

Applied machine learning Group assignment 4

Team members-Group 8:

- Abdelrhman Gaber Youssef Saad Rezkallah
- Eman Metwally Mohammed Abood
- Basma Reda Shaban Abd-Elsalam Abd-Elwahab

Part 1: Calculation:

Table 1:

Weather	Temperature	Humidty	Wind	Hiking
(F1)	(F2)	(F3)	(F4)	(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

Hiking
$$\rightarrow$$
 P(yes) = $\frac{3}{10}$, P(No) = $\frac{7}{10}$

a-

Gini =
$$1 - \sum_{i=1}^{NC} (p)^2$$

Gini Index For Weather(F1):

$$Gini_{cloudy} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$

$$Gini_{Sunny} = 1 - (\frac{1}{4})^2 - (\frac{3}{4})^2 = \frac{3}{8}$$

$$Gini_{Rainy} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$

Gini Split For Weather =
$$\frac{3}{10} * \frac{4}{9} + \frac{3}{10} * \frac{4}{9} + \frac{4}{10} * \frac{3}{8} = \frac{5}{12}$$

Gini Index For Temperature (F2):

$$Gini_{Cool} = 1 - (\frac{0}{3})^2 - (\frac{3}{3})^2 = 0$$

$$Gini_{Hot} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$

$$Gini_{Mild} = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 = \frac{1}{2}$$

Gini Split For Temperature =
$$\frac{3}{10}*0 + \frac{3}{10}*\frac{4}{9} + \frac{4}{10}*\frac{1}{2} = \frac{1}{3}$$

Gini Index For Humidity (F3):

$$\begin{aligned} \textit{Gini}_{Normal} &= 1 - (\frac{1}{4})^2 - (\frac{3}{4})^2 = \frac{3}{8} \\ \textit{Gini}_{High} &= 1 - (\frac{2}{6})^2 - (\frac{4}{6})^2 = \frac{4}{9} \\ \textbf{Gini Split For Humidity} &= \frac{4}{10} * \frac{3}{8} + \frac{6}{10} * \frac{4}{9} = \frac{5}{12} \end{aligned}$$

Gini Index For Wind(F4):

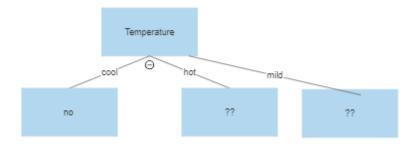
$$Gini_{Weak} = 1 - (\frac{2}{4})^2 - (\frac{2}{4})^2 = \frac{1}{2}$$

$$Gini_{Strong} = 1 - (\frac{1}{6})^2 - (\frac{5}{6})^2 = \frac{5}{18}$$
Gini Split For Wind= $\frac{4}{10} * \frac{1}{2} + \frac{6}{10} * \frac{5}{18} = \frac{11}{30}$

After computing Gini Split, the minimum value of Gini index is for feature Temperature, So, We Chose Temperature as a root node.

For the "Cool" branch

The Gini index for Weather, Humidity and wind are Equal Zero With Cool, this mean that the "Cool" branch is pure.



For the "Hot" branch

weather	Temperature	Humidity	Wind	Hiking
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Strong	No

The Gini Index For Weather (F1):

$$Gini_{Sunny} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$

Gini Split For Weather = $\frac{3}{3}*\frac{4}{9} = \frac{4}{9}$

Gini Index For Humidity(F3):

$$Gini_{High} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$

Gini Split For Humidity=
$$\frac{3}{3}*\frac{4}{9}=\frac{4}{9}$$

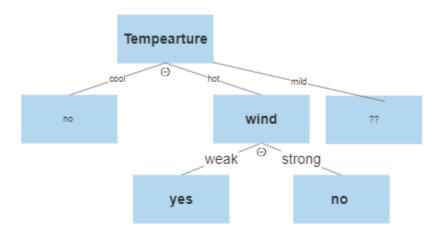
Gini Index For Wind(F4):

$$Gini_{Weak} = 1 - (\frac{1}{1})^2 = 0$$

$$Gini_{Strong} = 1 - (\frac{2}{2})^2 = 0$$

$$Gini \ Split \ For \ Wind = \frac{1}{1}*0 + \frac{2}{2}*0 = 0$$

From the results the minimum value of Gini index is "Wind" so it will be chosen as the child node for "hot" branch.



For the "Mild" branch

weather	Temperature	Humidity	Wind	Hiking
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

The Gini Index For Weather (F1):

$$\begin{aligned} &\textit{Gini}_{Rainy} = 1 - (\frac{1}{1})^2 = 0 \\ &\textit{Gini}_{Sunny} = 1 - (\frac{1}{1})^2 = 0 \\ &\textit{Gini}_{cloudy} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2} \\ &\textit{Gini Split For Weather} = \frac{1}{4}*0 + \frac{2}{4}*\frac{1}{2} + \frac{1}{4}*0 = \frac{1}{4} \end{aligned}$$

Gini Index For Humidity(F3)

$$Gini_{Normal} = 1 - (\frac{1}{1})^2 = 0$$

$$Gini_{High} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$$
Gini Split For Humidity= $\frac{1}{4}*0 + \frac{3}{4}*\frac{4}{9} = \frac{1}{3}$

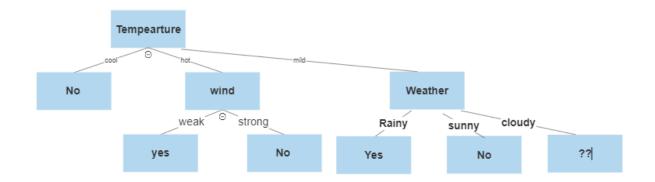
Gini Index For Wind(F4)

$$Gini_{Weak} = 1 - (\frac{1}{1})^2 = 0$$

 $Gini_{Strong} = 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9}$

Gini Split For Wind=
$$\frac{1}{4}$$
*0 + $\frac{3}{4}$ * $\frac{4}{9}$ = $\frac{1}{3}$

From the results the minimum value of Gini index is "Weather" so it will be chosen as the child node for "Mild" branch.



For the "Cloudy" branch

weather	Temperature	Humidity	Wind	Hiking
Cloudy	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

Gini Index For Humidity(F3):

$$Gini_{High} = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = \frac{1}{2}$$

Gini Split For Humidity= $\frac{2}{2}*\frac{1}{2}=\frac{1}{2}$

Gini Index For Wind(F4)

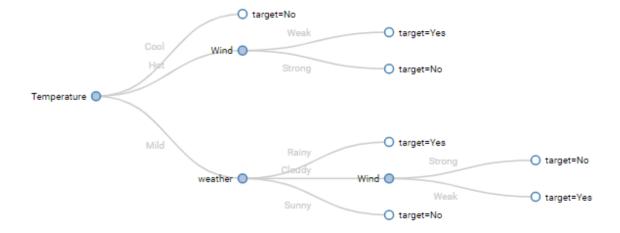
$$Gini_{Weak} = 1 - (\frac{1}{1})^2 = 0$$

 $Gini_{Strong} = 1 - (\frac{1}{1})^2 = 0$

Gini Split For Wind=
$$\frac{1}{2}$$
*0 + $\frac{1}{2}$ *0 = 0

From the results the minimum value of Gini index is "Wind" so it will be chosen as the child node for "Cloudy" branch.

The final decision tree based on the Gini index split:



b-

The Entropy of "hiking" labels

Entropy =
$$\frac{-3}{10} \log_2 \frac{3}{10} - \frac{7}{10} \log_2 \frac{7}{10} = 0.88$$

Information gain for weather (f1)

P(Cloudy and Hiking) =
$$\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

P(Sunny and Hiking) = $\frac{-1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} = 0.81$
P(Rainy and Hiking) = $\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$

Gain (Hiking, Weather (F1)) =
$$0.88 - \frac{3}{10} * 0.918 - \frac{4}{10} * 0.81 - \frac{3}{10} * 0.918 = 0.097$$

Information gain for Temperature (f2)

P(Cool and Hiking) =
$$\frac{0}{3} \log_2 \frac{0}{3} - \frac{3}{3} \log_2 \frac{3}{3} = 0$$

P(Hot and Hiking) = $\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$
P(Mild and Hiking) = $\frac{-2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$

Gain (Hiking, Temperature(F2)) =
$$0.88 - \frac{3}{10} * 0 - \frac{4}{10} * 0.918 - \frac{4}{10} * 1 = 0.2$$

Information gain for Humidity(F3)

P(High and Hiking) =
$$\frac{-2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} = 0.918$$

P(Normal and Hiking) = $\frac{-1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} = 0.81$

Gain (Hiking, Humidity(F3)) =
$$0.88 - \frac{4}{10} * 0.81 - \frac{6}{10} * 0.918 = 0.006$$

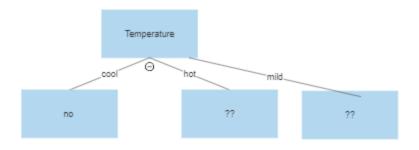
Information gain for Wind(F4)

P(Strong and Hiking) =
$$\frac{-1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = 0.65$$

P(Weak and Hiking) = $\frac{-2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$

Gain (Hiking, Wind(F4)) =
$$0.88 - \frac{4}{10} * 1 - \frac{6}{10} * 0.65 = 0.09$$

From the results the maximum value of Information Gain is "Temperature" so it will be chosen as the Root node.



For the "Cool" branch

The Information gain for Weather, Humidity and wind are Equal Zero With Cool, this mean the "Cool" branch was pure.

For the "Hot" branch

weather	Temperature	Humidity	Wind	Hiking
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Strong	No

The Entropy =
$$\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

Information gain for weather (f1)

P(Sunny and Hiking) =
$$\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

Gain (Hiking, Weather (F1)) = $0.918 - \frac{3}{3}*0.918 = 0$

Information gain for Humidity(F3)

P(High and Hiking) =
$$\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

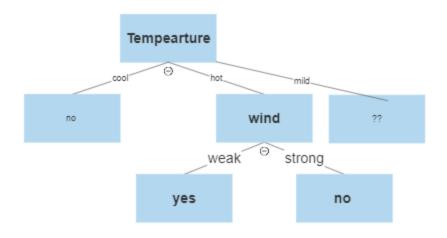
Gain (Hiking, Humidty(F3)) = $0.918 - \frac{3}{3} * 0.918 = 0$

Information gain for Wind(F4)

P(Strong and Hiking) =
$$-\frac{2}{2}\log_2\frac{2}{2} = 0$$

P(Weak and Hiking) = $\frac{-1}{1}\log_2\frac{1}{1} = 0$
Gain (Hiking, Wind(F4)) = $0.918 - \frac{1}{3}*0 - \frac{2}{3}*0 = 0.918$

From the results the maximum value of Information Gain is "Wind" so it will be chosen as the child node for "Hot" branch.



For the "Mild" branch

weather	Temperature	Humidity	Wind	Hiking
Rainy	Mild	Normal	Strong	Yes
Cloudy	Mild	High	Strong	No
Sunny	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

The Entropy =
$$\frac{-2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$$

Information gain for weather (f1)

P(Cloudy and Hiking) = $\frac{-1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$ P(Sunny and Hiking) = $\frac{-1}{1} \log_2 \frac{1}{1} = 0$ P(Rainy and Hiking) = $\frac{-1}{1} \log_2 \frac{1}{1} = 0$ Gain (Hiking, Weather (F1)) = $1 - \frac{2}{4} * 1 - \frac{1}{4} * 0 - \frac{1}{4} * 0 = 0.5$

Information gain for Humidity(F3)

P(High and Hiking) =
$$\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$$

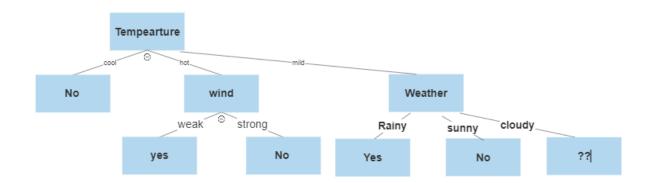
P(Normal and Hiking) = $\frac{-1}{1}\log_2\frac{1}{1} = 0$
Gain (Hiking, Humidity(F3)) = $1 - \frac{3}{4}*0.918 - \frac{1}{4}*0 = 0.31$

Information gain for Wind(F4)

P(Strong and Hiking) =
$$\frac{-1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

P(Weak and Hiking) = $\frac{-1}{1} \log_2 \frac{1}{1} = 0$
Gain (Hiking, Wind(F4)) = $1 - \frac{3}{4} * 0.918 - \frac{1}{4} * 0 = 0.31$

From the results the maximum value of Information Gain is "Weather" so it will be chosen as the child node for "Mild" branch.



For the "Cloudy" branch

weather	Temperature	Humidity	Wind	Hiking
Cloudy	Mild	High	Strong	No
Cloudy	Mild	High	Weak	Yes

The Entropy =
$$\frac{-1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

Information gain for Humidty(F3)

P(High and Hiking) = $\frac{-1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$

Gain (Hiking, Humidty(F3)) = $1 - \frac{1}{2} * 1 = 0$

Information gain for Wind(F4)

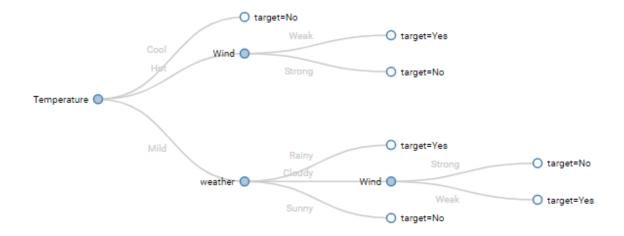
P(Strong and Hiking) = $\frac{-1}{1} \log_2 \frac{1}{1} = 0$

P(Weak and Hiking) = $\frac{-1}{1} \log_2 \frac{1}{1} = 0$

Gain (Hiking, Wind(F4)) = $1 - \frac{1}{2} * 0 - \frac{1}{2} * 0 = 1$

From the results the maximum value of Information Gain is "Wind" so it will be chosen as the child node for "Cloudy" branch.

The final decision tree based on the Information gain split:



C-

Advantages of Gini inde	ex Vs. Information gain
Gini index	Information gain
1- Used by CART algorithms	1- Used in ID3, C4.5 algorithms
2- computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of Gini Coefficient.	2- computes the difference between entropy before and after split and specifies the impurity in class elements.
3- can handle the values that are non- negative because it is measured by subtracting the sum of squared probabilities of each class from one.	3- measures the entropy differences before and after splitting and depicts the impurity in class variables.
4- Facilitates larger distributions that are very easy to implement.	4- Supports smaller distributions with smaller numbers and more specific values
Disadvantages of Gini inc	lex Vs. Information gain
Gini index	Information gain
1- prone to systematic and random data errors. Therefore, inaccurate data can distort the validity of the coefficient.	1- supports smaller partitions (distribution) with a variety of different values.
2- operates on the categorical target variables in terms of "success" or "failure" and performs only binary split.	2- can't handle the values that are non- positive.

Part 2: programming

Import Important Libraries and load the dataset

```
import pandas as pd #for working with dataframes
import numpy as np #to deal with arrays
import matplotlib.pyplot as plt #for plotting
import seaborn as sns
%matplotlib inline
```

Load the training dataset:

```
columns = ['pixel ' + str(i) for i in range(17)]
columns[-1] = 'label'
#load the train dataset
train = pd.read_csv('/content/pendigits-tra.csv',names=columns)
train.head()
```

	pixel 0	pixel 1	pixel 2	pixel 3	pixel 4	pixel 5	pixel 6	pixel 7	pixel 8	pixel 9	pixel 10	pixel 11	pixel 12	pixel 13	pixel 14	pixel 15	label
0	47	100	27	81	57	37	26	0	0	23	56	53	100	90	40	98	8
1	0	89	27	100	42	75	29	45	15	15	37	0	69	2	100	6	2
2	0	57	31	68	72	90	100	100	76	75	50	51	28	25	16	0	1
3	0	100	7	92	5	68	19	45	86	34	100	45	74	23	67	0	4
4	0	67	49	83	100	100	81	80	60	60	40	40	33	20	47	0	1

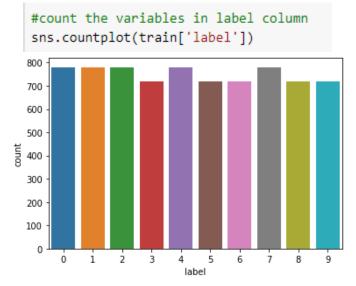


Get some information about the training dataset:

#get some information about the train dataset train.info()

```
<<class 'pandas.core.frame.DataFrame'>
    RangeIndex: 7494 entries, 0 to 7493
   Data columns (total 17 columns):
    # Column Non-Null Count Dtype
    ---
                  -----
       pixel 0 7494 non-null
                               int64
                7494 non-null
       pixel 1
       pixel 2 7494 non-null int64
    3 pixel 3 7494 non-null int64
       pixel 4 7494 non-null int64
       pixel 5
                  7494 non-null
       pixel 6 7494 non-null int64
       pixel 7 7494 non-null int64
    8 pixel 8 7494 non-null int64
       pixel 9
                               int64
                 7494 non-null
    10 pixel 10 7494 non-null
                                 int64
    11 pixel 11 7494 non-null
                               int64
    12 pixel 12 7494 non-null int64
    13 pixel 13 7494 non-null int64
    14 pixel 14 7494 non-null
15 pixel 15 7494 non-null
16 label 7494 non-null
                               int64
                                 int64
                 7494 non-null int64
   dtypes: int64(17)
   memory usage: QQ5 / KR
```

Count the numbers occurrence in the label column:



Split data into x_train and y_train:

#split the data into x_train and y_train
x_train = train.drop('label',axis= 1)
y_train = train['label']

Load the test dataset:

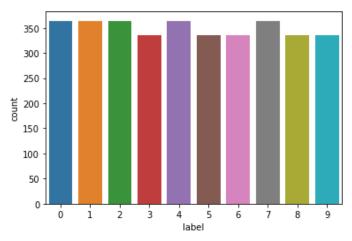
```
[6] columns = ['pixel ' + str(i) for i in range(17)]
    columns[-1] = 'label'
    #load the test dataset
    test = pd.read_csv('/content/pendigits-tes.csv',names=columns)
    test.head()
```

Ð pixel 0 pixel 1 pixel 2 pixel 3 pixel 4 pixel 5 pixel 6 pixel 7 pixel 8 pixel 9 pixel 10 pixel 11 pixel 12 pixel 13 pixel 14 pixel 15 label



Count the numbers occurrence in label column:

#count the numbers occurrence in label column
sns.countplot(test['label'])



Split the test data into x_test and y_test:

```
[8] #split the test dataset into x_test and y_test
x_test = test.drop('label',axis= 1)
y_test = test['label']
```

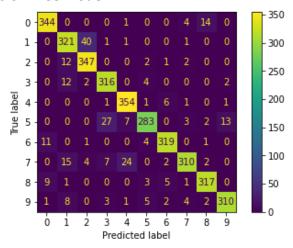
Train the Decision Tree classification Model:

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.

Accuracy of Decision Tree model: Accuracy 0.9208118925100057 Classification report of Decision Tree model:

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

Confusion matrix of Decision Tree model:



Train SVM model:

SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

```
from sklearn.svm import SVC

#train SVM model

svm = SVC(random_state=0)

svm.fit(x_train,y_train)

y_pred = svm.predict(x_test)
```

Accuracy score of SVM model: Accuracy 0.9817038307604345

Classification report and Confusion matrix of SVM model:

	precision	recall	f1-score	support		0 -	353	0	0	0	0	0	0	0	10	0	- 350
0	1.00	0.97	0.99	363			_	25.0	12				0				
1	0.95	0.96	0.96	364		1 -	0	350	13	0	1	0	U	0	0	0	- 300
2	0.96	0.99	0.98	364		2 -	0	2	362	0	0	0	0	0	0	0	
3	0.99	0.99	0.99	336				-	502		10						- 250
4	1.00	0.99	0.99	364		3 -	0	1	0	333	0	0	0	0	0	2	250
5	0.99	0.98	0.98	335	<u>a</u>						250			0			200
6	1.00	1.00	1.00	336	label	4 -	0	0	0	0	359	4	1	U	0	0	- 200
7	0.99	0.95	0.97	364	<u>r</u>	5 -	0	0	0	4	0	329	0	0	0	2	
8	0.97	1.00	0.98	336	乒	_						22.5	Ľ	Ň	Ĭ	_	- 150
9	0.98	0.98	0.98	336		6 -	0		0		0		336	0	0	0	
						_	0	12		0	0	0	0	347	0	4	- 100
accuracy			0.98	3498		7 -	٥	12	1	v	·	v	·	347	U	. *	
macro avg	0.98	0.98	0.98	3498		8 -	0	0	0	0	0	1	0	0	335	0	
weighted avg	0.98	0.98	0.98	3498								-				_	- 50
						9 -	0	2	0		0		0	3	1	330	
							_	•		_	٠.	Ţ	Ţ		_	<u> </u>	0
							0	1	2	3	4	5	6	7	8	9	
										Pre	dicte	ed la	bel				

Apply Hard and soft Voting Classifiers:

Hard voting entails picking the prediction with the highest number of votes.

soft voting entails combining the probabilities of each prediction in each model and picking the prediction with the highest total probability.

```
from sklearn.ensemble import VotingClassifier
#train the hard voting classifier
hard_votting = VotingClassifier(estimators=[('dt', dt), ('svm', svm)], voting='hard')
hard_votting.fit(x_train, y_train)
y_pred = hard_votting.predict(x_test)

[22] #train the soft voting classifier
svm = SVC(probability=True)
soft_votting = VotingClassifier(estimators=[('dt', dt), ('svm', svm)], voting='soft')
soft_votting.fit(x_train, y_train)
y_pred = soft_votting.predict(x_test)
```

Acc	curacy of	hard v	oting		Accuracy of soft voting						
0.9	405374	4997	14122		0.9	92081:	18925	10005	7		
Classifica	ation rep	ort of	hard vo	ting	Classific	cation r	eport o	of soft v	oting		
р	precision re		f1-score	support	р	recision	recall	f1-score	support		
0 1 2 3 4 5 6 7 8 9 accuracy macro avg	0.94 0.85 0.95 0.89 0.93 0.95 0.97 0.98 0.97 0.99	0.98 0.97 0.96 0.95 0.89 0.95 0.85 0.94 0.92	0.96 0.91 0.96 0.92 0.96 0.92 0.96 0.91 0.95 0.95	363 364 364 336 364 335 336 364 336 336 3498 3498 3498	0 1 2 3 4 5 6 7 8 9 accuracy macro avg weighted avg	0.94 0.87 0.88 0.89 0.91 0.94 0.95 0.95 0.95	0.95 0.88 0.95 0.94 0.97 0.84 0.95 0.85 0.94 0.92	0.95 0.88 0.92 0.91 0.94 0.89 0.95 0.90 0.94 0.92 0.92	363 364 364 336 364 335 336 364 336 336 3498 3498		
Confus	ion matri	ix of h	ard voti	ng	Confu	sion ma	atrix of	soft vo	ting		
1 - 0 353 1 2 - 0 14 3 3 - 0 12 4 - 0 0 5 - 0 0 6 - 11 0 7 - 0 24 8 - 9 1 9 - 1 9	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	8 0 0 0 0 0 0 2 0 0 0 1 0 0 0 0 316 0 2 309 8 9	- 350 - 300 - 250 - 200 - 150 - 100 - 50	0 - 344 0 1 - 0 321 2 - 0 12 3 - 0 12 9 4 - 0 0 9 5 - 0 0 6 - 11 0 7 - 0 15 8 - 9 1 9 - 1 8 0 1	0 0 1 40 1 1 347 0 0 2 316 0 0 1 35. 0 27 7 1 0 0 4 7 24 0 0 0 0 3 1	283 0 4 319	4 14 0 1 0 0 2 0 0 0 0 2 1 0 1 3 2 13 0 1 0 310 2 0 1 317 0 4 2 310 7 8 9	- 350 - 300 - 250 - 200 - 150 - 100 - 50		

Apply Bagging Classifier in range[10:200]:

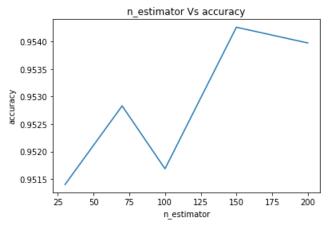
A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

```
from sklearn.ensemble import BaggingClassifier
scores =[]
for i in [30,70,100,150,200]:
  ba = BaggingClassifier(n_estimators=i,random_state=0)
  ba.fit(x_train,y_train)
  y_pred = ba.predict(x_test)
  print('Accuracy ', accuracy_score(y_test,y_pred))
  print(classification_report(y_test,y_pred))
  plot_confusion_matrix(ba,x_test,y_test)
  scores.append(accuracy_score(y_test,y_pred))
```

Accuracies	Cl				Co	nf	usi	on										
0.951400800457	,	precision	recall	f1-score	support		0 -	349	0	0	0	۸	۸	۸	۸	1//	0 0 0 0 0 0 0 2 0 0 0 0 0 1 0 2 0 2 3 12 1 0 2 0 330 0 1 321	
4043							-											
	0	0.98	0.96	0.97	363		1 -	0	337	24	1	1	0	0	1	0	0	
	1	0.90	0.93	0.91	364		2 -	0	3	357	1		1		2		0	
	2	0.93	0.98	0.95	364		3 -	0	4	0	330	۸	۸	۸	٥	٥	2	
	3	0.94	0.98	0.96	336	<u></u>						•			۰			
	4	0.95	0.98	0.96	364	ape	4 -	0	1	0	2	355	2	2	0	0	2	
	5	0.97	0.90	0.93	335	Fue label	5 -	0	1	0	10	8	301	0	2	3	10	
	6	0.99	0.97	0.98	336	卢												ı
	7 8	0.98	0.89 0.98	0.93 0.96	364		6 -	2	0			2	4	327	0	1	0	ı
	8	0.94 0.96	0.98	0.95	336 336		7 -	0	23	3	5	7			324	2	0	
	9	0.90	0.95	0.95	550		8 -	5	1	0	0	0	1	0	0	329	0	ı
	accuracy			0.95	3498				÷		Ĭ		Ō					
	macro avg	0.95	0.95	0.95	3498		9 -	0	6	0	1	2	2	1	3	2	319	
	weighted avg	0.95	0.95	0.95	3498			ò	í	2	3	4	5	6	7	8	9	
	weighted dvg	0.55	0.55	0.55	5450						Pre	dicte	d la	bel				
0.952830188679	,	precision	recall	f1-score	support		۸	349	0	0	0	۸	٥	0	_	1	4 0	
2453							U	345	Ľ	ď	٠		٠	٠	٠	' 1	ď	
	0	0.99	0.96	0.97	363		1	0	335	25	2	1	0	0	1	. () 0	
	1	0.89	0.92	0.91	364		2	0	3	356	2	0	0	0	2) 1	
	2	0.93	0.98	0.95	364				ı,									
	3	0.94	0.98	0.96	336	_	3	0	5	0	329	9 0	_ 0	0	0) () 2	
	4	0.96	0.98	0.97	364	Fue label	4	0			1	355	5 3	3	0) () 2	
	5	0.98	0.90	0.94	335	<u> </u>	5	0	0	0	9	Q	301	0	,		12	
	6	0.99	0.99	0.99	336	골	Э.				,		50.				, 12	
	7	0.98	0.89	0.93	364		6	1					2	33	2 0)]	. 0	
	8	0.94	0.98	0.96	336		7	0	27	2	5	3	0	0	32	5	2 0	
	9	0.95	0.96	0.95	336							Ž		Ĭ	22			
							8	4	1	0	0	0	1	0	0	33	0	
	accuracy			0.95	3498		9	0	5		3	2	0	1	. 3	1	321	
	macro avg	0.95	0.95	0.95	3498			0	i	2	3	,	Ļ	į			,	
	weighted avg	0.95	0.95	0.95	3498			U	1	2	_	4 edict	_	_) 9	
											Γľ	cuici	ieu I	ane	ı			

0.951686678101																
7724		precision	recall	f1-score	support	0	-349	0	0	0	0	0	0		14	0
						1	0	336	25	1	1			1		0
	0	0.99	0.96	0.97	363	2	0	4	355	2	0	1	0	2	0	0
	1	0.88	0.92	0.90	364		U	7	333	۷.		1	٠.	-	٧	٠
	2	0.92	0.98	0.95	364	3	0	5	0	329						2
	3	0.95	0.98	0.96	336	lag 4	0	0	0	1	356	4	3	0	0	0
	4	0.96	0.98	0.97	364	<u>a</u>				٠, '		20.7				
	5	0.97	0.90	0.93	335	<u>P</u> 5	- 0	0	0	7	8	301	0	0	5	14
	6	0.99	0.98	0.99	336	6	- 1	1			1	2	330		1	0
	7	0.98	0.88	0.93	364	7	- 0	28	5	4	3	0	0	322	2	0
	8	0.93	0.98	0.96	336									-		
	9	0.95	0.96	0.95	336	8		1		0		1		0	330	0
	accuracy			0.95	3498	9	0	5	0	2	2	1	1	3	1 3	21
	macro avg	0.95	0.95	0.95	3498		Ò	1	2	3	4	5	6	7	8	9
	weighted avg	0.95	0.95	0.95	3498					Pred	dicte	d lab	el			
0.954259576901		precision	recall	f1-score	support	^	-349	0	0	0	0	0	0	0	14	0
0863																
	0	0.99	0.96	0.97	363	1	- 0	338	23	1	1	0	0	1		0
	1	0.88	0.93	0.90	364	2	0	4	357					2		1
	2	0.93	0.98	0.95	364	3	- 0	5	0	329	0	0	0	0	0	2
	3	0.97	0.98	0.97	336											
	4	0.96	0.98	0.97	364	<u>q</u> 4	0		1	0	355	4	4	0		0
	5	0.98	0.91	0.94	335	Fue label	- 0			5	8	304			5 1	13
	6	0.99	0.98	0.98	336	<u>اس</u> 6	1	0	0	0	1	2	330	0	1	1
	7	0.98	0.89	0.93	364								-			
	8	0.93	0.98	0.96	336	7	0	30	3	3	2		0	324	2	0
	9	0.95	0.96	0.95	336	8	4	1				1		0 3	330	0
						9	0	5	0	2	2	0	1	3	1 3	22
	accuracy			0.95	3498		_	,	-	-	1	_	ċ	7		9
	macro avg	0.96	0.95	0.95	3498		0	1	2	3 Dred	4 dicte	5 d lab	6 el	7	8	9
	weighted avg	0.96	0.95	0.95	3498					1100	JICCC.	u iui				
0.953973699256		precisi	ion r	ecall f	1-score	0	-351	0	0	0	0	0	0	0	12	0
7182										Ĭ,						
			99	0.97	0.98	1	- 0	337	24	1	1	0	0	1	0	0
			89	0.93	0.91	2	- 0	4	356	1	0		0	2	0	1
			92	0.98	0.95	3	- 0	5	0	329	0	0	0	0	0	2
			96	0.98	0.97	_				_						
			96	0.98	0.97	Fue label	- 0	0	2	0	355	4	. 3	0	0	0
			.97 .99	0.90 0.99	0.94 0.99	g 5	- 0	0	0	5	9	303	0	0	5	13
			98	0.89	0.99		- 0	0	0	0	0	2	332	0	1	1
			94	0.98	0.95											
			95	0.95	0.95	7	- 0	27	3	5	3	0	0	324	2	0
		. 0.		0.55	0.55	8	4	1	0	0	0	1	0	0	330	0
	accurac	у			0.95	9	- 0	5	0	3	2	1	1	3	1	320
	macro av		96	0.95	0.95		Ó	i	2	3	4	5	6	'n	8	9
	weighted av	g 0.	95	0.95	0.95		U	1	2	_		ed la		,	0	3
											.arct	-4 10				

Plot the numbers of estimators vs. accuracy score:



So, the highest accuracy at n = 150

Train bagging classifier with the best number of estimators:

Accuracy is: 0.9542595769010863

Classification report and confusion of bagging with estimator number = 150:

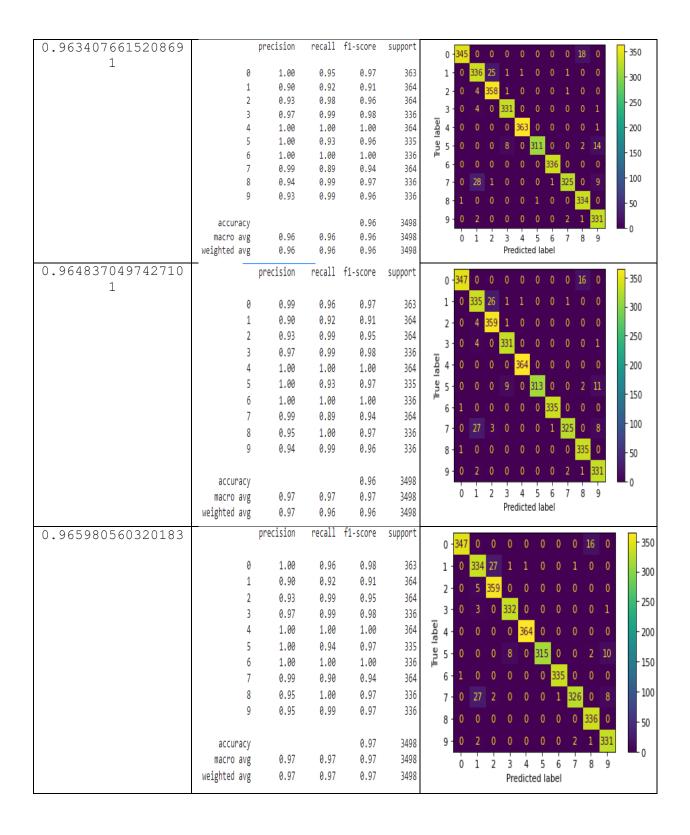
	precision	recall	f1-score	support		0 -	349	0	0	0	0	0	0	0	14	0		- 350
0	0.99	0.96	0.97	363		1 -	0	338	23	1	1			1				- 300
1	0.88	0.93	0.90	364		2 -	0	4	357	0	0	0	0	2	0			
2	0.93	0.98	0.95	364		2	۰	7	357	v	ıŭ	۰	۰	-	۰	•		- 250
3	0.97	0.98	0.97	336		3 -		5		329	0					2		
4	0.96	0.98	0.97	364	label	1 -	٥	٥	1	0	355	4	4	0	0	0		- 200
5	0.98	0.91	0.94	335		7	۰	۰	•	ď	333	_	Α,	۰	۰	ď		200
6	0.99	0.98	0.98	336	Γue	5 -				5	8	304	0		5	13		- 150
7	0.98	0.89	0.93	364	<u></u>	6 -	1	٥	0	0	1	2	330	0				150
8	0.93	0.98	0.96	336		0	1	ď				-	550	ď	٠.	•		100
9	0.95	0.96	0.95	336		7 -		30	3	3	2			324	2			- 100
						8 -	4	1	0	0	0	1	0	0	330	0		- 50
accuracy			0.95	3498														30
macro avg	0.96	0.95	0.95	3498		9 -	0	5	0	2	2	0	1	3	1	322		
weighted avg	0.96	0.95	0.95	3498			Ó	1	2	3	4	5	6	7	8	9	'	0
							-	-	-	Pre	dict	ed la	bel		-	-		

Apply Random Forest classifier: (additional model code)

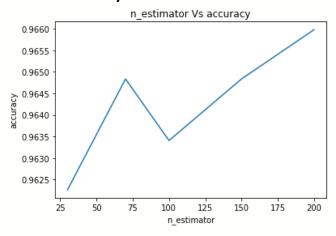
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

```
[35] from sklearn.ensemble import RandomForestClassifier
    scores =[]
    for i in [30,70,100,150,200]:
        rf = RandomForestClassifier(n_estimators=i,random_state=0)
        rf.fit(x_train,y_train)
        y_pred = rf.predict(x_test)
        #accuracy score for random forest
        print('Accuracy ', accuracy_score(y_test,y_pred))
        #classification report for random forest
        print(classification_report(y_test,y_pred))
        #confusion matrix for random forest
        plot_confusion_matrix(rf,x_test,y_test)
        scores.append(accuracy_score(y_test,y_pred))
```

Accuracies	Cla	ssificat	ion re	ports		Confusion matrices
0.962264150943396		precision		f1-score	support	0-344 0 0 0 0 0 0 19 0 -350
2	0	0.99	0.95	0.97	363	1 - 0 <mark>335</mark> 26 1 1 0 0 1 0 0 -300
	1	0.91	0.92	0.91	364	2-063561000100
	2	0.93	0.98	0.95	364	3 - 0 4 0 331 0 0 0 0 0 1
	3	0.97	0.99	0.98	336	
	4	1.00	0.99	1.00	364	8/4 0 0 0 0 362 0 0 0 0 2 -200 9/5 0 0 0 0 0 2 9 -150
	5	1.00	0.94	0.97	335	월 5 - 0 0 0 8 0 <mark>316</mark> 0 0 2 9
	6	1.00	1.00	1.00	336	6-0 0 0 0 0 0 336 0 0 0
	7 8	0.98 0.94	0.89 0.99	0.93	364 336	7 - 0 22 2 0 0 0 1 324 0 15 -100
	9	0.94	0.99	0.96 0.95	336	
	9	0.32	0.99	0.93	550	8 - 2 0 0 0 0 1 0 2 <mark>331</mark> 0 - ₅₀
	accuracy			0.96	3498	9 - 0 2 0 0 0 0 0 2 1 <mark>331</mark>
	macro avg	0.96	0.96	0.96	3498	0 1 2 3 4 5 6 7 8 9
	weighted avg	0.96	0.96	0.96	3498	Predicted label
0.964837049742710		precision	recall	f1-score	support	0- <mark>343</mark> 0 0 0 0 0 1 0 19 0 -350
1	0	1.00	0.94	0.97	363	1 - 0 335 26 1 1 0 0 1 0 0
	1	0.92	0.92	0.92	364	2 - 0 2 360 1 0 0 0 1 0 0
	2	0.93	0.99	0.96	364	- 250
	3	0.97	0.98	0.98	336	3 - 0 4 0 330 0 0 0 0 0 2
	4	1.00	1.00	1.00	364	실 4 - 0 0 0 0 <mark>363</mark> 0 0 0 0 1 - 200
	5	1.00	0.94	0.97	335	2 4 - 0 0 0 0 <mark>363</mark> 0 0 0 0 1 -200 2 5 - 0 0 0 8 0 <mark>315</mark> 0 0 2 10 -150
	6	0.99	1.00	1.00	336	
	7	0.99	0.90	0.94	364	6 - 0 0 0 0 0 0 <mark>336</mark> 0 0 0
	8	0.94	1.00	0.97	336	7 - 0 23 3 0 0 0 1 327 0 10 -100
	9	0.94	0.99	0.96	336	8-0 0 0 0 0 1 0 0 335 0 -50
						9-0 2 0 0 0 0 0 2 1 331
	accuracy			0.96	3498	0
	macro avg	0.97	0.97	0.97	3498	0 1 2 3 4 5 6 7 8 9
	weighted avg	0.97	0.96	0.96	3498	Predicted label



plot numbers of estimators Vs. accuracy scores:



So, the highest accuracy is at n = 200

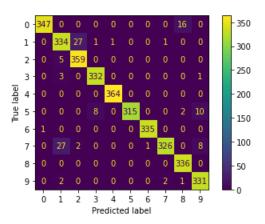
Train random forest with estimator = 200:

```
[36] rf = RandomForestClassifier(n_estimators=200,random_state=0)
    rf.fit(x_train,y_train)
    y_pred = rf.predict(x_test)
    print('Accuracy ', accuracy_score(y_test,y_pred))
    print(classification_report(y_test,y_pred))
    plot_confusion_matrix(rf,x_test,y_test)
```

Accuracy score: 0.965980560320183

Classification report and confusion matrix:

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

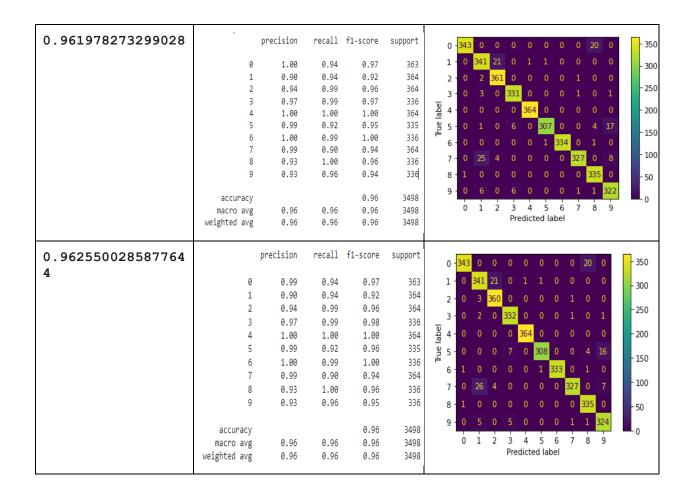


Train Boosting Classifier in range[10:200]:

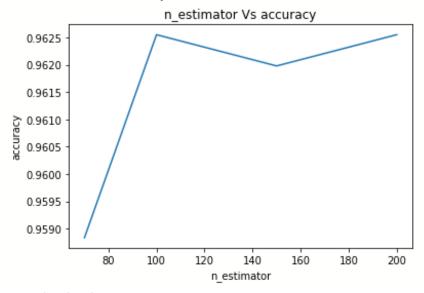
Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers.

```
from sklearn.ensemble import GradientBoostingClassifier
scores_1 =[]
for i in [70,100,150,200]:
   gb = GradientBoostingClassifier(n_estimators=i,random_state=0)
   gb.fit(x_train,y_train)
   y_pred = gb.predict(x_test)
   print('Accuracy ', accuracy_score(y_test,y_pred))
   print(classification_report(y_test,y_pred))
   plot_confusion_matrix(gb,x_test,y_test)
   scores_1.append(accuracy_score(y_test,y_pred))
```

Accuracies	Classific	ation re	ports			Co	nfu	sio	n ı	ma	tri	ces	6				
0.958833619210977		precision	recall	f1-score	support	C	34:	1 0	0	0	0	0	0	0	22	0	- 350
7	0 1 2 3 4 5 6 7 8 9 accuracy macro avg weighted avg	1.00 0.90 0.92 0.97 0.99 0.99 1.00 0.98 0.93 0.93	0.94 0.92 0.99 0.98 0.99 0.93 0.99 1.00 0.96	0.97 0.91 0.95 0.97 0.99 0.96 1.00 0.94 0.96 0.94	363 364 364 336 336 336 336 336 336 3498 3498 3498	Tue label	2 - 0 3 - 0 4 - 0 5 - 0 7 - 0	334 4 4 1 0 0 23 0 6	27 359 0 0 0 0 3 0	0 0 330 0 5 0 0	1 0 0 362 0 0 3 0 0	2 3 0 0 0	34 0 0 0	1 0 0 0 324 0	0 0 336	0 0 1 0 14 0 11 0	- 300 - 250 - 200 - 150 - 100 - 50
0.962550028587764	0 1 2 3 4 5 6 7 8 9 accuracy macro avg	precision 1.00 0.91 0.94 0.97 1.00 0.99 1.00 0.99 0.93 0.91	recall 0.94 0.93 0.99 0.99 1.00 0.93 0.99 0.90 1.00 0.96	f1-score 0.97 0.92 0.96 0.98 1.00 0.96 1.09 0.94 0.96 0.93 0.96 0.96	support 363 364 364 336 364 335 336 336 3498 3498 3498	1 2 3 4 4 5 6 6 7 8 8	343 - 0 - 0 - 0 - 0 - 0 - 0 - 0 - 0	340 3 1 0 0 19 0	0 22 360 0 0 0 0 0	0 0 0 3311 0 5 0 0 0	0 1 0 0 363 0 0 0	1 0 0 0 311 1 3 0	0 0 0 0 0 0 0 0 0 0	0 1 1 0 0 0 27 0 3	0 0 0 4 1 0		- 350 - 300 - 250 - 200 - 150 - 100 - 50



plot numbers of estimators Vs. accuracy scores:

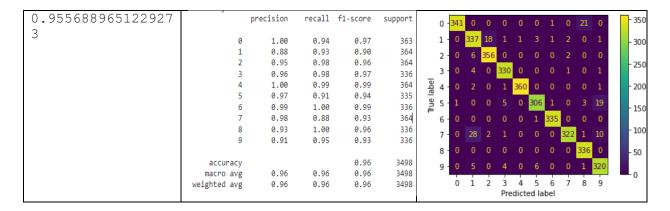


So, the highest accuracy score is at n estimator = 100

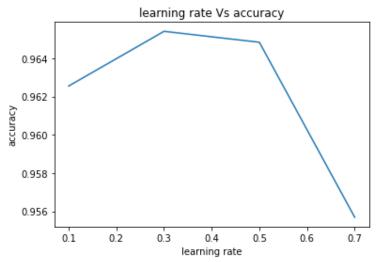
Train the boosting classifier at n estimator = 100 and with range [0.1:0.9]:

```
for i in [0.1,0.3,0.5,0.7]:
   gb = GradientBoostingClassifier(n_estimators=100,learning_rate=i,random_state=0)
   gb.fit(x_train,y_train)
   y_pred = gb.predict(x_test)
   print('Accuracy ', accuracy_score(y_test,y_pred))
   print(classification_report(y_test,y_pred))
   plot_confusion_matrix(gb,x_test,y_test)
   scores_2.append(accuracy_score(y_test,y_pred))
```

Accuracies	Classifica	tion re	ports			Confusion matrices
0.962550028587764	Ciassinica	precision	•	1-score	support	
4						0 943 0 0 0 0 0 0 20 0
4	0 1	1.00 0.91	0.94 0.93	0.97 0.92	363 364	1 - 0 340 22 0 1 1 0 0 0 0 - 300
	2	0.94	0.99	0.96	364	2-03 360 0 0 0 0 1 0 0
	3	0.97	0.99	0.98	336	3 - 0 3 0 <mark>331</mark> 0 0 0 1 0 1
	4	1.00	1.00	1.00	364	2 4 - 0 1 0 0 363 0 0 0 0 0 0 -200 2 5 - 0 0 0 5 0 311 0 0 4 15
	5 6	0.99 1.00	0.93 0.99	0.96 1.00	335 336	9 5 - 0 0 0 5 0 311 0 0 4 15
	7	0.99	0.99	0.94	364	150
	8	0.93	1.00	0.96	336	
	9	0.91	0.96	0.93	336	7 - 0 19 3 0 0 0 0 327 0 15
						8 - 0 0 0 0 0 0 0 <mark>336</mark> 0 _{- 50}
	accuracy macro avg	0.96	0.96	0.96 0.96	3498 3498	9-0 6 0 6 0 0 0 1 1 322
	weighted avg	0.96	0.96	0.96	3498	0 1 2 3 4 5 6 7 8 9
	merginee eng	0.50		0.50	2130	Predicted label
0.965408805031446	_	precision	recall	f1-score	support	0 - 343 0 0 0 0 0 0 0 20 0 - 350
5	0	1.00	0.94	0.97	363	1 - 0 346 15 0 1 1 0 1 0 0
	1	0.91	0.95	0.93	364	2 - 0 3 361 0 0 0 0 0 0 0
	2	0.95	0.99	0.97	364	- 250
	3	0.97	0.99	0.98	336	3 0 3 0 351
	4	1.00	1.00	1.00	364	음 4-0 0 0 0 <mark>363</mark> 1 0 0 0 0 -200
	5	0.99	0.93	0.96	335	9 5 - 0 0 0 7 0 310 0 0 4 14 - 150
	6 7	1.00	0.99	1.00 0.94	336	6-0 0 0 0 0 1 334 0 1 0
	8	0.99 0.93	0.90 1.00	0.94	364 336	· · · · · · · · · · · · · · · · · · ·
	9	0.93	0.97	0.95	336	7 - 0 25 3 0 0 0 0 328 0 8
	,	0.55	0.57	0.55	330	8 - 0 0 0 0 0 0 0 0 <mark>336</mark> 0 _{- 50}
	accuracy			0.97	3498	9-0 5 0 4 0 0 0 1 1 325
	macro avg	0.97	0.97	0.97	3498	0 1 2 3 4 5 6 7 8 9
	weighted avg	0.97	0.97	0.97	3498	Predicted label
0.964837049742710		precision	recall	f1-score	support	0 - 343 0 0 0 0 0 0 0 20 0 - 350
1	0	1.00	0.94	0.97	363	1 - 0 348 12 0 1 1 0 0 1 1
	1	0.89	0.96	0.92	364	2 - 0 2 362 0 0 0 0 0 0
	2	0.96	0.99	0.98	364	- 250
	3	0.97	0.99	0.98	336	
	4	1.00	1.00	1.00	364	을 4-0 0 0 0 <mark>363</mark> 1 0 0 0 0 -200
	5	0.99	0.91	0.95	335	-200 2 5 - 0 1 0 7 0 306 0 0 4 17
	6	1.00	1.00	1.00	336	
	7	0.99	0.89	0.94	364	
	8	0.93	1.00	0.96	336	, 0 31 3 2 0 0 0 323 0 3
	9	0.94	0.97	0.95	336	8 - 0 0 0 0 0 0 0 <mark>336</mark> 0 _{- 50}
	accuracy			0.96	3498	9-0 6 0 3 0 0 0 1 1 <mark>325</mark>
	macro avg	0.97	0.97	0.96	3498	0 1 2 3 4 5 6 7 8 9
	weighted avg	0.97	0.96	0.96	3498	Predicted label



Plot the learning rate Vs. accuracy scores:



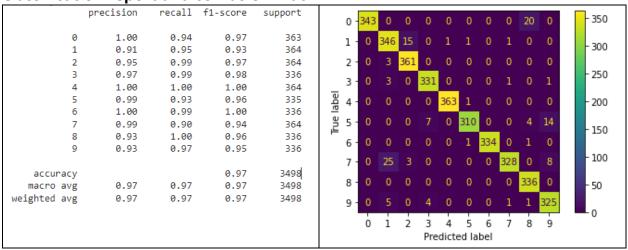
So, the best accuracy is when number of estimator = 100 and with learning rate = 0.3

Train Boosting classifier with n_estimators=100, learning_rate=0.3:

```
gb = GradientBoostingClassifier(n_estimators=100,learning_rate=0.3,random_state=0)
gb.fit(x_train,y_train)
y_pred = gb.predict(x_test)
print('Accuracy ', accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
plot_confusion_matrix(gb,x_test,y_test)
```

Accuracy score: 0.9654088050314465

Classification report and confusion matrix:



Apply XGBOOST classification model with n_estimators=100, learning_rate=0.3:

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.

Accuracy score: 0.9668381932532876

Classification report and confusion matrix:

-	precision	recall	f1-score	support		0 -	343	0	0	0	0	0	0	0	20	0	
0	1.00	0.94	0.97	363		1 -		343	18		1	1		1	0		
1	0.90	0.94	0.92	364		2 -	0	3	361	0	0	0	0	0	0	0	
2	0.95	0.99	0.97	364			0	3	0	331	0	0	0	,	0	,	
3	0.97	0.99	0.98	336	-	3 -	v	٥	U	221	U	٠,	v	1	U	1	
4	1.00	1.00	1.00	364	label	4 -					363	1			0		
5	0.99	0.96	0.98	335	<u>e</u>	5 -	0	0	0	6	0	323	0	1	0	5	
6	1.00	1.00	1.00	336	Tue	,	ŭ							ıŤ.			
7	0.99	0.89	0.93	364		6 -	0	0	0	0	0	1	335	0	0	0	
8	0.94	1.00	0.97	336		7 -		29	2					323	0	10	
9	0.95	0.97	0.96	336		8 -	1	0						0	335	0	
accuracy			0.97	3498		9 -		4		5				1	1	325	
macro avg	0.97	0.97	0.97	3498			ó	1	2	3	4	- 5	6	7	8	9	J
weighted avg	0.97	0.97	0.97	3498			U	1	2	Pre	dict	ed la	_	′	0	,	

Which metric is the best to compare performance, accuracy or the confusion matrix?

Accuracy performance metrics can be critical when dealing with imbalanced data and overfitting.

The confusion matrix, recall, precision, and F1 score gives good obviousness of prediction results comparing with the accuracy.

The F1-score can tell us if there is overfitting or not.

After seeing the F1-score, we notice that there is no overfitting in the data, so the accuracy is good here than the confusion matrix.

Conclusion:

During this assignment we learned how to apply different classification techniques, such as decision tree classifier, and apply bagging and boosting with the best estimator and learning rate.

After comparing the models with each other, we noticed that the **decision tree** model is the **lowest accuracy**, which is **0.92**, and the **SVM** model is the **highest accuracy**, which is **0.98**

After applying soft voting and hard voting, the hard voting is the best, which have accuracy = **0.94**, and the soft voting is the worst, which have accuracy = **0.92**.

After comparing bagging and boosting, the **best accuracy is boosting classifier**, which have **accuracy = 0.965**,

When comparing the XGBOOST model(with the best number of estimator and hyperparameter like the boosting classifier) with Gradient Boosting classifier, the accuracy of XGBOOST is the higher which ism = 0.9668

References:

https://scikit-

<u>learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html</u>

https://xgboost.readthedocs.io/en/stable/python/python_api.html

https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/#:~:text=The%20idea%20behind%20%22gradient%20boosting,Approximately%20Correct%20Learning%20(PAC).

https://scikit-

learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

Google Colab link for the code assignment:

https://colab.research.google.com/drive/1CgJFBf0r9Hc9js783LEYCEZ8dj0iTfpW?u sp=sharing