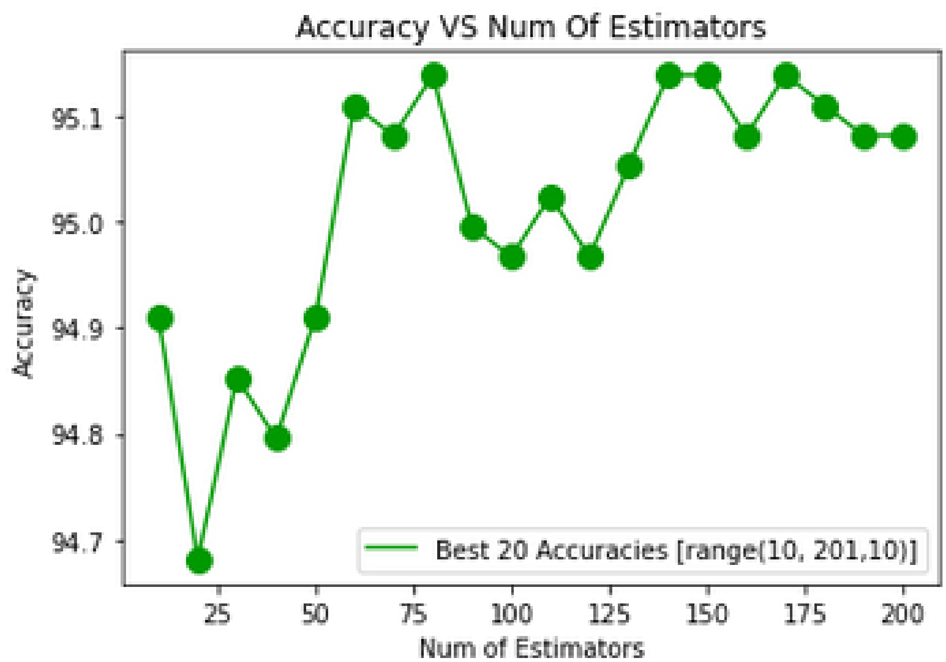


3(b): to find the best number of estimators we have used a for loop to iterate on this range [10,200] to try as many options as we can, so we go for range(10,201,10), that's mean that we will have 20 different accuracies, and after that we have sorted these accuracies to know which n_estimators values will give use the highest accuracies.

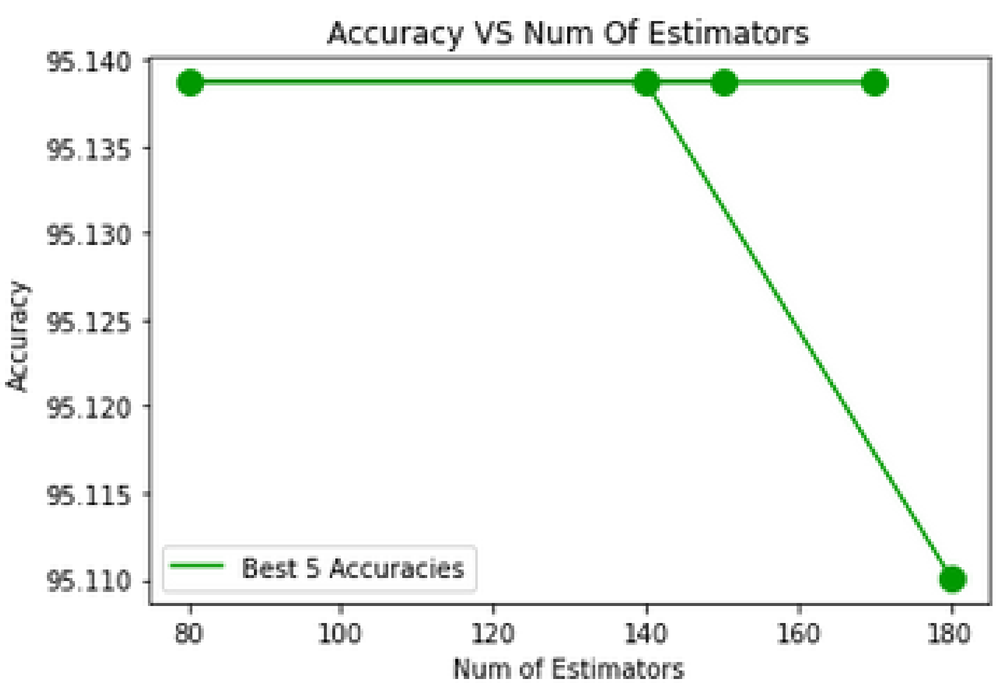
```
In [20]: # 3(b) -------------------------------------- #Bagging
# Decision Tree
DT_Acc = []
nofEst = []
for numOfEst in range(10, 201,10):
  DT_estimator = BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=numOfEst, random_state=0).fit(XTrain,
  DT_Ypred = DT_estimator.predict(XTest)
  DT_Acc.append(AccuracyTest(Ytest, DT_Ypred))
  nofEst.append(numOfEst)

EstandAcc = pd.concat([pd.DataFrame(DT_Acc), pd.DataFrame(nofEst)],axis=1 , ignore_index = True).astype(float)
EstandAcc = EstandAcc.sort_values(by=[0],ascending=False)
```

- **Best values for n_estimators that gave us the highest accurices.**

```
In [35]: print(EstandAcc)

            0      1
16  95.138690  170.0
7   95.138690   80.0
14  95.138690  150.0
13  95.138690  140.0
17  95.110094  180.0
5   95.110094   60.0
19  95.081498  200.0
18  95.081498  190.0
6   95.081498   70.0
15  95.081498  160.0
12  95.052902  130.0
10  95.024307  110.0
8   94.995711   90.0
11  94.967115  120.0
9   94.967115  100.0
4   94.909923   50.0
0   94.909923   10.0
2   94.852731   30.0
3   94.795539   40.0
1   94.681155   20.0
```



- **Accuracies VS n_estimators (Full DataFrame)**



- **Accuracies VS n_estimators (Best 5 (4 of them are have the same Accuracy))**

4(a): here we did the same idea like **3(b) to tune the n_estimators values from this range [10,200].**

and we found that the best 4 values for **n_estimators is [200,160,150,140].**

In [23]:
```
# 4(a) ------------------------------------- #Boosting
# Tuning the number of estimators Parameter
Boosting_Acc = []
numOfEst = []
for i in range(10,201,10):
   Boosting_estimator = GradientBoostingClassifier(n_estimators=i, random_state=0).fit(XTrain, YTrain)
   Boosting_Ypred = Boosting_estimator.predict(XTest)
   Boosting_Acc.append(AccuracyTest(Ytest, Boosting_Ypred))
   numOfEst.append(i)

Boosting_Est = pd.concat([pd.DataFrame(Boosting_Acc), pd.DataFrame(numOfEst)],axis=1 , ignore_index = True).astype(float)
Boosting_Est = Boosting_Est.sort_values(by=[0],ascending=False)
print(Boosting_Est.iloc[:4,1])
```

```
19    200.0
15    160.0
14    150.0
13    140.0
Name: 1, dtype: float64
```

- here we did the same idea like **3(b) to tune the learning_rate values from this range [0.1 -> 0.9].**
  and we found that the best 4 values for **is [0.2,0.3,0.1,0.7].**

In [24]:
```
# Tuning learning rate parameter
Lr_rate = np.array([0.1,0.2,0.3, 0.4, 0.5, 0.6,0.7,0.8,0.9])
Boosting_Acc = []
Lr = []
for i in Lr_rate:
   Boosting_estimator = GradientBoostingClassifier(learning_rate=i, random_state=0).fit(XTrain, YTrain)
   Boosting_Ypred = Boosting_estimator.predict(XTest)
   Boosting_Acc.append(AccuracyTest(Ytest, Boosting_Ypred))
   Lr.append(i)

Boosting_Lr = pd.concat([pd.DataFrame(Boosting_Acc), pd.DataFrame(Lr)],axis=1 , ignore_index = True).astype(float)
Boosting_Lr = Boosting_Lr.sort_values(by=[0],ascending=False)
print(Boosting_Lr.iloc[:4,1])
```

```
1    0.2
2    0.3
0    0.1
6    0.7
Name: 1, dtype: float64
```

- This for loop for finding the best combination between the best values of the parameters.

```
In [25]: # Train GradientBoostingClassifier
Boosting_Acc = []
est = []
lr = []
estLS = list(Boosting_Est.iloc[:4,1])
LrLS = list(Boosting_Lr.iloc[:4,1])

for i in range(len(Boosting_Est.iloc[:4,1])):
    for j in range(len(Boosting_Lr.iloc[:4,1])):
        Boosting_estimator = GradientBoostingClassifier(n_estimators=trunc(estLS[i]),learning_rate=LrLS[j], random_state=0).fit(
        Boosting_Ypred = Boosting_estimator.predict(XTest)
        Boosting_Acc.append(AccuracyTest(Ytest, Boosting_Ypred))
        print('Done')
        est.append(trunc(estLS[i]))
        lr.append(LrLS[j])

Boosting = pd.concat([pd.DataFrame(Boosting_Acc), pd.DataFrame(est), pd.DataFrame(lr)],axis=1 , ignore_index = True).astype(flo
Boosting = Boosting.sort_values(by=[0],ascending=False)
print(Boosting.iloc[:4])
```

	0	1	2
2	96.568487	200.0	0.1
1	96.511295	200.0	0.3
6	96.482699	160.0	0.1
10	96.482699	150.0	0.1

- **Best combination of the parameters together that gave the highest accuracy.**

We found that the best combinations are…

Learning_rate = [0.1,0.3]

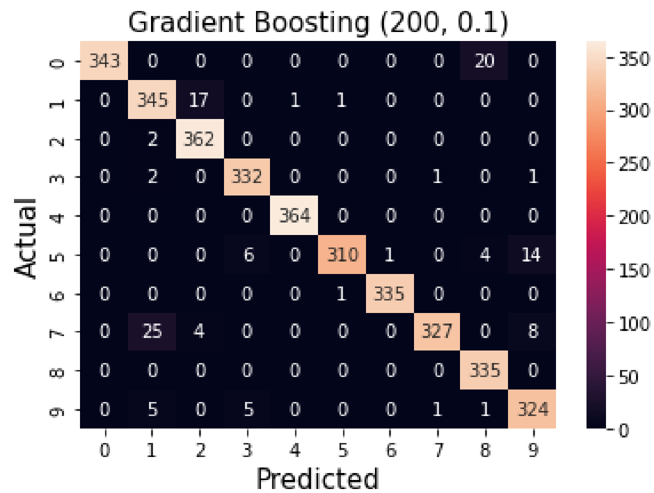
n_estimators = [160,150,200]

with these combinations we have trin gradient boost again and we have to obtain **6 different Confusion matrices,** and **6 different classification_reports.**

```
for i in range(len(best_estLS)):
    for j in range(len(best_LrLS)):
        Boosting_estimator = GradientBoostingClassifier(n_estimators=trunc(best_estLS[i]),learning_rate=best_LrLS[j], random_sta
        Boosting_Ypred = Boosting_estimator.predict(XTest)

        GB_report = classification_report(Ytest, Boosting_Ypred)
        GB_reports.append(DT_report)
        print('\t\tGradient Boosting Best 6 Accuracies:',"\tNum of Est = ",trunc(best_estLS[i]), '\tLearning Rate = ',best_LrLS[
        GB_cf = ConfusionMatrix(Ytest, Boosting_Ypred)
        PLOT_ConfusionMatrix(GB_cf, f'Gradient Boosting {trunc(best_estLS[i]),best_LrLS[j]}')
        GB_Acc.append(AccuracyTest(Ytest, Boosting_Ypred))
        GBlr.append(best_LrLS[j])
        GBEst.append(trunc(best_estLS[i]))
        print('Done')
```

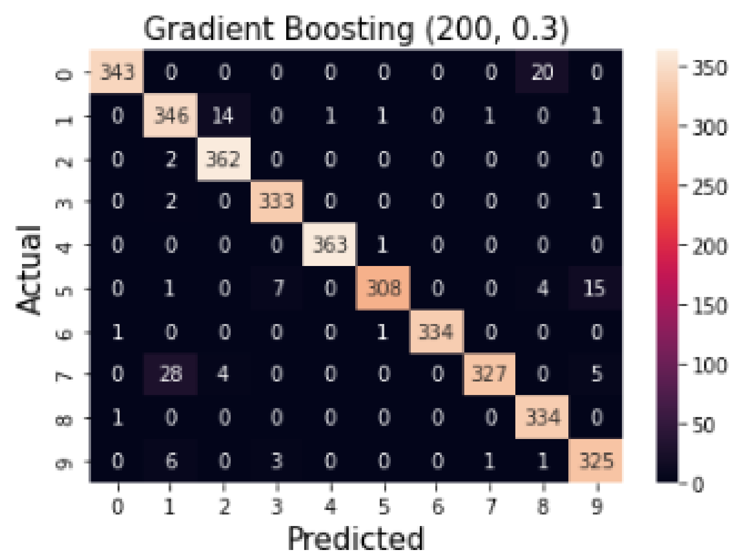
- **First model (num of est = 200, lr_rate = 0.1)**

```
            Gradient Boosting Best 6 Accuracies:    Num of Est =  200       Learning Rate =  0.1
              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      1.00      1.00       336
           7       0.99      0.90      0.94       364
           8       0.93      1.00      0.96       335
           9       0.93      0.96      0.95       336

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



- **Second model (num of est = 200, lr_rate = 0.3)**

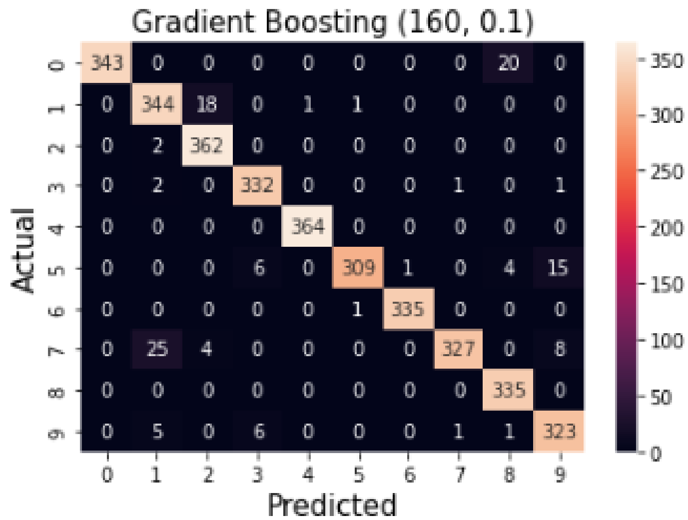
```
            Gradient Boosting Best 6 Accuracies:    Num of Est =  200       Learning Rate =  0.3
              precision    recall  f1-score   support

           0       0.99      0.94      0.97       363
           1       0.90      0.95      0.92       364
           2       0.95      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      1.00      1.00       364
           5       0.99      0.92      0.95       335
           6       1.00      0.99      1.00       336
           7       0.99      0.90      0.94       364
           8       0.93      1.00      0.96       335
           9       0.94      0.97      0.95       336

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



- **Third model (num of est = 160, lr_rate = 0.1)**

Gradient Boosting Best 6 Accuracies: Num of Est = 160 Learning Rate = 0.1

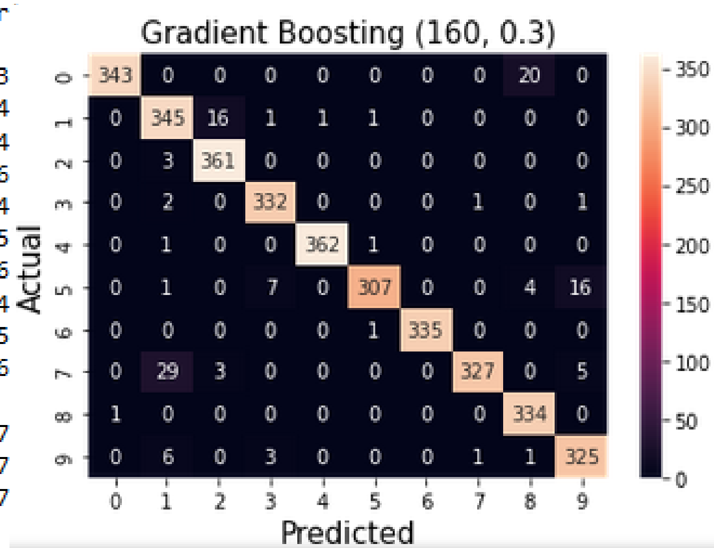
	precision	recall	f1-score	support
0	1.00	0.94	0.97	363
1	0.91	0.95	0.93	364
2	0.94	0.99	0.97	364
3	0.97	0.99	0.98	336
4	1.00	1.00	1.00	364
5	0.99	0.92	0.96	335
6	1.00	1.00	1.00	336
7	0.99	0.90	0.94	364
8	0.93	1.00	0.96	335
9	0.93	0.96	0.95	336
accuracy			0.96	3497
macro avg	0.97	0.97	0.96	3497
weighted avg	0.97	0.96	0.96	3497



- **Fourth model (num of est = 160 , lr_rate = 0.3)**

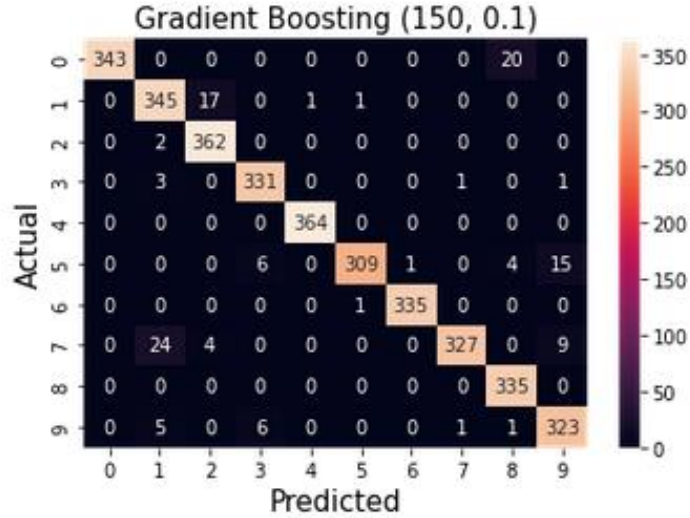
Gradient Boosting Best 6 Accuracies: Num of Est = 160 Learning Rate = 0.3

	precision	recall	f1-score	suppor
0	1.00	0.94	0.97	363
1	0.89	0.95	0.92	364
2	0.95	0.99	0.97	364
3	0.97	0.99	0.98	336
4	1.00	0.99	1.00	364
5	0.99	0.92	0.95	335
6	1.00	1.00	1.00	336
7	0.99	0.90	0.94	364
8	0.93	1.00	0.96	335
9	0.94	0.97	0.95	336
accuracy			0.96	3497
macro avg	0.97	0.96	0.96	3497
weighted avg	0.97	0.96	0.96	3497



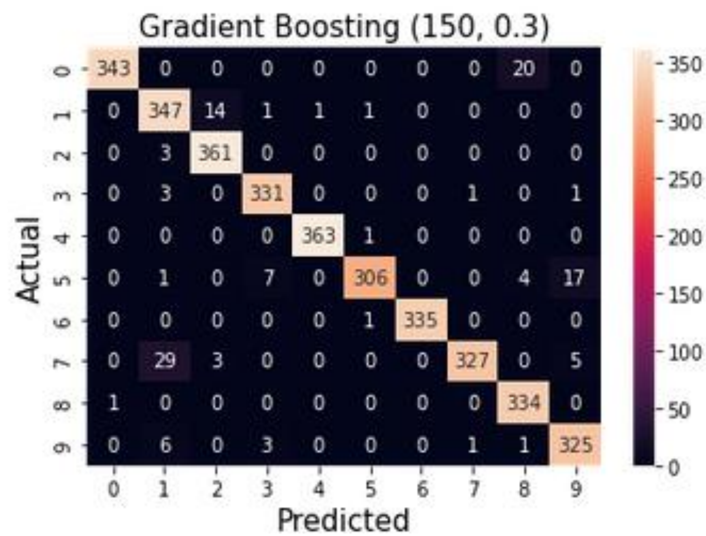
- **Fifth model (num of est = 150 , lr_rate = 0.1)**

```
Gradient Boosting Best 6 Accuracies:   Num of Est =  150        Learning Rate =  0.1
              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      0.97       336
           4       1.00      1.00      1.00       364
           5       0.99      0.92      0.96       335
           6       1.00      1.00      1.00       336
           7       0.99      0.90      0.94       364
           8       0.93      1.00      0.96       335
           9       0.93      0.96      0.94       336

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



- **Sixth model (num of est = 150 , lr_rate = 0.3)**

```
Gradient Boosting Best 6 Accuracies:   Num of Est =  150        Learning Rate =  0.3
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       363
           1       0.89      0.95      0.92       364
           2       0.96      0.99      0.97       364
           3       0.97      0.99      0.98       336
           4       1.00      1.00      1.00       364
           5       0.99      0.91      0.95       335
           6       1.00      1.00      1.00       336
           7       0.99      0.90      0.94       364
           8       0.93      1.00      0.96       335
           9       0.93      0.97      0.95       336

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



4(b): now we will use the same combinations to train XG_Boost models.

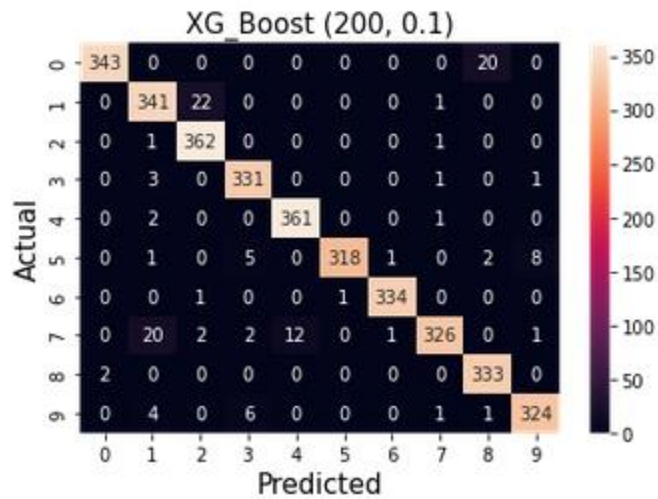
Learning_rate = [0.1,0.3]

n_estimators = [160,150,200]

- **First XG_Boost model (num of est = 200, lr_rate = 0.1)**

XG_Boost Best 6 Accuracies: Num of Est = 200 Learning Rate = 0.1

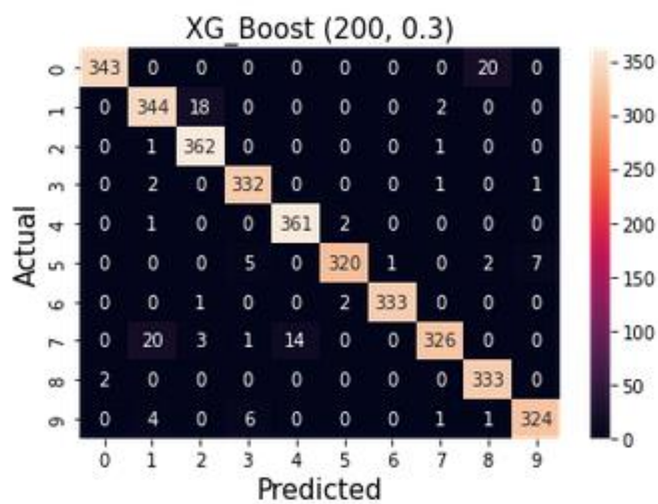
	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.92	0.94	0.93	364
2	0.94	0.99	0.96	364
3	0.96	0.99	0.97	336
4	0.97	0.99	0.98	364
5	1.00	0.95	0.97	335
6	0.99	0.99	0.99	336
7	0.98	0.90	0.94	364
8	0.94	0.99	0.96	335
9	0.97	0.96	0.97	336
accuracy			0.96	3497
macro avg	0.97	0.97	0.96	3497
weighted avg	0.97	0.96	0.96	3497



- **Second XG_Boost model (num of est = 200, lr_rate = 0.3)**

XG_Boost Best 6 Accuracies: Num of Est = 200 Learning Rate = 0.3

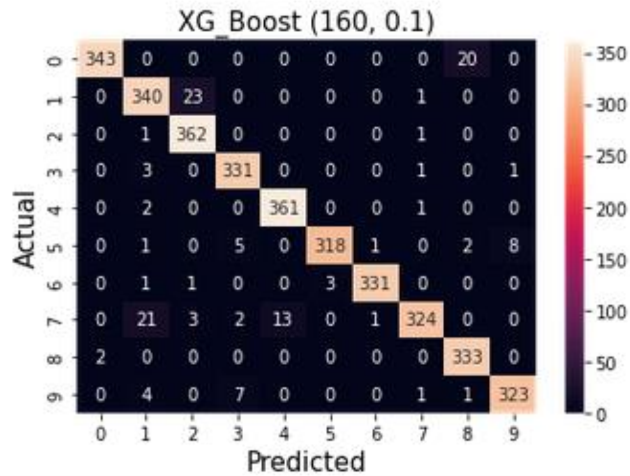
	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.92	0.95	0.93	364
2	0.94	0.99	0.97	364
3	0.97	0.99	0.98	336
4	0.96	0.99	0.98	364
5	0.99	0.96	0.97	335
6	1.00	0.99	0.99	336
7	0.98	0.90	0.94	364
8	0.94	0.99	0.96	335
9	0.98	0.96	0.97	336
accuracy			0.97	3497
macro avg	0.97	0.97	0.97	3497
weighted avg	0.97	0.97	0.97	3497



- **Third XG_Boost model (num of est = 160, lr_rate = 0.1)**

XG_Boost Best 6 Accuracies: Num of Est = 160 Learning Rate = 0.1

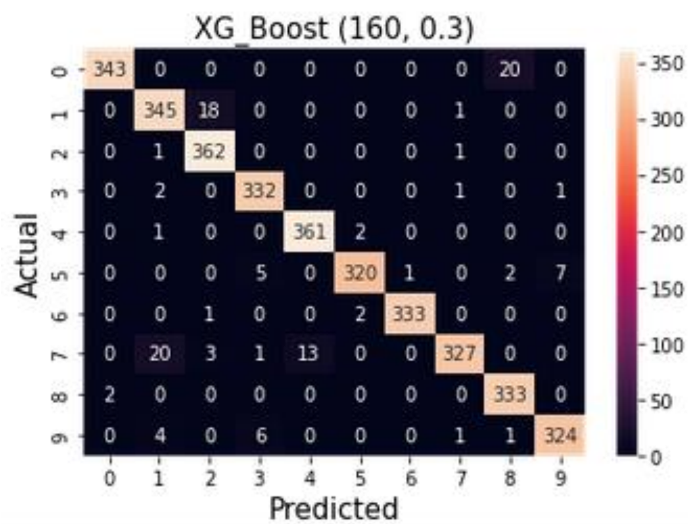
	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.91	0.93	0.92	364
2	0.93	0.99	0.96	364
3	0.96	0.99	0.97	336
4	0.97	0.99	0.98	364
5	0.99	0.95	0.97	335
6	0.99	0.99	0.99	336
7	0.98	0.89	0.94	364
8	0.94	0.99	0.96	335
9	0.97	0.96	0.97	336
accuracy			0.96	3497
macro avg	0.96	0.96	0.96	3497
weighted avg	0.96	0.96	0.96	3497



- **Fourth XG_Boost model (num of est = 160, lr_rate = 0.3)**

XG_Boost Best 6 Accuracies: Num of Est = 160 Learning Rate = 0.3

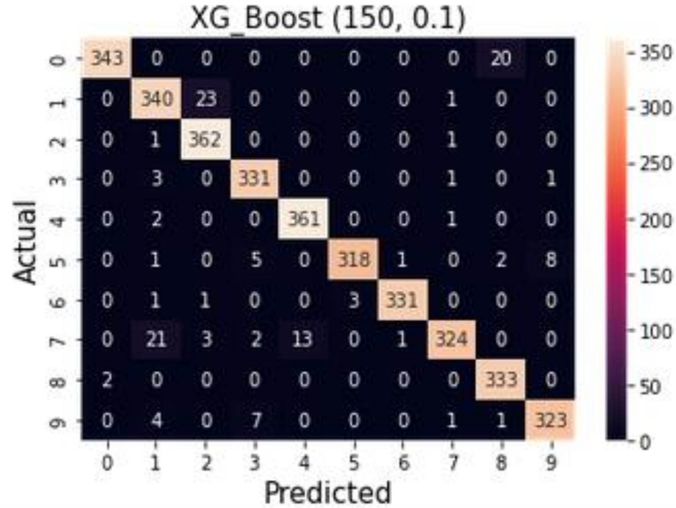
	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.92	0.95	0.94	364
2	0.94	0.99	0.97	364
3	0.97	0.99	0.98	336
4	0.97	0.99	0.98	364
5	0.99	0.96	0.97	335
6	1.00	0.99	0.99	336
7	0.99	0.90	0.94	364
8	0.94	0.99	0.96	335
9	0.98	0.96	0.97	336
accuracy			0.97	3497
macro avg	0.97	0.97	0.97	3497
weighted avg	0.97	0.97	0.97	3497



- **Fifth XG_Boost model (num of est = 150, lr_rate = 0.1)**

XG_Boost Best 6 Accuracies: Num of Est = 150 Learning Rate = 0.1

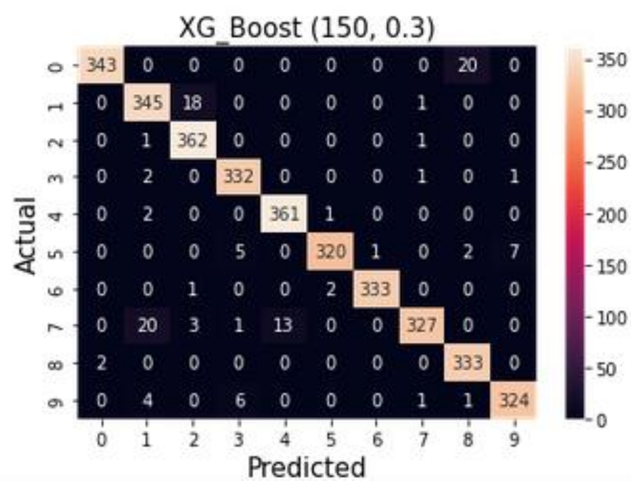
	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.91	0.93	0.92	364
2	0.93	0.99	0.96	364
3	0.96	0.99	0.97	336
4	0.97	0.99	0.98	364
5	0.99	0.95	0.97	335
6	0.99	0.99	0.99	336
7	0.98	0.89	0.94	364
8	0.94	0.99	0.96	335
9	0.97	0.96	0.97	336
accuracy			0.96	3497
macro avg	0.96	0.96	0.96	3497
weighted avg	0.96	0.96	0.96	3497



- **Sixth XG_Boost model (num of est = 150, lr_rate = 0.3)**

XG_Boost Best 6 Accuracies: Num of Est = 150 Learning Rate = 0.3

	precision	recall	f1-score	support
0	0.99	0.94	0.97	363
1	0.92	0.95	0.93	364
2	0.94	0.99	0.97	364
3	0.97	0.99	0.98	336
4	0.97	0.99	0.98	364
5	0.99	0.96	0.97	335
6	1.00	0.99	0.99	336
7	0.99	0.90	0.94	364
8	0.94	0.99	0.96	335
9	0.98	0.96	0.97	336
accuracy			0.97	3497
macro avg	0.97	0.97	0.97	3497
weighted avg	0.97	0.97	0.97	3497



4(c): here we have measured the time that each method (**XG_Boost** and **Gradient boost**) takes to train the six model, and we found that **XG_Boost is faster by Approximately 8 times.**

and also, XG_Boost gave better accuracies.

```
Done
19.37900400161743

           0      1    2
3  96.654275  160.0  0.3
5  96.654275  150.0  0.3
1  96.597083  200.0  0.3
0  96.454104  200.0  0.1
2  96.253932  160.0  0.1
4  96.253932  150.0  0.1
```

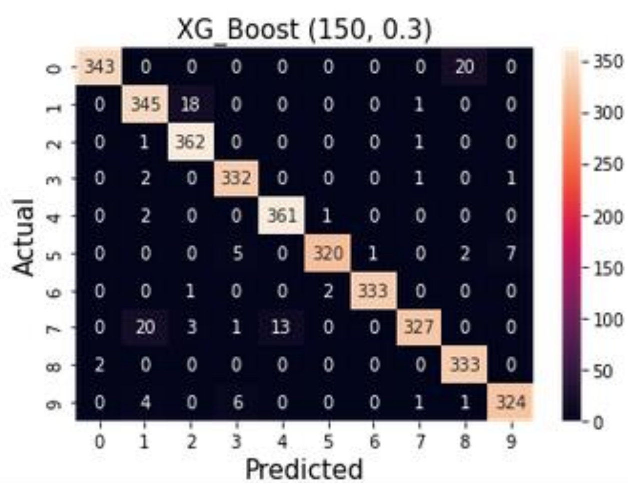
- **XB_Boost Time and accuracies**

```
Done
153.97868871688843

           0      1    2
0  96.568487  200.0  0.1
1  96.511295  200.0  0.3
2  96.482699  160.0  0.1
4  96.482699  150.0  0.1
5  96.425508  150.0  0.3
3  96.396912  160.0  0.3
```

Gradient boost Time and accuracies

- We believe that the both evaluation metrics are important **(accuracy** and **confusion matrix),** but in this case and this dataset (pen digits) we think that **confusion matrix would be more important**, because it will tell us which numbers mislead the model and why…

- for example, when we analyze this confusion matrix, we will find that the model misled the digit ‘7’ and it predicted it as ‘1’ (20 times) and that give us indicator that ‘7’ is kind of similar to ‘1’ in handwritten digits.



- Based on question 3 and 4, we have notice Bagging is the best option to avoid over-fitting.