ELG5255 Applied Machine Learning

Group Assignment #4

Group 4:

1. Yomna Mohamed Sayed Ahmed ID: 300327217
2. Khaled Mohamed Mohamed Mahmoud ID: 300327242
3. Marwa Hamdi Boraie Mahmoud ID: 300327267

Part 1: *Numerical Questions*

→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.

Table 1:

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

(a) Please build a decision tree by using **Gini Index** (i.e., Gini = 1-$\sum_{i=1}^{NC}(Pi)^2$, where NC is the number of classes).

(1) build a Decision tree by using Gini Index

• $Gini = 1 - \sum_{i=1}^{NC} (P_i)^2$, NC → number of classes

• Compute the Gini Index for the overall Collection of training Examples. , • there are two Possible output variables Yes, No.

• the data has 3 instances of (Yes), and 7 instances of (No). (dataset)

• $Gini(S) = 1 - [(3/10)^2 + (7/10)^2] = 1 - 0.58 = 0.42$

• Computation of Gini Index for (weather) Attribute.

It has three Possible values of cloudy (3 examples), sunny (4 examples) and Rainy (3 examples).

- for weather = cloudy , there are 2 examples with "No" and 1 example with "yes"
  - $Gini(S) = 1 - [(2/3)^2 + (1/3)^2] = \boxed{4/9}$
- for weather = sunny , there are 3 exampes with "No" and 1 example with "yes"
  - $Gini(S) = 1 - [(3/4)^2 + (1/4)^2] = \boxed{3/8}$
- for weather = Rainy , there are 2 examples with "No" and 1 exampe with "yes"
  - $Gini(S) = 1 - [(2/3)^2 + (1/3)^2] = \boxed{4/9}$

- weighted Average (weather) = $(4/9 * 3/10) + (3/8 * 4/10) + (4/9 * 3/10) = \boxed{5/12}$

- Computation of Gini Index for (Temperature) Attribute.
- It has three Possible values of Cool (3 examples), Hot (3 examples) and Mild (4 examples).

- for Temp = Cool , there are 3 examples, all with "No"
  - $Gini(S) = 1 - (3/3)^2 = \boxed{0}$
- for Temp = Hot , there are 2 examples "No" and 1 example "yes"
  - $Gini(S) = 1 - [(2/3)^2 + (1/3)^2] = \boxed{4/9}$
- for Temp = Mild , there are 2 examples "No" and 2 examples "yes"

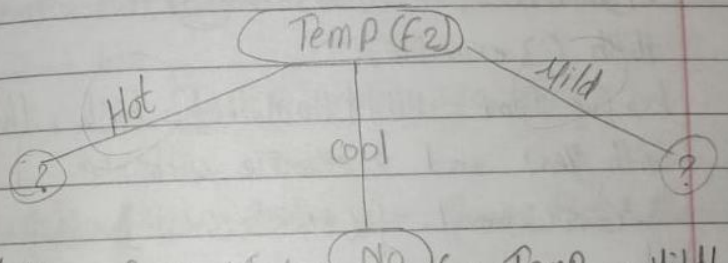
- $Gini(S) = 1 - [(2/4)^2 + (2/4)^2] = \boxed{1/2}$

- weighted Average (Temp) = $(0 * 3/10) + (4/9 * 3/10)$
$+ (1/2 * 4/10) = \boxed{1/3}$
- Computation of Gini Index for (Humidity) Attribute.
- It has two Possible values of Normal (4 examples) and High (6 examples)
- for Humidity = Normal, there are 3 examples with "No" and 1 example with "yes"
  - $Gini(S) = 1 - [(3/4)^2 + (1/4)^2] = \boxed{3/8}$
- for Humidity = High, there are 4 examples with "No" and 2 examples with "yes"
  - $Gini(S) = 1 - [(4/6)^2 + (2/6)^2] = \boxed{4/9}$
- weighted Average (Humidity) = $(3/8 * 4/10) + (4/9 * 6/10)$
$= \boxed{5/12}$
- Computation of Gini Index for (wind) Attribute.
- It has two Possible values of strong (6 examples) and weak (4 examples)
- for wind = Strong, there are 5 examples with "No" and 1 example with "yes"
  - $Gini(S) = 1 - [(5/6)^2 + (1/6)^2] = \boxed{5/18}$
- for wind = weak there are 2 examples → "yes" and 2 →
  - $Gini(S) = 1 - [(2/4)^2 + (2/4)^2] = \boxed{1/2}$

OSCAR

• weighted Average(wind) = $(5/18 * 6/10) + (1/2 * 4/10)$
$= \boxed{11/30}$

• Gini Index (weather) = 0.4167
• Gini Index (Temp) = 0.333
• Gini Index (Humidity) = 0.4167
• Gini Index (wind) = 0.3667



• computation of Gini Index for Temp = Mild | weather feature

• cloudy ②, • for Temp = Mild | weather = cloudy, there are 1 instance with "yes" and 1 instance with "No"

$Gini(S) = 1 - [(1/2)^2 + (1/2)^2] = 1/2 = \boxed{0.5}$

• Rainy (1 example)
for Temp = Mild | weather = Rainy there is 1 instance with "yes"

• $Gini(S) = 1 - (1/1)^2 = 0$

• Sunny (1 example)
↳ for Temp = Mild | weather = sunny, there is 1 instance with "No"

• $Gini(S) = 1 - (1/1)^2 = 0$

OSCAR

- weighted Average (Temp = Mild | weather) =
  $(0 * 1/4) + (0 * 1/4) + (0.5 * 2/4) = 0.25$
- computation of Gini Index for Temp = Mild | Humidity

  Normal (1 example)

  ↳ for temp = Mild | Humidity = Normal, there is 1 example with "yes"

  - $Gini(S) = 1 - (1/1)^2 = 0$

  High (3 example)

  ↳ for Temp = Mild | Humidity = high, there is 1 example with "yes" and 2 example with "No"

  - $Gini(S) = 1 - [(1/3)^2 + (2/3)^2] = 4/9$

- weighted Average (Temp = Mild | Humidity) =
  $(0 * 1/4) + (4/9 * 3/4) = 1/3$
- Computation of Gini Index for temp = Mild | wind

  weak (1 example)

  ↳ for temp = Mild | wind = weak, there is 1 instance of "yes"

  - $Gini(S) = 1 - (1/1)^2 = 0$

  Strong (3 examples)

  ↳ for temp = Mild | wind = strong there is 1 instance "yes" & 2 instance "No"

  $Gini(S) = 1 - [(1/3)^2 + (2/3)^2] = 4/9$

- Weighted Average (Temp = Mild | wind)
  $= (0 * 1/4) + (4/9 * 3/4) = 1/3$

- Gini Index (Temp = Mild | weather) = 0.25
- Gini Index (Temp = Mild | Humidity) = 0.333
- Gini Index (Temp = Mild | wind) = 0.333

- Computation of Gini Index for Temp = Hot | weather feature
- Sunny → 3 instances

for temp = hot | weater = sunny, there is 1 → yes and 2 → No

- $Gini(S) = 1 - [(1/3)^2 + (2/3)^2] = 4/9$

- Computation of Gini Index for Temp = Hot | Humidity feature.
  - High (3 examples) 1 → yes, 2 → No
  - $Gini(S) = 1 - [(1/3)^2 + (2/3)^2] = 4/9$

- Computation of Gini Index for Temp = Hot | wind feature

weak → 1 → yes

- $Gini(S) = 1 - [(1/1)^2] = 0$

strong → 2 → No

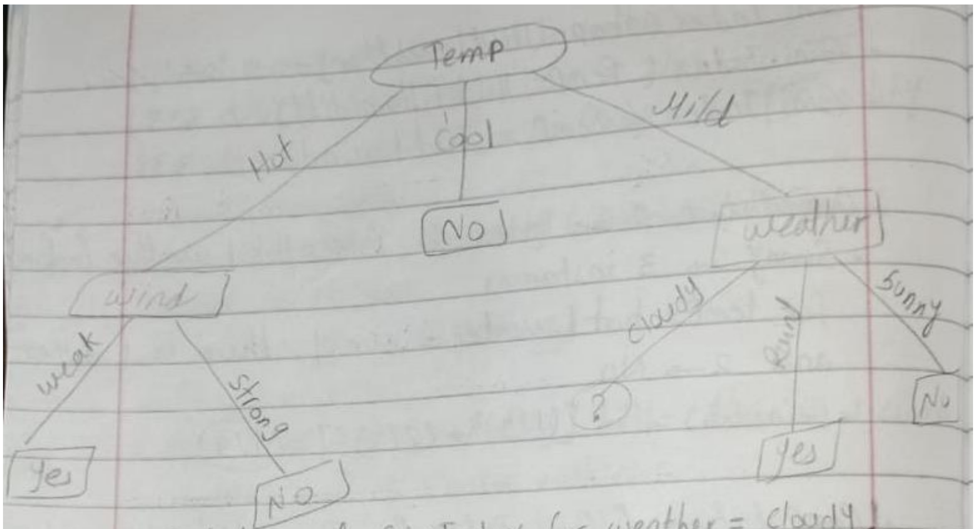
- $Gini(S) = 1 - [(2/2)^2] = 0$

Gini(Temp = Hot | Humidity = weak, strong) = 0

→ Gini Index (Temp = Hot | weather) = 0.444

→ Gini Index (Temp = Hot | Humidity) = 0.444

→ Gini Index (temp = Hot | wind) = 0



- Computation of Gini Index for weather = cloudy, Temp = Mild | Humidity
  - $Gini(S) = 1 - [(1/2)^2 + (1/2)^2] = 1/2$

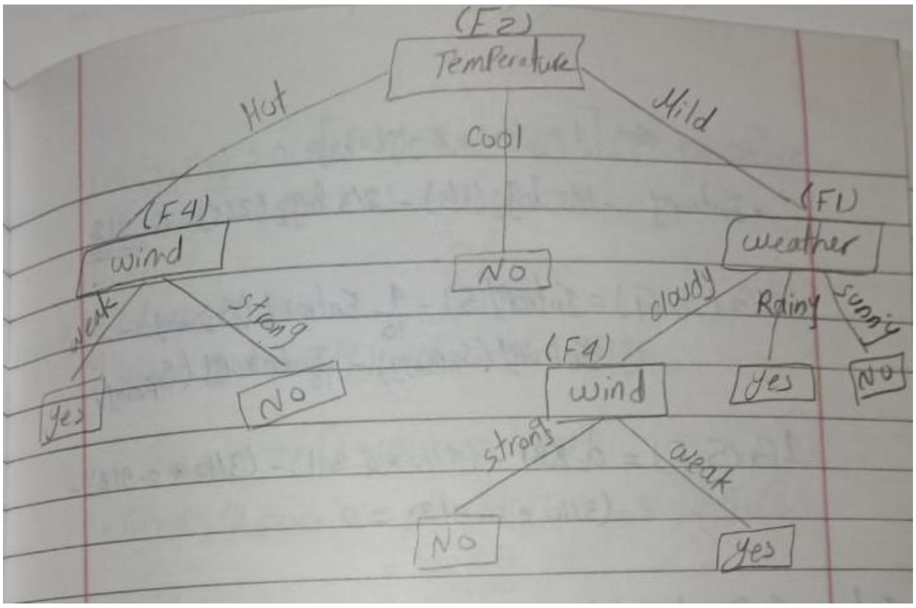
- Computation of Gini Index for weather = cloudy, Temp = Mild | wind
  - Strong → No
    $Gini(S) = 1 - (1/1)^2 = 0$
  - weak → Yes
    $Gini(S) = 1 - (1/1)^2 = 0$

  → 0

→ Gini Index (Temp = Mild , weather = cloudy | Humidity)
= 1/2

→ Gini Index (Temp = Mild , weather = cloudy | wind = zero

we will choose the wind feature which has the minimum Gini Index.



(b) Please build a decision tree by using **Information Gain** (i.e., IG(T, a) = Entropy(T)− Entropy(T|a)**.**

(2) build a decision tree by using Information Gain

• IG(T, a) = Entropy(T) − Entropy(T/a)

step 1: we will calculate the Entropy of data set :- (S)

(label) S = [3 + (yes), 7 − (No)]

• Entropy(S) = $-3/10 \log_2(3/10) - 7/10 \log_2(7/10) = 0.881$

step 2: (calculate the IG for 4 features)

Feature : weather(F1)

• Values(weather) = cloudy, Sunny, Rainy

$S_{sunny} \leftarrow$ [1 + (yes), 3 − (No)]

• Entropy(S_{sunny}) = $-1/4 \log_2(1/4) - 3/4 \log_2(3/4)$

= 0.811

$S_{cloudy} \leftarrow$ [1 + (yes), 2 − (No)]

• Entropy(S_{cloudy}) = $-1/3 \log_2(1/3) - 2/3 \log_2(2/3) = 0.918$

$S_{Rainy} \leftarrow [1+(yes), 2-(No)]$

- $Entropy = -1/3 \log_2(1/3) - 2/3 \log_2(2/3) = \underline{0.918}$

$\therefore IG(S, F_1) = Entropy(S) - \frac{4}{10} Entropy(S_{sunny}) - \frac{3}{10} Entropy(S_{cloudy}) - \frac{3}{10} Entropy(S_{Rainy})$

$IG(S, F_1) = 0.881 - (4/10 * 0.811) - (3/10 * 0.918) - (3/10 * 0.918) = 0.0058$

Feature: Temperature (F2)

- Values(Temperature) = Cool, Hot, Mild

$S_{Cool} \leftarrow [0+(yes), 3-(No)]$

- $Entropy(S_{Cool}) = -0/3 \log_2(0/3) - 3/3 \log_2(3/3) = \underline{0}$

$S_{Hot} \leftarrow [1+(yes), 2-(No)]$

- $Entropy(S_{Hot}) = -1/3 \log_2(1/3) - 2/3 \log_2(2/3) = \underline{0.918}$

$S_{Mild} \leftarrow [2+(yes), 2-(No)]$

- $Entropy = -2/4 \log_2(2/4) - 2/4 \log_2(2/4) = \underline{1}$

$\therefore IG(S, F_2) = Entropy(S) - \frac{3}{10} Entropy(S_{Cool}) - 3/10\, Entropy(S_{Hot}) - 4/10\, Entropy(S_{Mild})$

OSCAR

$IG(S, F_2) = 0.881 - ((3/10) * 0) - (3/10 * 0.918) - (4/10 * 1) = 0.2056$

Feature: Humidty (F_3)

- Values (Humidty) = Normal, High

$S_{Normal} \leftarrow [1+(Yes), 3-(No)]$

- $Entropy(S_{Normal}) = -1/4 \log_2(1/4) - 3/4 \log_2(3/4) = \underline{\underline{0.811}}$

$S_{High} \leftarrow [2+(Yes), 4-(No)]$

- $Entropy(S_{High}) = -2/6 \log_2(2/6) - 4/6 \log_2(4/6) = \underline{\underline{0.918}}$

$\therefore IG = Entropy(S) - 4/10\ Entropy(S_{Normal}) - 6/10\ Entropy(S_{High})$

$(S, F_3)\ IG = 0.881 - (4/10 * 0.811) - (6/10 * 0.918) = 0.0058$

Feature: Wind (F_4)

- Values (Wind) = Strong, Weak

$S_{Strong} \leftarrow [1+(Yes), 5-(No)]$

- $Entropy(S_{Strong}) = -1/6 \log_2(1/6) - 5/6 \log_2(5/6) = \underline{\underline{0.650}}$

$S_{weak} \leftarrow [2+(yes),\ 2-(No)]$

- $Entropy(S_{weak}) = -2/4 \log_2(2/4) - 2/4 \log_2(2/4) = \underline{1}$

$\therefore IG(S, F_4) = Entropy(S) - 6/10\ Entropy(S_{strong}) - 4/10\ Entropy(S_{weak})$

$IG(S, F_4) = 0.881 - (6/10 * 0.650) - (4/10 * 1) = 0.091$

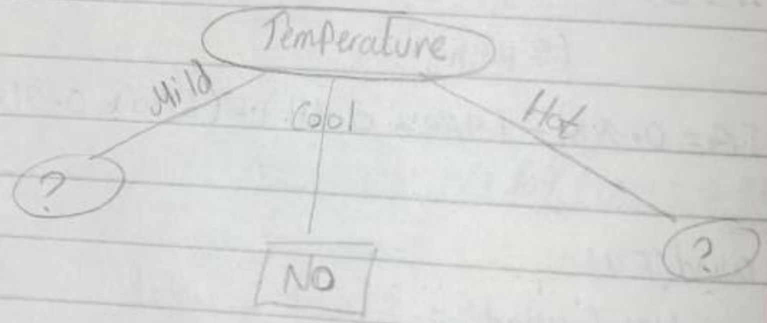
$\therefore$ Gain (S, weather) = 0.0058

, Gain (S, Temperature) = 0.2056 ✓

, Gain (S, Humidity) = 0.0058

, Gain (S, wind) = 0.091

↳ So, F2 (Temperature) it will be the first split (root) as it has the maximum value of Information Gain.



F(1)	(F 2)	(F 3)	(F 4)	(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

- Calculate the IG for (F1), (F 3) & (F 4) features with $F_2 = \underline{Mild}$

Feature: weather (F1)

- values (weather) = Sunny, Rainy, cloudy

$S_{Mild} \leftarrow [2+(yes), 2-(No)]$

- $Entropy(S_{Mild}) = -2/4 \log_2(2/4) - 2/4 \log_2(2/4)$
  $= \underline{\underline{1}}$

$S_{Sunny} \leftarrow [0+(yes), 1-(No)]$

- $Entropy(S_{Sunny}) = -0/1 \log_2(0/1) - 1/1 \log_2(1/1) = \underline{\underline{0}}$

$S_{Rainy} \leftarrow [1+(yes), 0-(No)]$

- $Entropy(S_{Rainy}) = -1/1 \log(1) - 0/1 \log_2(0/1) = \underline{\underline{0}}$

$S_{Cloudy} \leftarrow [1+(yes), 1-(No)]$

- $Entropy(S_{Cloudy}) = -1/2 \log_2(1/2) - 1/2 \log_2(1/2)$
  $= 1$ //

$$\therefore IG(S_{Mild}, F_1) = Entropy(S_{Mild}) - 1/4\ Entropy(S_{Sunny}) - 1/4\ Entropy(S_{Rainy}) - 2/4\ Entropy(S_{cloudy})$$

$$IG(S_{Mild}, F_1) = 1 - (1/4 * 0) - (1/4 * 0) - (2/4 * 1) = 0.5$$

Feature: Humidty (F_3)

- values (Humidty) = High, Normal

$$S_{High} \leftarrow [1+(yes),\ 2-(No)]$$

- $Entropy(S_{High}) = -1/3 \log_2(1/3) - 2/3 \log_2(2/3) = 0.918$

$$S_{Normal} \leftarrow [1+(yes),\ 0-(No)]$$

- $Entropy(S_{Normal}) = -1/1 \log_2(1) - 0/1 \log_2(0/1) = 0$

$$\therefore IG(S_{Mild}, F_3) = Entropy(S_{Mild}) - 3/4\ Entropy(S_{High}) - 1/4\ Entropy(S_{Normal})$$

$$IG(S_{Mild}, F_3) = 1 - (3/4 * 0.918) - (1/4 * 0) = 0.3115$$

Feature: wind (F_4)

- values (wind) = Strong, weak

$$S_{strong} \leftarrow [1+(yes),\ 2-(No)]$$

- Entropy (S_{strong}) $= -1/3 \log_2(1/3) - 2/3 \log_2(2/3)$
$= \underline{0.918}$

$S_{weak} \leftarrow [1+(yes), 0-(No)]$

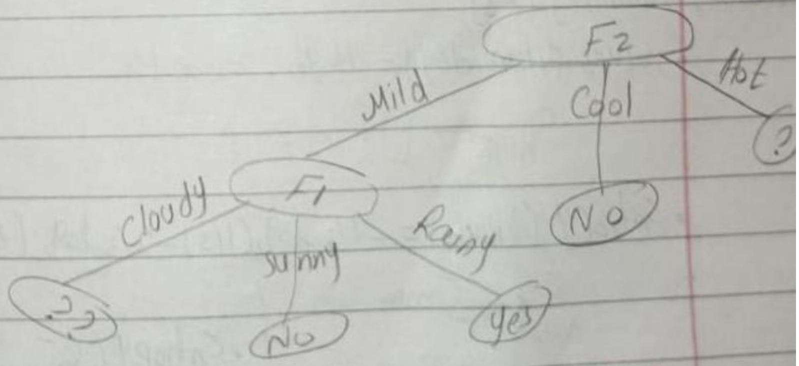
- Entropy (S_{weak}) $= \underline{0}$

$\therefore IG(S_{Mild}, F_4) =$ Entropy (S_{Mild}) $- 3/4$ Entropy (S_{strong}) $- 1/4$ Entropy (S_{weak})

$IG(S_{Mild}, F_4) = 1 - (3/4 * 0.918) - (1/4 * 0)$
$= \underline{0.3115}$

$\therefore$ Gain (S_{Mild}, weather) $= 0.5$ ✓
, Gain (S_{Mild}, Humidty) $= 0.3115$
, Gain (S_{Mild}, wind) $= 0.3115$

→ So, we will split Temperature = Mild with weather(F_1) feature; because it has the Maximum value of Information Gain.



- calculate (F1) (F3) & (F4) features with $F_2 = Hot$

$S_{Hot} \leftarrow [1+ (yes),\ 2-(No)]$

- $Entropy(S_{Hot}) = -1/3 \log_2 (1/3) - 2/3 \log_2 (2/3) = 0.918$

Feature: weather(F1)

- values (weather) = sunny, Rainy, cloudy

$S_{sunny} \leftarrow [1+ (yes),\ 2-(No)]$

- $Entropy(S_{sunny}) = -1/3 \log_2 (1/3) - 2/3 \log_2 (2/3) = 0.918$

$S_{Rainy} \leftarrow [0+, 0-]$ • $Entropy(S_{Rainy}) = 0$

$S_{cloudy} \leftarrow [0+, 0-]$ • $Entropy(S_{cloudy}) = 0$

$\therefore IG(S_{Hot}, F_1) = Entropy(S_{Hot}) - 3/3\ Entropy(S_{sunny}) - 0/3\ Entropy(S_{Rainy}) - 0/3\ Entropy(S_{cloudy})$

$IG(S_{Hot}, F_1) = 0.918 - (3/3 * 0.918) - (0/3 * 0) - (0/3 * 0) = 0$

Feature: Humidty (F3)

- values (Humidty) = High, Normal

$S_{High} \leftarrow [1+, 2-]$

- $Entropy(S_{High}) = -1/3 \log_2 (1/3) - 2/3 \log_2 (2/3) = 0.918$

$S_{Normal} \leftarrow [0+, 0-]$

- $Entropy(S_{Normal}) = 0$

OSCAR

$\therefore IG(S_{Hot}, F3) = Entropy(S_{Hot}) - 3/3\ Entropy(S_{High})$
$- 0/3\ Entropy(S_{Normal})$

$IG(S_{Hot}, F_3) = 0.918 - (3/3 * 0.918) - (\cancel{0/3 * 0})$
$= \underline{0}$

Feature: wind (F4)

values (F4) = strong, weak

$S_{strong} \leftarrow [0+, 2-]$

• $Entropy(S_{strong}) = -0/2 \log_2(0/2) - 2/2 \log_2(2/2)$
$= \underline{0}$

$S_{weak} \leftarrow [1+, 0-]$ • $Entropy(S_{weak}) = \underline{0}$

$\therefore IG(S_{hot}, F_4) = Entropy(S_{hot}) - 2/3\ Entropy(S_{strong}) -$
$1/3\ Entropy(S_{weak})$

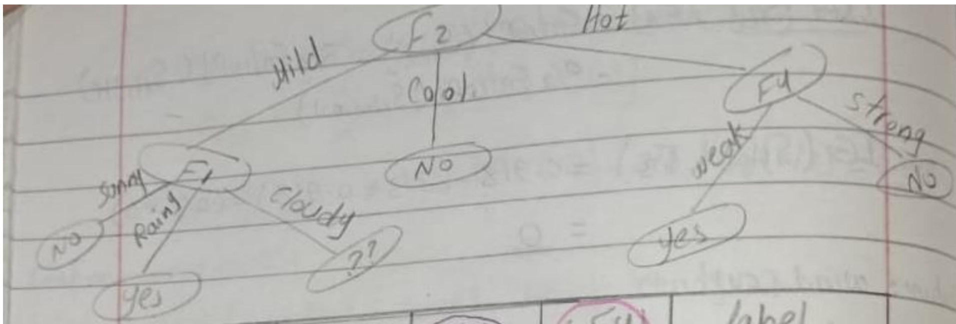
$IG(S_{hot}, F_4) = 0.918 - (\cancel{2/3 * 0}) - (\cancel{1/3 * 0}) = \underline{\underline{0.918}}$

$\therefore Gain(S_{Hot}, weather) = 0$

, $Gain(S_{Hot}, Humidty) = 0$

, $Gain(S_{Hot}, wind) = 0.918$ ✓

⇒ So, we will split temperature = Hot with wind (F4) feature, because it has the maximum value of Information Gain.



(F1)	(F2)	(F3)	(F4)	label
cloudy	Mild	high	Strong	No
Cloudy	Mild	high	weak	yes

- Calculate (F3) & (F4) with F2 = Mild & F1 = Cloudy

  $S_{Mild, cloudy} \leftarrow [1+, 1-]$

  - $Entropy(S_{Mild, cloudy}) = -1/2 \log_2(0.5) - 1/2 \log_2(0.5) = 1$

Feature: Humidty (F3)

- values (Humidty) = high, Normal

  $S_{High} \leftarrow (1+, 1-)$

- $Entropy(S_{High}) = -1/2 \log_2(0.5) - 1/2 \log_2(0.5)$

  $= 1$

  $S_{Normal} \leftarrow (0+, 0-)$

  - $Entropy(S_{Normal}) = 0$

$\therefore IG(S_{Mild, Cloudy}, F3) = Entropy(S_{Mild, cloudy}) - 2/2\, Entropy(S_{High}) - 0/2\, Entropy(S_{Normal})$

$IG(S_{Mild,Cloudy}, F_3) = 1 - (2/2 * 1) - (\cancel{0/2} * 0)$
$= 0$

feature: wind (F_4)

- values (wind) = Strong, weak

$S_{strong} \leftarrow (0+, 1-)$

- $Entropy(S_{strong}) = -0/1 \log_2(0/1) - 1/1 \log_2(1/1)$
$= 0$

$S_{weak} \leftarrow (1+, 0-)$

- $Entropy(S_{weak}) = -1/1 \log_2(1) - 0/1 \log_2(0/1)$
$= 0$

$\therefore IG(S_{Mild, cloudy}, F_4) = Entropy(S_{Mild, cloudy}) - 1/2\ Entropy(S_{strong}) - 1/2\ Entropy(S_{weak})$

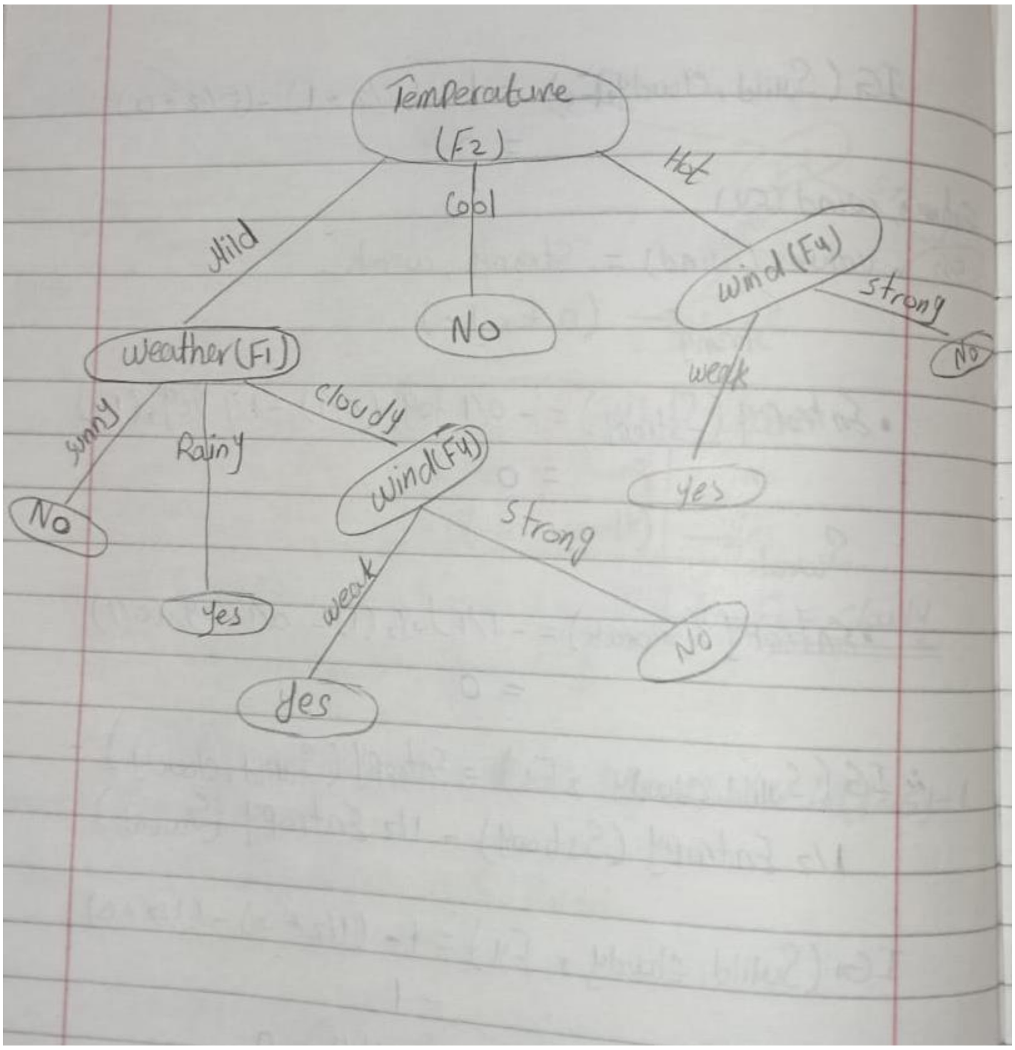
$IG(S_{Mild, cloudy}, F_4) = 1 - (\cancel{1/2} * 0) - (\cancel{1/2} * 0)$
$= 1$

$\therefore Gain(S_{Mild, cloudy}, Humidity) = 0$

, $Gain(S_{Mild, cloudy}, wind) = 1$ ✓

→ So, we will split weather = Cloudy & Temperature = Mild with wind (F_4) feature, because it has the maximum value of Information Gain.

OSCAR


Temperature
(F2)
Mild
Cool
Hot
weather(F1)
No
wind(F4)
strong
No
weak
yes
sunny
Rainy
cloudy
No
wind(F4)
Yes
strong
weak
No
Yes

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

	Advantages	Disadvantages
Gini Index	• favors larger partitions (distributions) and is very easy to implement and interpret. • Modification of the information gain that reduces its bias. • It deals with inequality, so it can judge the distribution pattern better.	• Sample Bias, the validity of the Gini index can be dependent on sample size. • Data Inaccuracy, the Gini index is sometimes prone to random and systematic data errors; it can create problems with the index value. • Degeneracy, in some exceptional cases, the Gini index value can be the same for different distributions.
Information Gain	• Information gain ratio biases the decision tree against considering attributes with a large number of distinct values. So, it solves the drawback of information gain namely, information gain applied to attributes that can take on a large number of distinct values might learn the training set too well. • It creates a comprehensive analysis of consequences along each branch and identifies decision nodes that need further analysis.	• A notable problem occurs when information gain is applied to attributes that can take on a large number of distinct values (biased). • Subsets are more likely to be pure if there are a large number of values (overfitting).

Part 2: *Programming Questions*

2-

Reading the Pen-Digits dataset

```
[17] import numpy as np
     import pandas as pd
     import re

     #============= Read CSV and apply data preperation ======
     df_train = pd.read_csv("pendigits-tra.csv",header=None)
     df_test=pd.read_csv("pendigits-tes.csv",header=None)
```

```
[18] X_train=df_train.iloc[:,:-1]
     y_train=df_train.iloc[: , -1]
     X_test=df_test.iloc[:,:-1]
     y_test=df_test.iloc[: , -1]
```

Apply decision tree to classify testing set

```
[5]  from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.metrics import classification_report
     from sklearn.metrics import accuracy_score

     estimator = DecisionTreeClassifier(random_state=2022)
     estimator.fit(X_train, y_train)
     y_pred = estimator.predict(X_test)
     report = classification_report(y_test, y_pred)
     print(report)
     acc = accuracy_score(y_test, y_pred)
     print(acc)
```

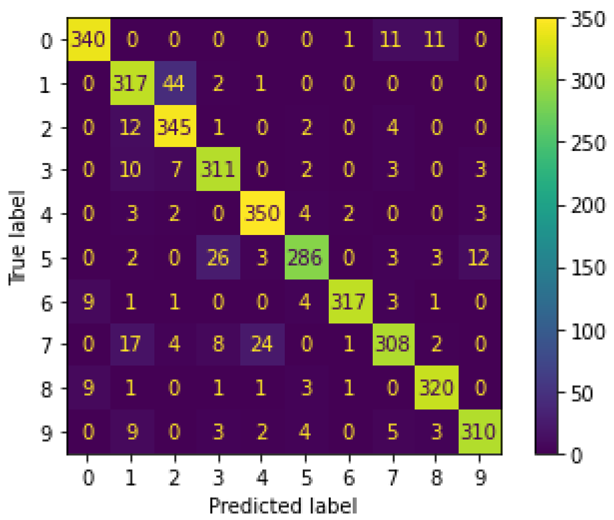
And this is the classification report

```
              precision    recall  f1-score   support

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

    accuracy                           0.92      3498
   macro avg       0.92      0.92      0.92      3498
weighted avg       0.92      0.92      0.92      3498

0.9159519725557461
```

This function draw a confusion matrix

```
[8] def draw_cm(y_test, y_pred):
      from sklearn import metrics
      import matplotlib.pyplot as plt
      confusion_matrix =metrics.confusion_matrix(y_test, y_pred)
      cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = ["0","1", "2", "3","4'

      cm_display.plot()
```

And this is the confusion matrix



3-(a)

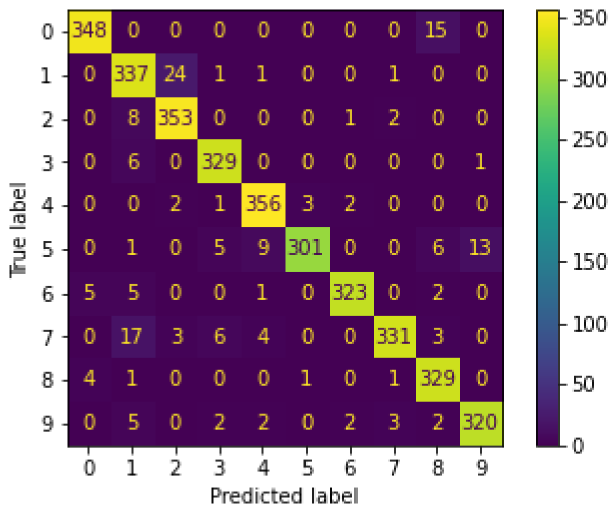
`Bagging using Decision Tree`

```
[40] from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import accuracy_score
     from sklearn.metrics import classification_report
     from sklearn.metrics import accuracy_score

     clf = DecisionTreeClassifier(random_state=2022)
     estimator=BaggingClassifier(base_estimator=clf,n_estimators=30,
                 max_samples=1.0)
     estimator.fit(X_train, y_train)
     y_pred = estimator.predict(X_test)
     report = classification_report(y_test, y_pred)
     print(report)
     acc = accuracy_score(y_test, y_pred)
     print(acc)
     draw_cm(y_test, y_pred)
```

And this is the classification report for Decision Tree

```
              precision    recall  f1-score   support

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

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

0.951114922813036
```

And this is the confusion matrix for Decision Tree



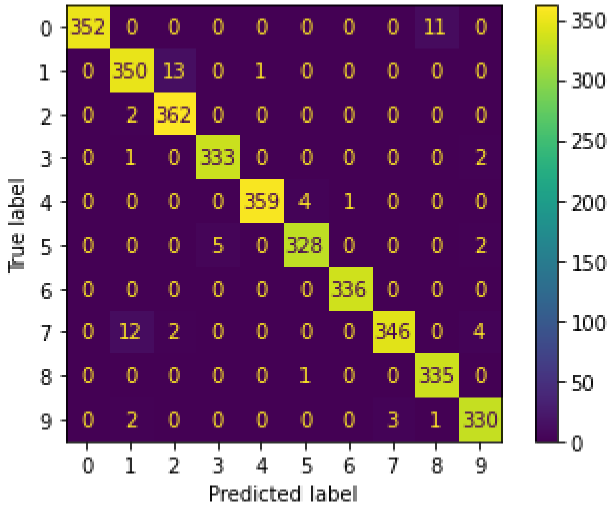
Bagging using SVM

```
from sklearn import svm

model = svm.SVC()
estimator_SVM=BaggingClassifier(base_estimator=model,n_estimators=30,
                    max_samples=1.0)
estimator_SVM.fit(X_train, y_train)
y_pred_SVM = estimator_SVM.predict(X_test)
report = classification_report(y_test, y_pred_SVM)
print(report)
acc = accuracy_score(y_test, y_pred_SVM)
print(acc)
draw_cm(y_test, y_pred_SVM)
```

And this is the classification report for SVM

```
              precision    recall  f1-score   support

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

    accuracy                           0.98      3498
   macro avg       0.98      0.98      0.98      3498
weighted avg       0.98      0.98      0.98      3498

0.98084619782733
```

And this is the confusion matrix for SVM

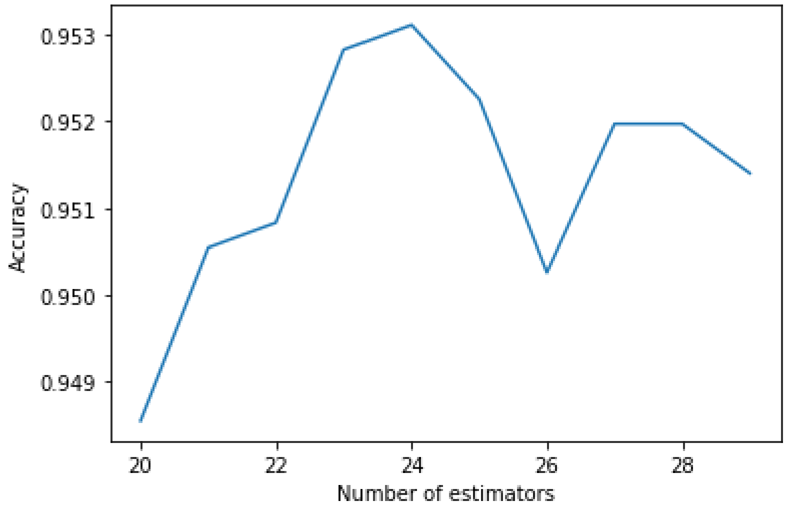


(b) Find the best number of estimators

```
from sklearn.ensemble import BaggingClassifier
from matplotlib import pyplot as plt
def get_best_num_est(n1,n2):
  estimatorts=[]
  accuracys=[]
  max_acc=0
  c=0
  for nEst in range(n1, n2):
    clf=DecisionTreeClassifier(random_state=2022)
    estimator = BaggingClassifier(base_estimator=clf,n_estimators=nEst, random_state=2022)
    estimator.fit(X_train, y_train)
    y_pred = estimator.predict(X_test)
    report = classification_report(y_test, y_pred)
    #print(report)
    acc = accuracy_score(y_test, y_pred)
    if acc>max_acc:
      max_acc=acc
      c=nEst
    estimatorts.append(nEst)
    accuracys.append(acc)
    #print(acc)

  plt.plot(estimatorts, accuracys)
  plt.xlabel("Number of estimators")
  plt.ylabel("Accuracy")

  plt.show()
  return(max_acc,c)
```

Get values [20,30] to the Estimators to find the best estimator that give us the highest accuracy

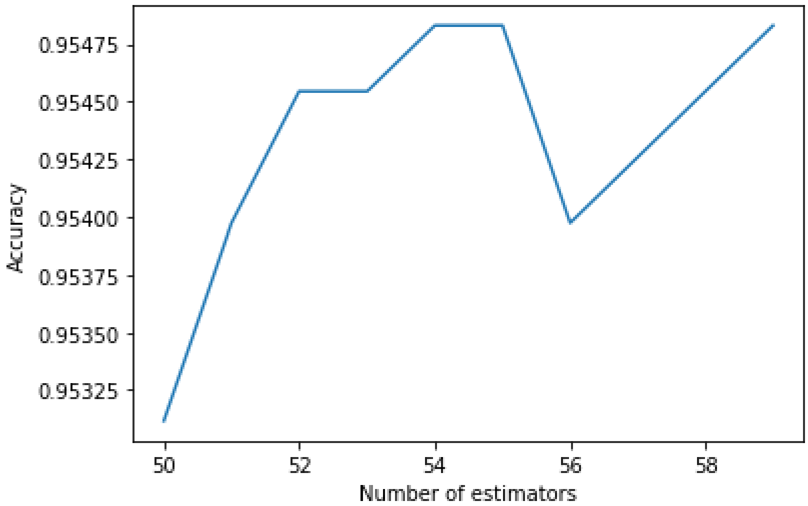
```
[47] print(get_best_num_est(20,30))
```




```
(0.9531160663236135, 24)
```

Get values [50,60] to the Estimators to find the best estimator that give us the highest accuracy

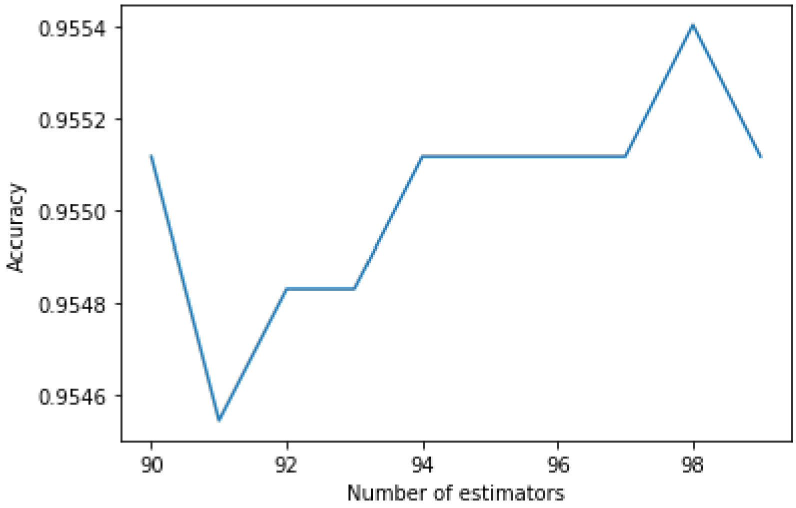
```
[48] print(get_best_num_est(50,60))
```




```
(0.9548313321898227, 54)
```

Get values [90,100] to the Estimators to find the best estimator that give us the highest accuracy

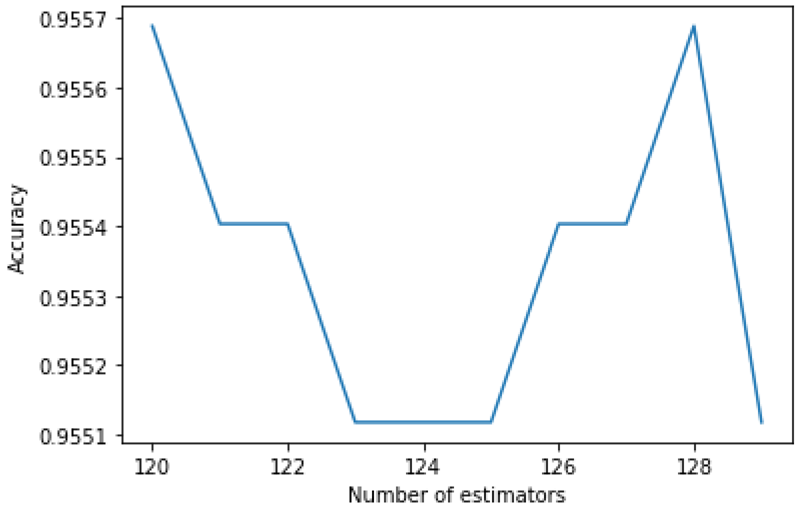
```
[49] print(get_best_num_est(90,100))
```




```
(0.9554030874785592, 98)
```

Get values [120,130] to the Estimators to find the best estimator that give us the highest accuracy

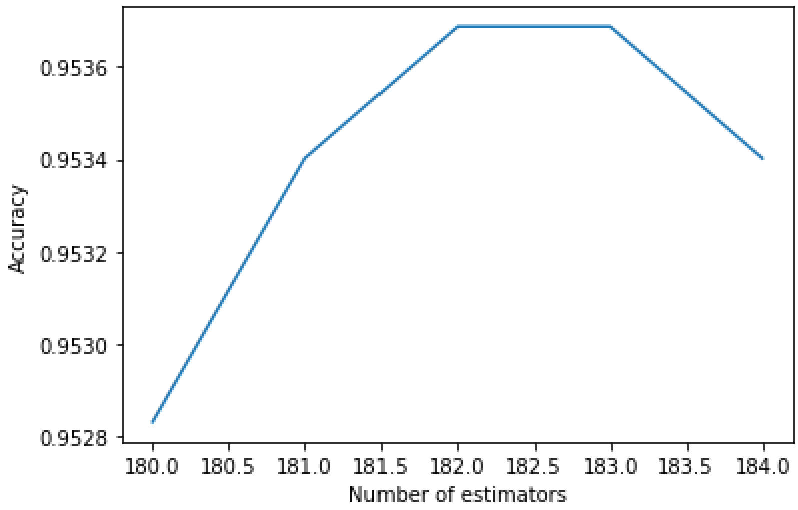
```
[50] print(get_best_num_est(120,130))
```




```
(0.9556889651229273, 120)
```

Get values [180,185] to the Estimators to find the best estimator that give us the highest accuracy

```
print(get_best_num_est(180,185))
```



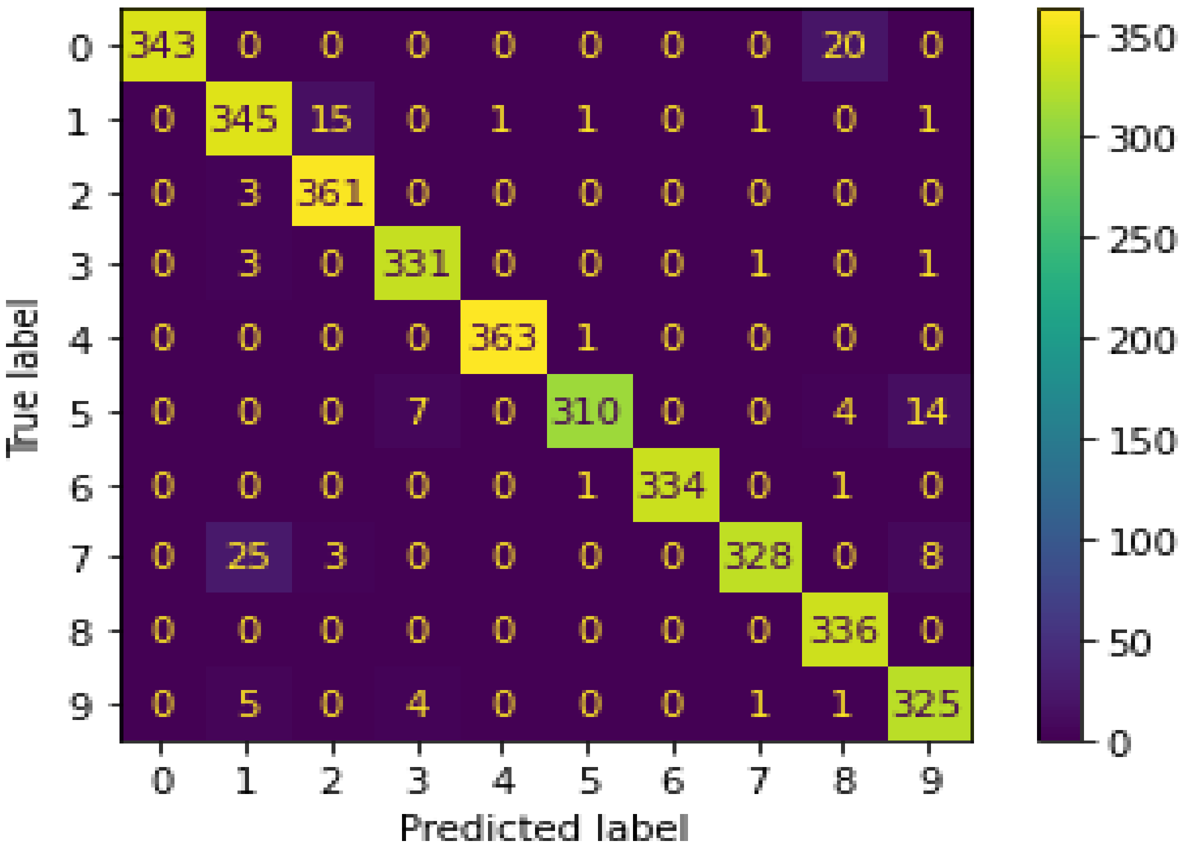

```
(0.9536878216123499, 182)
```

4) Boosting:

A) GradientBoosting classifier tuning number of estimator from [10, 100 , 150, 200] then the learning rate from [.1,.3,.6,.9] We found the best accuracy = 0.965 when the number of estimators = 100 and the learning rate = 0.3

```python
from sklearn.ensemble import GradientBoostingClassifier
best_acc = 0
best_nEst= 0
# best_pred = []
for nEst in [10, 100, 150, 200]:
  estimator = GradientBoostingClassifier(n_estimators=nEst, random_state=2022)
  estimator.fit(X_train, y_train)
  y_pred = estimator.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  if acc > best_acc:
    best_acc = acc
    best_nEst = nEst
    # best_pred = y_pred
print("best number of estimators  = ",best_nEst)
best_acc = 0
best_learning_rate= 0
best_pred = []
for lRate in [.1, .3, .6, .9]:
  estimator = GradientBoostingClassifier(n_estimators=best_nEst, learning_rate=lRate, random_state=2022)
  estimator.fit(X_train, y_train)
  y_pred = estimator.predict(X_test)
  acc = accuracy_score(y_test, y_pred)
  if acc > best_acc:
    best_acc = acc
    best_learning_rate = lRate
    best_pred = y_pred

print(f'the best accuracy score is = {best_acc} when number of estimators = {best_nEst} and learning rate ={best_learning_rate}')
report = classification_report(y_test, best_pred)
print(report)
draw_cm(y_test, best_pred)
```

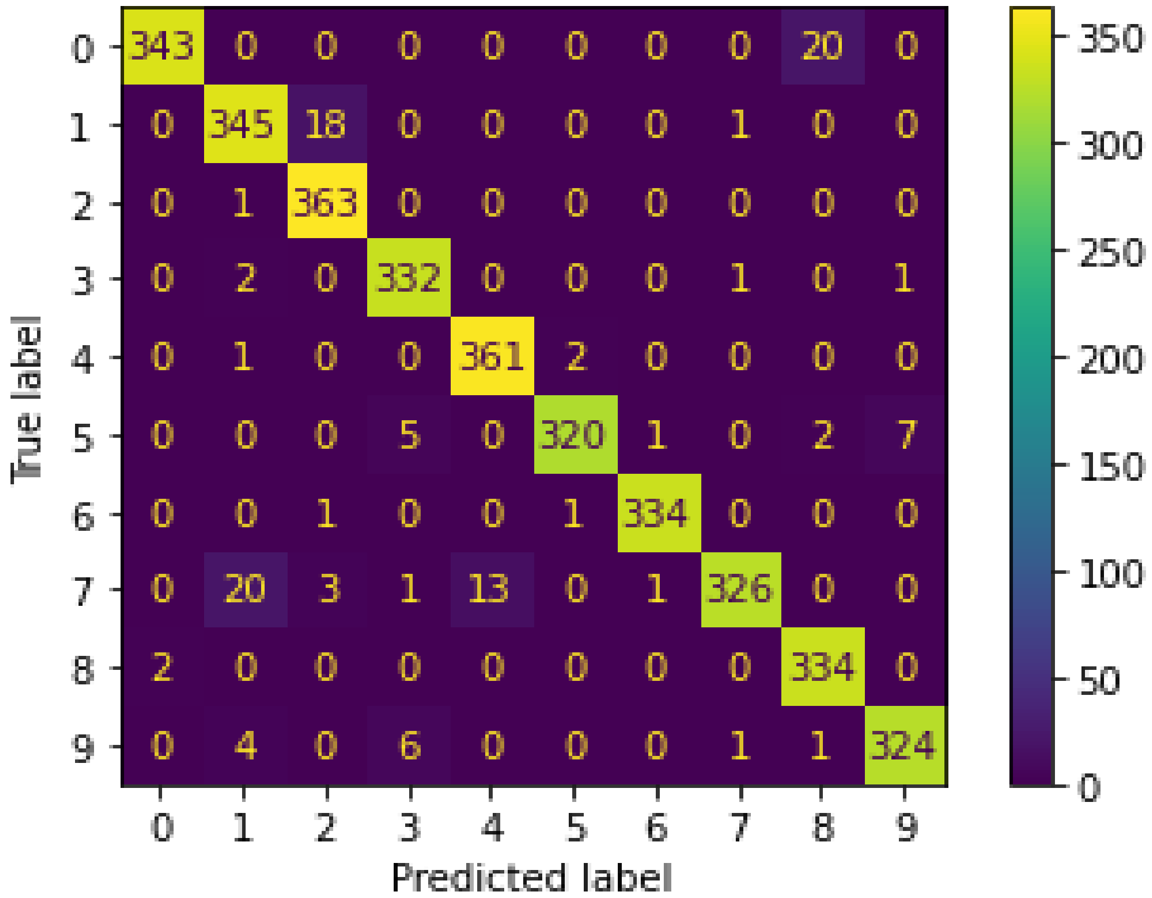
```
the best accuracy score is = 0.9651229273870783 when number of estimators = 100 and learning rate =0.3
              precision    recall  f1-score   support

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

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



B) XGBoos Classifier with the same best hyper parameters
(number of estimators =100 and learning rate = 0.3)
The accuracy = 0.9668

```
from xgboost import XGBClassifier
estimator = XGBClassifier(n_estimators=best_nEst, learning_rate=best_learning_rate, random_state=2022)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f'the XGBClassifier accuracy score is = {acc} with number of estimators = {best_nEst} and learning rate ={best_learning_rate}')
report = classification_report(y_test, y_pred)
print(report)
draw_cm(y_test, y_pred)
```

```
the XGBClassifier accuracy score is = 0.9668381932532876 with number of estimators = 100 and 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      1.00      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       0.99      0.99      0.99       336
           7       0.99      0.90      0.94       364
           8       0.94      0.99      0.96       336
           9       0.98      0.96      0.97       336

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



C) The accuracy for XGBoost = 96.68 which is slightly higher than the accuracy of the GradientBoosting = 96.51.
XGBoost has high predictive power and is almost 10 times faster than GradientBoosting

➔ GradientBoosting-> precision: there are more values = 1.0 for classes (1, 4, 6) which are higher than XGBoost but the low values in GradientBoosting = (0.91, 0.93, 0.93), meanwhile the values in XGBoost = (0.92, 0.94, 0.98) respectively for classes 1, 8 and 9.
On Average, they are equal for the precision, recall, and F1-score.

➔ It is much easier to use the accuracy metric for the comparison as here we have many classes (10 classes) to track the performance of the confusion matrix and comparing the performance between GradientBoosting and XGBoost for each class is really hard.

➔ Bagging is a parallel homogenous ensemble learning used to solve the high variance problem. Meanwhile, Boosting is a sequential homogeneous used to solve the high bias problem.

➔ Comparing the accuracies between Boosting (XGBoost) and bagging (decision tree) as the default base learner for XGBoost is the decision tree.

- Boosting (XGBoost): 96.68. Number_estimators = 100
- Bagging (decision tree): 95.56. Number_ estimators = 120

As we can see the boosting technique gives slightly higher accuracy over the test data. But if we could choose the right base learner we would reach a higher accuracy, as when we used the Bagging with SVM model, the accuracy = 0.98