Assignment 4
(Decision Tree and Ensemble Learning)

Ahmed Shehata Mahmoud Sarah Hossam Elmowafy

Part 1: Numerical Questions:

1- **Gini index**

① Gini Split for weather

weather : 10

Sunny	Cloudy	Rainy
3 points	3 points	4 points
yes:2 no:1	yes:2 no:3	yes:1 no:3

$gini = 1 - \sum p(i)^2$

$= 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = .44$ $\quad = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = .44$ $\quad = 1 - (\frac{1}{4})^2 - (\frac{3}{4})^2 = .375$

- gini split $= 0.44 * \frac{3}{10} + 0.44 * \frac{3}{10} + 0.375 * \frac{4}{10} = 0.417$

② Gini Split for temperature

- hot 4 (2 yes, 2 no)

$gini = 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

- mild 5 (3 yes, 2 no)

$gini = 1 - (\frac{3}{5})^2 - (\frac{2}{5})^2 = 0.48$

- Cool = 1 (0 yes, 1 no)

$gini = 1 - 0 - 1 = 0$

- gini split $= 0.5 \times \frac{4}{10} + 0.48 \times \frac{5}{10} + 0 = 0.44$

③ Gini Split for humidity

- high 7 (3 yes, 4 no)

$gini = 1 - (\frac{3}{7})^2 - (\frac{4}{7})^2 = 0.487$

Normal 3(2 yes, 1 no)

$gini = 1 - (\frac{2}{3})^2 - (\frac{1}{3})^2 = 0.44$

$gini\ split = 0.489 \times \frac{7}{10} + 0.44 \times \frac{3}{10} = 0.476$

④ Split for wind

- weak 4(3 yes, 1 no)

$gini = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = 0.375$

- strong = 6(2 yes, 4 no)

$gini = 1 - (\frac{2}{6})^2 - (\frac{4}{6})^2 = 0.44$

- $gini\ split = 0.375 \times \frac{4}{10} + 0.44 \times \frac{6}{10} = 0.417$

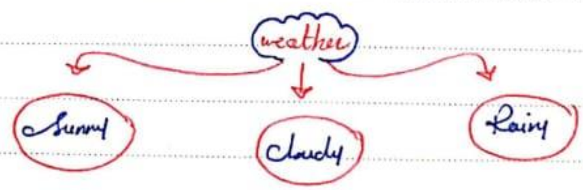
Gini Indecis

① - weather : 0.417 ← This will be the root node

② - temprature : 0.44

③ - Humedity : 0.476

④ - wind : 0.417



→ for sunny

f1	f2	f3	f4	label
sunny	hot	high	weak	yes
sunny	mild	normal	strong	yes
sunny	hot	high	weak	no

→ Gini split for f_2 Temprature (3)

- hot 2(1 yes, 1 no)

  gini $= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

- mild = 1(1 yes, 0 no)

  $= 1 - 0 - 1 = 0$

- gini split $= 0.5 \times \frac{2}{5} + 0 = 0.33$

→ Gini Split for Humidity (3)

- high 2(1 yes, 1 no)

  gini $= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

- normal = 1(1 yes, 0 no)

  gini $= 1 - 0 - 1 = 0$

- gini split $= 0.5 \times \frac{2}{3} + 0 = 0.33$

→ Gini Split for wind (3)

- weak 1(1 yes, 0 no)
- strong 2(1 yes, 1 no)

gini (weak) $= 1 - 0 - 1 = 0$

gini (strong) $= 1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

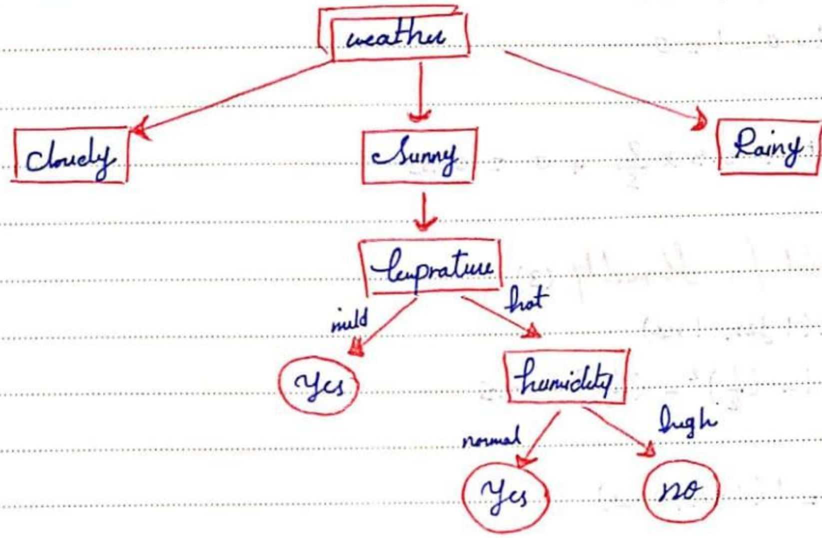
gini split $= 0 . 0.5 \times \frac{2}{3} = 0.33$

Gini Indicies

① Temprature : 0.33 ← The left most as all are equal.

② Humidity : 0.33

③ wind : 0.33



for cloudy

f_1	f_2	f_3	f_4	label
cloudy	hot	high	weak	no
cloudy	mild	high	strong	yes
cloudy	hot	normal	weak	yes

Gini split for Temprature ③

- hot 2(1y, 1n)

  $gini = 1-(\frac{1}{2})^2-(\frac{1}{2})^2 = 0.5$

- mild 1(1y, 0n)

  $gini = 1-0-1 = 0$

- gini split $= 0.5 \times \frac{2}{3} + 0 = 0.33$

Gini split for humidity

⊖ high 2 (1y, 1n)

⊖ normal 1 (1y, 0n)

gini split = 0.33

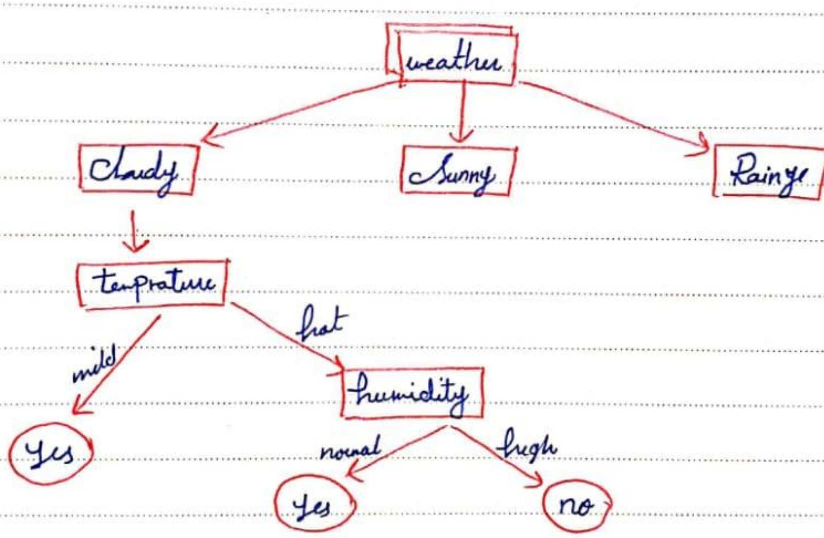
gini split for wind 0.33

Gini Indecis

temprature : 0.33

humidity : 0.33

wind : 0.33



for Rainy

f1	f2	f3	f4	label
rainy	mild	high	strong	no
rainy	cool	normal	strong	no
rainy	mild	high	weak	yes
rainy	mild	high	strong	no

Gini split for temp

- mild 3 (1 yes, 2 no)
- Cool 1 (0 yes, 1 no)
- gini $= 1 - (\frac{1}{3})^3 - (\frac{2}{3})^2 = 0.44$
- gini $= 0.44 \times \frac{3}{4} + 0 = 0.33$

Gini Split for humidity

- high 3 (1 yes, 2 no)
- normal 1 (0 yes, 1 no)
- gini $= 1 - (\frac{1}{3})^2 - (\frac{2}{3})^2 = \frac{4}{9} = 0.44$
- gini $= 1 - 0 - 1 = 0$

Gini Split for wind

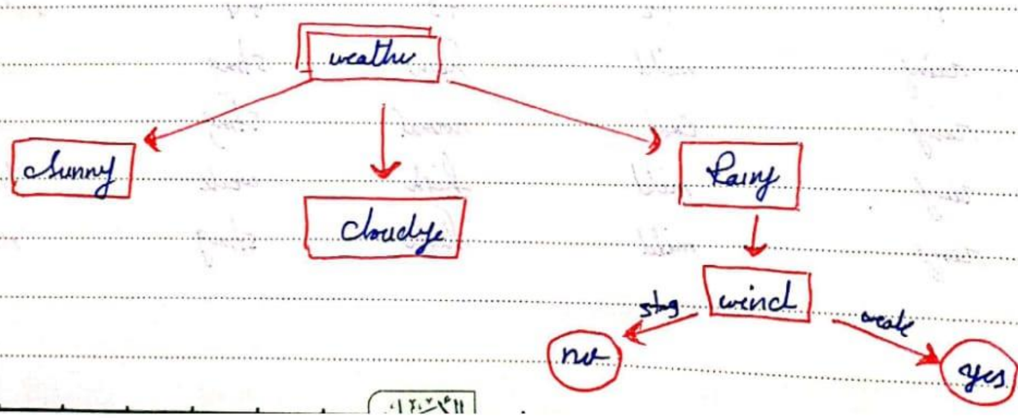
- weak 1 (0 yes, 1 no)
- strong 3 (0 yes, 3 no)
- gini $= 1 - 0 - 1 = 0$
- gini $= 1 - (\frac{0}{3})^2 - (\frac{3}{3})^2 = 0$

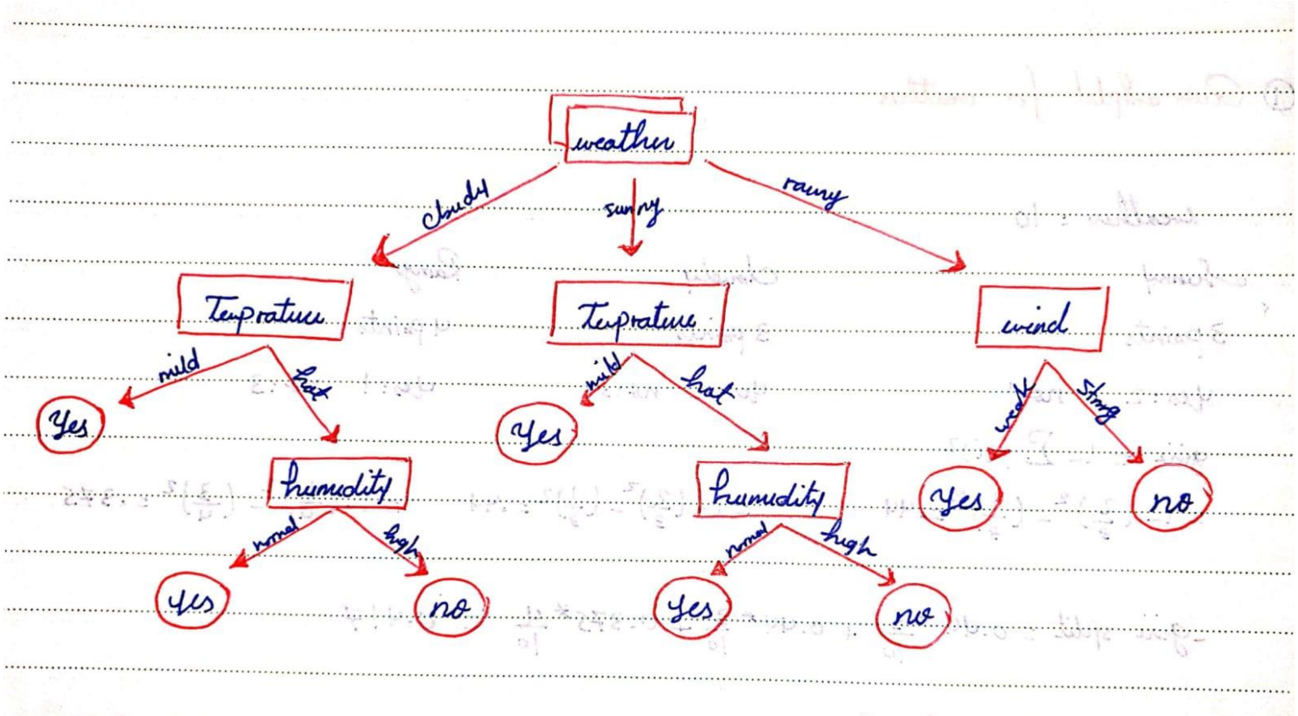
Gini Indices

Temprature $= 0.33$

Humidity $= 0.33$

wind $= 0$ ← we will chose the least gini value





2- Information Gain

Information Gain

① → $Entropy(S) = -P_{yes}\log_2 P_{yes} - P_{no}\log_2 P_{no}$

$= -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10} = 1$

② find which feature has the maximal information gain.

① → $Gain(S, weather) = 1 - \frac{|S_{cloudy}|}{10} Entropy(S_{cloudy}) -$

$\frac{|S_{sunny}|}{10} Entropy(S_{sunny}) - \frac{|S_{rainy}|}{10} Entropy(S_{rainy})$

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

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

$1 + (-.275 - .275 - 0.324) = 1 - (-.874) = 0.126$

② → $Gain(S, Temprature) = 1 - \frac{|S_{hot}|}{10} Entropy(S_{hot}) -$

$\frac{|S_{mild}|}{10} Entropy(S_{mild}) - \frac{|S_{cool}|}{10} Entropy(S_{cool})$

$= 1 - \frac{4}{10}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{5}{10}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{3}{5}\right)$

$- \frac{1}{10}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) = 1 - .4 - 0.485 - 0 = .115$

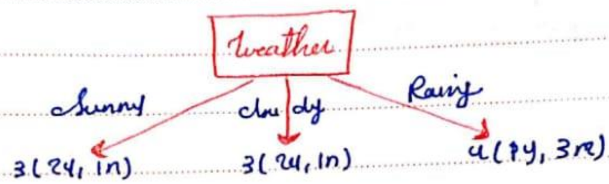
③ → $Gain(S, Humidity) = 1 - \frac{7}{10}\left(-\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7}\right)$

$- \frac{3}{10}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) = 1 - .689 - .275 = .036$

$$Gain(S, wind) = 1 - \frac{6}{10}\left(\frac{-4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right)$$

$$- \frac{4}{10}\left(\frac{-3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right) = 1 - .55 - 32 = .126$$

⇒ Now, The weather and wind gain have the same value so, we will go with the left most which is weather. so, our root node is weather.



→ let's start with Sunny

→ Entropy(Sunny) $= P_{yes}(\log_2 P_{yes}) - P_{no}(\log_2 P_{no})$

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

1. $Gain(S_{sunny}, temp) = 0.918 - \frac{|S_{hot}|}{3} Entropy(S_{hot}) - \frac{|S_{mild}|}{3} Entropy(S_{mild})$

$$= 0.918 - \frac{2}{3}\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) - \frac{1}{3}\left(\frac{-1}{1}\log 1\right)$$

$$= 0.918 - 0.667 - 0 = \boxed{0.251}$$

2. $Gain(S_{sunny}, Humidity) = 0.918 - \frac{|S_{high}|}{3} Entropy(S_{high})$

$$- \frac{|S_{normal}|}{3} Entropy(S_{normal})$$

$$= 0.918 - \frac{2}{3}\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) - \frac{1}{3}\left(\frac{-1}{1}\log 1\right)$$

$$= 0.918 - 0.667 - 0 = 0.251$$

3) Gain (S_{sunny}, wind) $= 0.918 - \frac{|S_{weak}|}{3}$ Entropy(S_{weak})

$- \frac{|S_{strong}|}{3}$ Entropy(S_{strong})

$= 0.918 - \frac{2}{3}\left(\frac{-1}{1}\log 1\right) - \frac{2}{3}\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right)$

$= 0.918 - 0.667 - 0 = 0.251$

Entropy(cloudy) $= -P_{yes}\log_2 yes - P_{no}\log_2 no$

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

Gain (S_{cloudy}, Temp) $= \frac{|S_{hot}|}{3}$ Entropy(S_{hot}) $- \frac{|S_{mid}|}{3}$ Entrop(S_{mid})

$= 0.918 - \frac{2}{3}\log\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) - \frac{1}{3}(-1\log 1)$

$= 0.918 - 0.667 - 0 = \boxed{0.251}$

Entropy(Rainy) $= -P_{yes}\log_2 yes - P_{no}\log_2 no$

$= -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 0.8112$ for later

Gain (S_{cloudy}, Humidity) $= 0.918 - \frac{|S_{high}|}{3}$ Entropy(S_{high})

$- \frac{|S_{normal}|}{3}$ Entropy(S_{normal})

6 $= 0.918 - \frac{2}{3}\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) - \frac{1}{3}(-1\log 1)$

$= 0.918 - 0.667 - 0 = 0.251$

Gain (S_{cloudy}, wind) $= 0.918 - \frac{|S_{weak}|}{3}$ Entropy(S_{weak})

7 $- \frac{|S_{strong}|}{3}$ Entropy(S_{strong})

$= 0.918 - \frac{2}{3}\left(\frac{-1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) - \frac{1}{3}(-1\log 1)$

8 $= 0.918 - 0.667 - 0 = 0.251$

Gain (S_{Rain}, Temp) $= 0.8112 - \frac{|S_{mild}|}{4}$ Entropy(S_{mild}) $- \frac{|S_{cool}|}{4}$ Entropy

(S_{cool}) $= 0.8112 - \frac{3}{4}\left(\frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) - \frac{1}{4}(-1\log 1)$

$= .8112 - 0.6887 - 0 = 0.122$

$$\text{Gain}(S_{Rainy}, \text{Humidity}) = 0.812 - \frac{|S_{high}|}{4}\,\text{Entropy}(S_{high}) - \frac{|S_{normal}|}{4}\,\text{Entropy}(S_{normal})$$

$$= 0.8122 - \frac{3}{4}\left(\frac{-2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) - \frac{1}{4}(-1\log 1)$$

$$= 0.8122 - 0.6887 - 0 = 0.1225$$

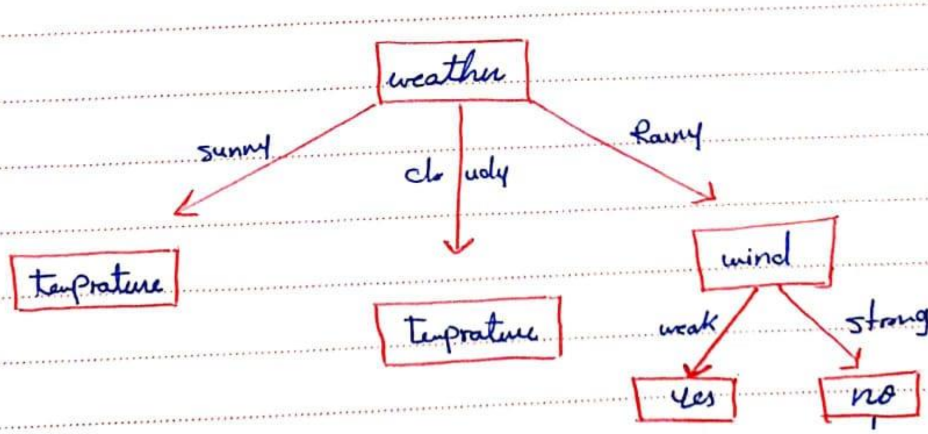
$$\text{Gain}(S_{Rainy}, \text{wind}) = 0.8112 - \frac{|S_{strong}|}{4}\,\text{Entropy}(S_{strong}) - \frac{|S_{weak}|}{4}\,\text{Entropy}(S_{weak})$$

$$= 0.8112 - \frac{3}{4}\left(\frac{-3}{3}\log\frac{3}{3}\right) - \frac{1}{4}(-1\log 1)$$

$$= 0.8112 - 0 - 0 = \boxed{0.8112}$$

→ for sunny and cloudy all splits have the same gain so we will go with the left most temprature

→ for Rainy wind has the higher gain



→ for Sunny

$$\text{Entropy}(S_{hot}) = -P_{yes}\log(P_{yes}) - P_{no}(\log P_{no})$$
$$= -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

$$\text{Gain}(S_{hot}, \text{Humidity}) = 1 - \frac{|S_{high}|}{2}\text{Entropy}(S_{high})$$
$$= 1 - \frac{2}{2}\left(-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) = 0$$

$$\text{Gain}(S_{hot}, \text{wind}) = 1 - \frac{|S_{weak}|}{2}\text{Entropy}(S_{weak}) - \frac{|S_{strong}|}{2}\text{Entropy}(S_{strong})$$
$$= 1 - \frac{1}{2}(-1\log 1) - \frac{1}{2}(-\log 1) = \boxed{1}$$

→ By logic, mild is a pure node here, it will be yes

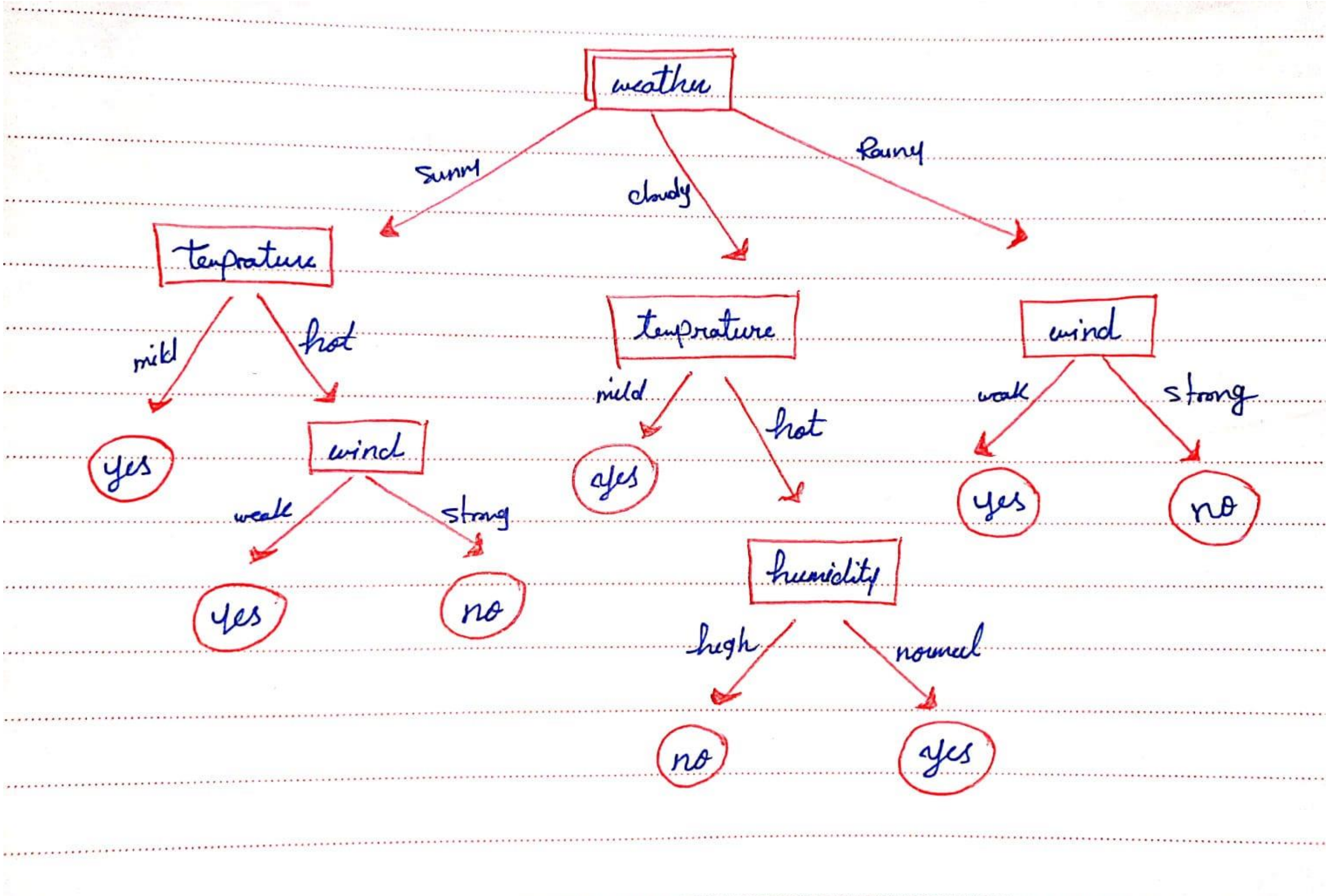
→ for Cloudy

$$\text{Entropy}(S_{hot}) = -P_{yes}\log_{yes} - P_{no}\log_{no}$$
$$= -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$$

$$\text{Gain}(S_{hot}, \text{Humidity}) = 1 - \frac{|S_{high}|}{2}\text{Entropy}(S_{high}) - \frac{|S_{normal}|}{2}\text{Entropy}(S_{normal})$$
$$= 1 - \frac{1}{2}(-1\log 1) - \frac{1}{2}\log 1 = \boxed{1}$$

$$\text{Gain}(S_{hot}, \text{wind}) = 1 - \frac{|S_{weak}|}{2}\text{Entropy}(S_{weak})$$
$$= 1 - \frac{2}{2}\left(-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}\right) = 0$$

→ for mild, pure node always yes

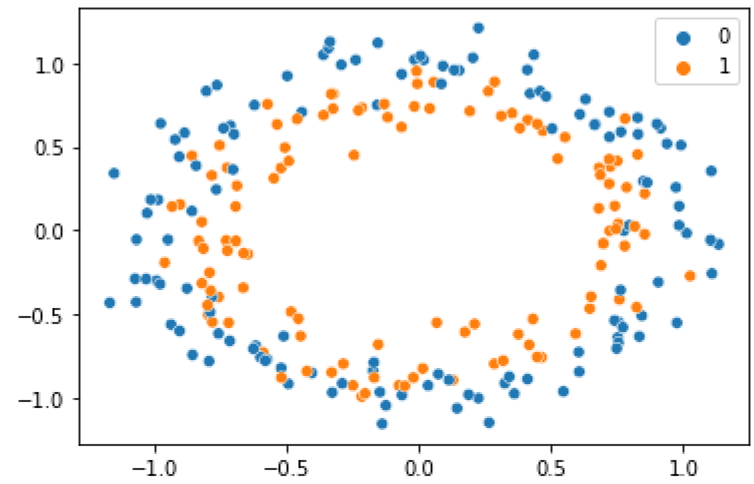

weather
sunny
cloudy
Rainy
temprature
mild
hot
yes
wind
weak
strong
yes
no
temprature
mild
hot
yes
humidity
high
normal
no
yes
wind
weak
strong
yes
no

Part 2: Programming Questions

- **First:** Create Circle and Classification datasets using sklearn make_circles and make_classification methods.

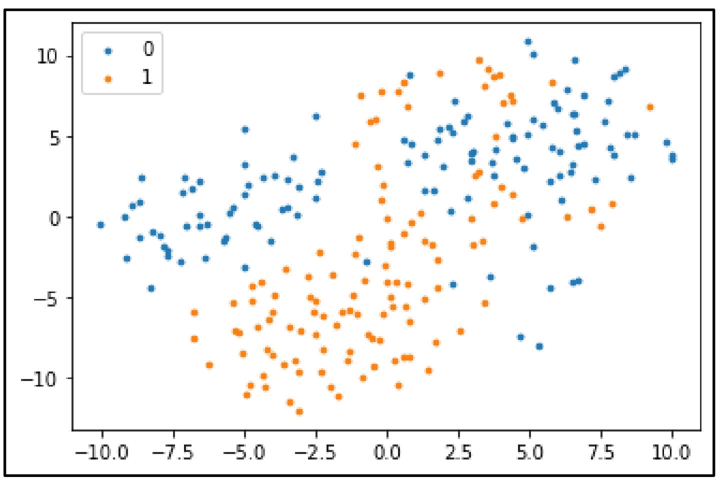
Circle Dataset (300, 2)

	0	1
0	-0.492932	0.414743
1	-0.131501	0.756043
2	0.826503	0.674707
3	-0.693855	0.142170
4	-0.788881	-0.358277



Classification Dataset (300,20)

	0	1	2	3	4	5	6	7	8	9
0	-1.325981	0.095412	-1.037029	-0.057757	-1.448467	1.226664	0.666355	-0.904076	0.596930	-0.726712
1	1.702056	-0.335052	1.078571	-1.342867	1.297082	-0.167253	-0.604033	2.628537	0.230704	0.415251
2	-0.492697	-0.182905	0.215240	-0.134822	-0.857703	0.045166	1.859346	-2.777665	-1.436356	-0.584094
3	0.870360	-2.263660	1.177830	-1.385278	-0.413804	-0.927363	0.888339	3.462582	1.556596	0.102178
4	-1.556177	-1.230546	-1.143981	-2.163098	0.085843	-1.406460	-0.174823	-1.007583	0.736853	-0.396123



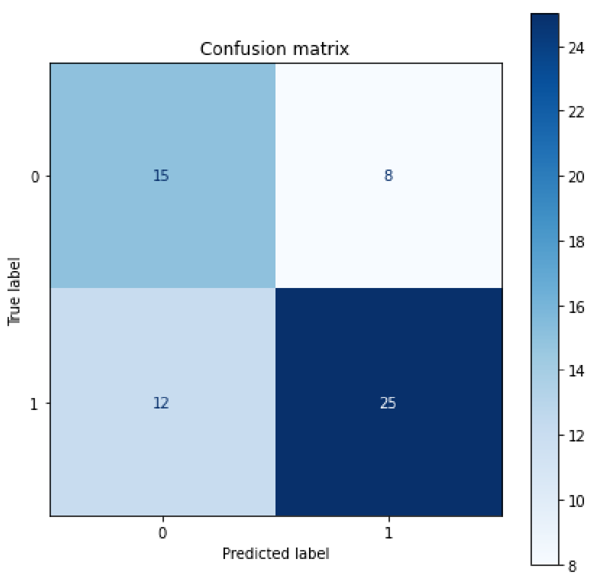
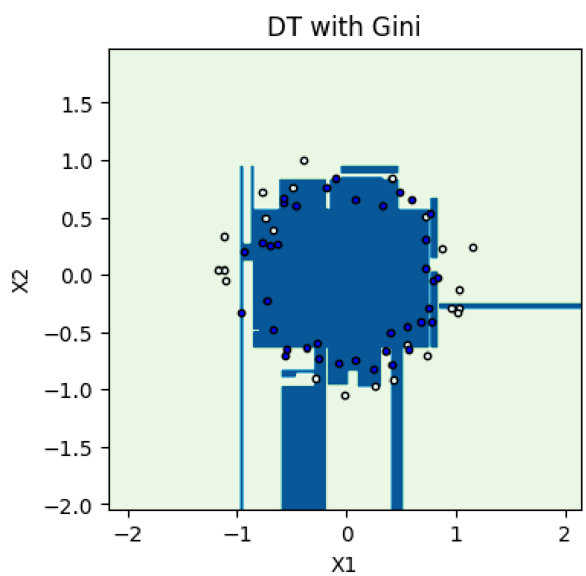
- **Decision Tree**
  - **Q4:**

```python
from sklearn.tree import DecisionTreeClassifier

def build_dt(crit, title):

    dt = DecisionTreeClassifier(random_state=0, criterion= crit)
    dt.fit(X_train_ci, y_train_ci)

    dt_pred = dt.predict(X_test_ci)

    print_accuracy(dt, y_test_ci, dt_pred, X_test_ci)
    plot_decision_boundary(X_test_ci, y_test_ci, dt, title)

    return dt, dt_pred
```

Decision Tree Using Gini (Accuracy: 0.67)

1. Decision Tree with gini

```python
dt1, y_pred_dt1 = build_dt("gini", "DT with Gini")
```

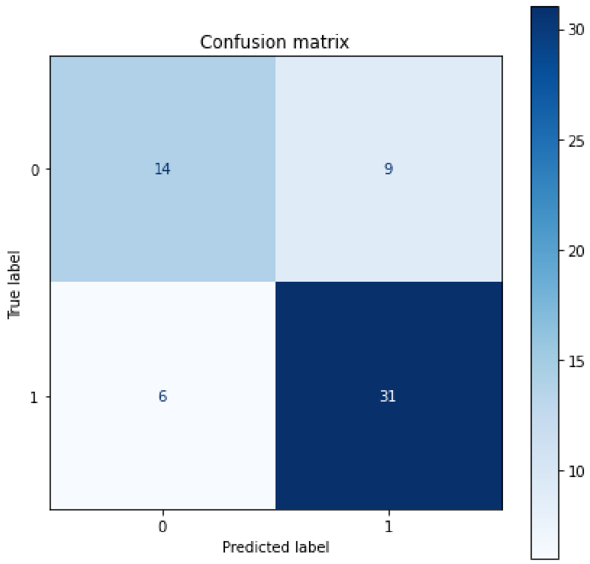
Confusion Matrix	**Decision Boundary**
	

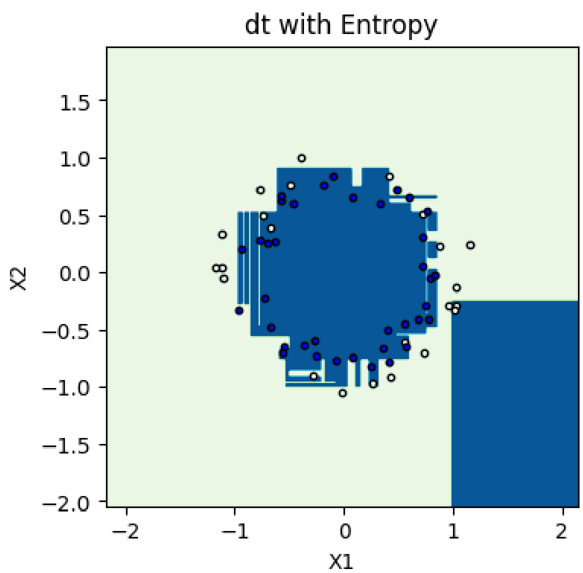
Decision Tree Using Entropy (Accuracy: 0.75)

Decision Tree with entropy

```
dt2, y_pred_dt2 = build_dt("entropy", "dt with Entropy")
```

Confusion Matrix	Decision Boundary



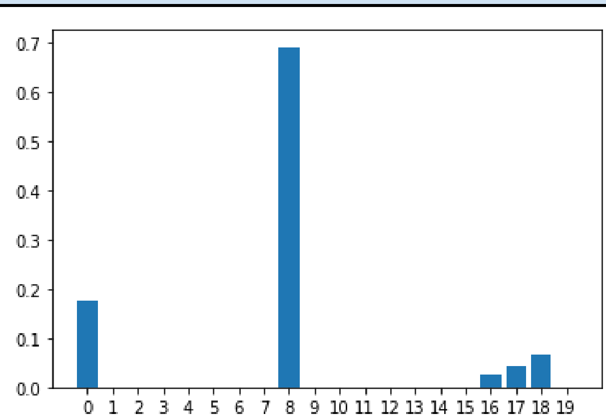


- As we see from the two decision tree models based on gini and entropy, the second model based on entropy get higher accuracy than the first one as the entropy add more complexity to the model due to logarithm function, so it fit the data better than gini.
- And due to logarithm function Entropy may be a little slower to compute

- **Q5**
  - Her, I get the importance features after applying decision tree to classification dataset, we got 8 features from 19

Get importance features	Importance features

```
# get importance
importance = dt_cl.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: {}, Score: {:.5f}'.format(i,v))

# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.xticks([x for x in range(len(importance))])
plt.show()
```



- Then, I create a list of 7 lists, the first list has 1 feature, second has 2 features, third has 3 features, up to list number 7 has the 7 features.
- Fit decision tree to list of 7 a lists which has the most importance feature as I mentioned using cross validation of 4
- We got 4 accuracy at each list of lists
-

```
valid_acc = []
test_acc =[]

dt_model = DecisionTreeClassifier(random_state=0, criterion= 'entropy')

for i in lst:
    cv_acc = cross_validate(dt_model, X_train_cl[:,i], y_train_cl, cv = 4)
#     print(cv_acc)
    valid_acc.append(cv_acc['test_score']*100)

    dt_model.fit(X_train_cl[:, i], y_train_cl)
    y_preds = dt_model.predict(X_test_cl.iloc[:, i])
    test_accuracy =  accuracy_score(y_test_cl, y_preds)
    test_acc.append(test_accuracy* 100)

cv_acc = list(map(list, zip(*valid_acc)))
```

- Then, I plot the Validation Accuracy (y-axis) vs Top K Important Feature (x-axis) curve, and test accuracy

<table>
<tr><th>Validation</th><th>Test</th></tr>
<tr><td>

Plot Fold Accuracy vs Top K features

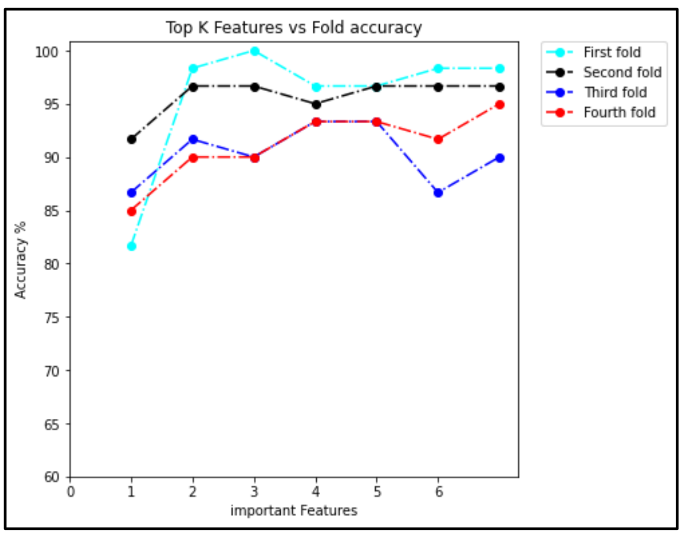
```
values = [x for x in range(1,8)]
plot_importance_vs_accuracys(values[:8], cv_acc,
                             "Top K Features vs Fold accuracy")
```

</td><td>

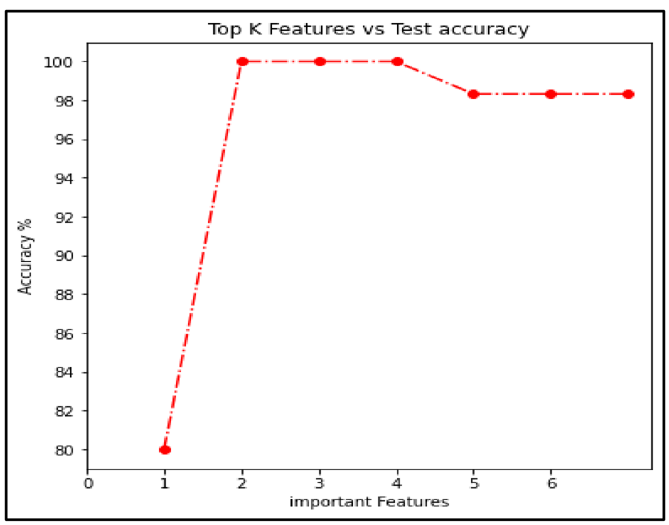
Plot Test Accuracy vs Top K features

```
plot_importance_vs_accuracys(values[:8], test_acc,
                             "Top K Features vs Test accuracy")
```

</td></tr>
<tr><td>



</td><td>

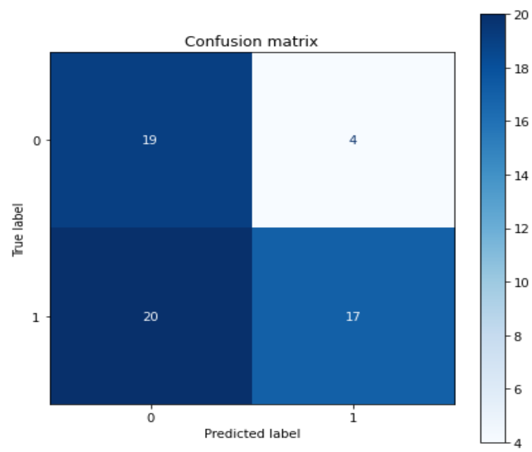
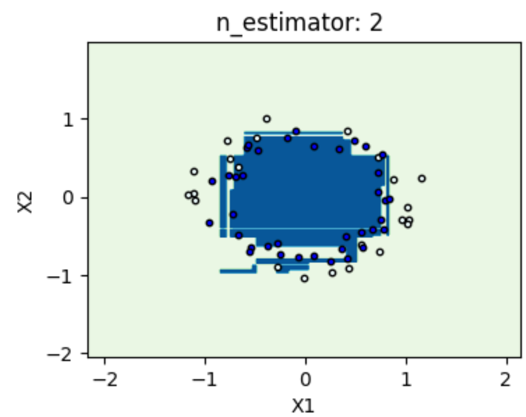
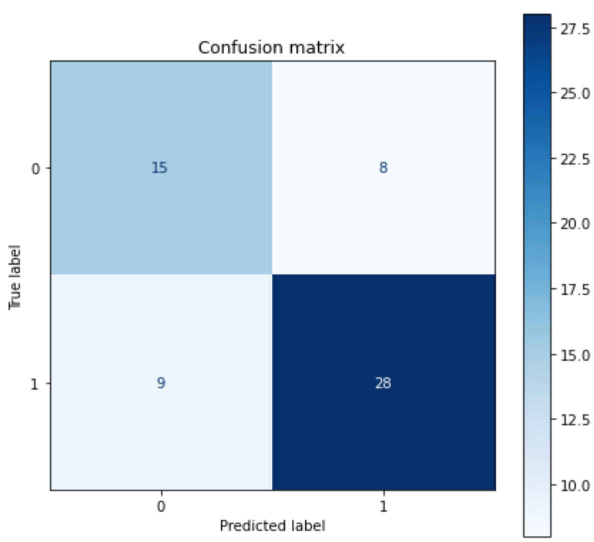
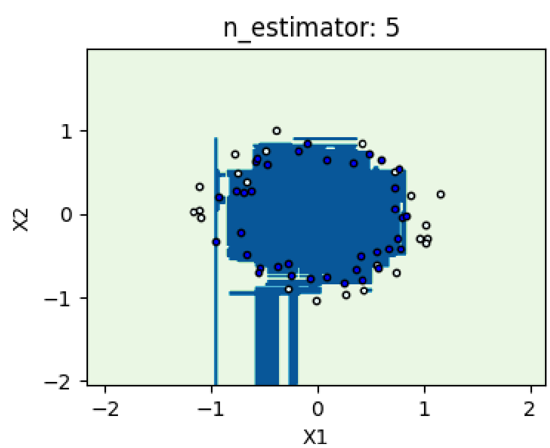


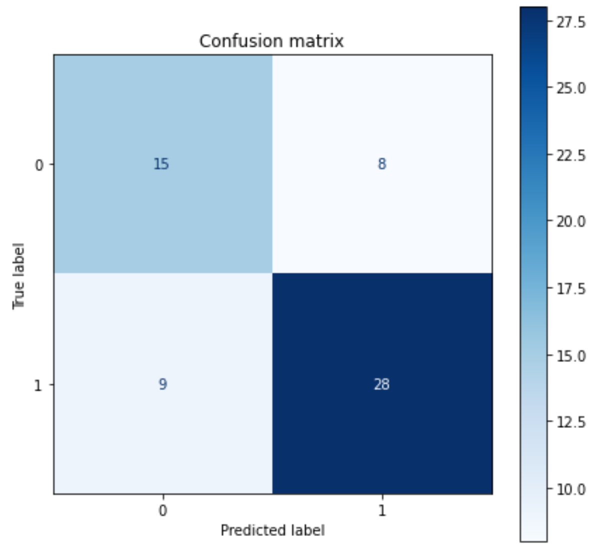
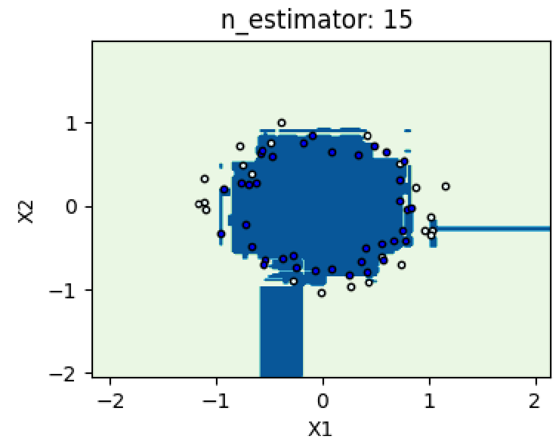
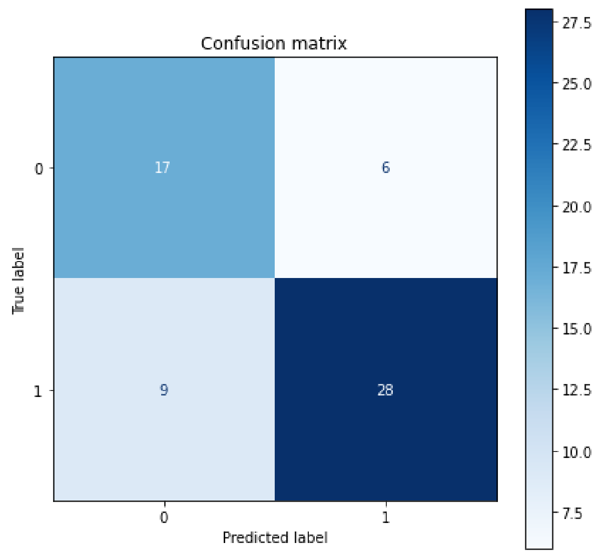
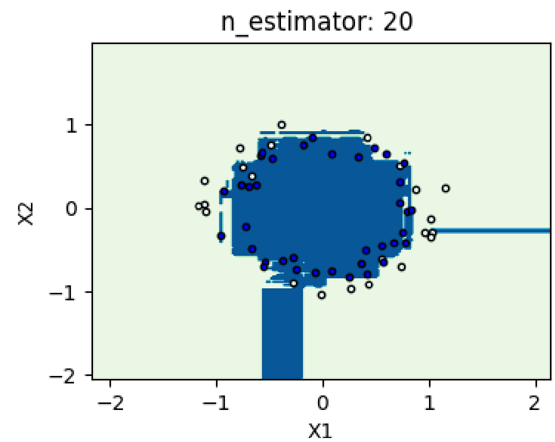
</td></tr>
</table>

Bagging

- **Q6**

```
from sklearn.ensemble import BaggingClassifier
# base_estimator = DecisionTreeClassifier

c = 1
n_estimators = [2, 5, 15, 20]
for i in n_estimators:
    bc_model = BaggingClassifier(n_estimators = i, random_state=0)
    bc_model.fit(X_train_ci, y_train_ci)

    bc_pred = bc_model.predict(X_test_ci)

    print("\nModel: {}".format(c))
    title_bc = "n_estimator: {} ".format(i)
    plot_decision_boundary(X_test_ci, y_test_ci, bc_model, title_bc)

    print_accuracy(bc_model, y_test_ci, bc_pred, X_test_ci)
    print("------------------------------------------------------------")

    c += 1
```

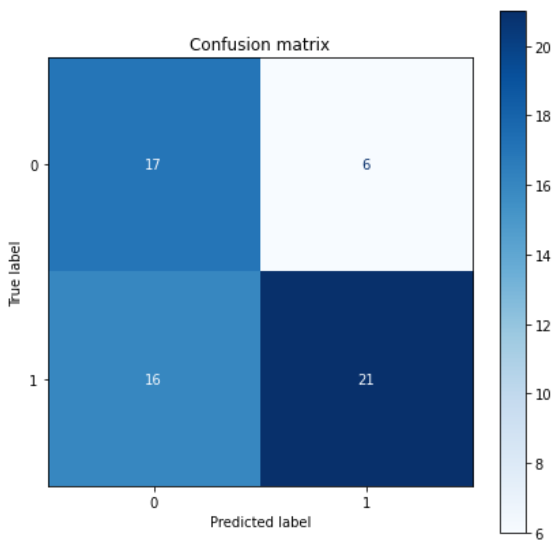
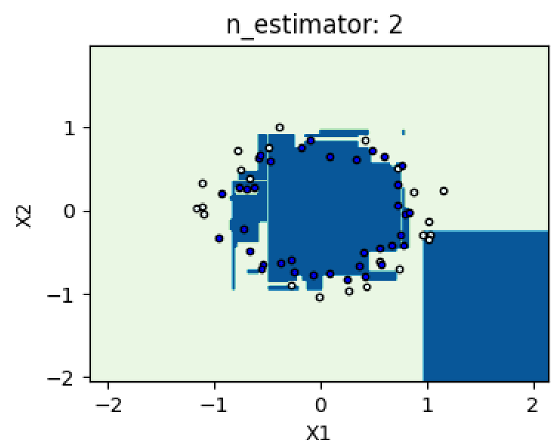
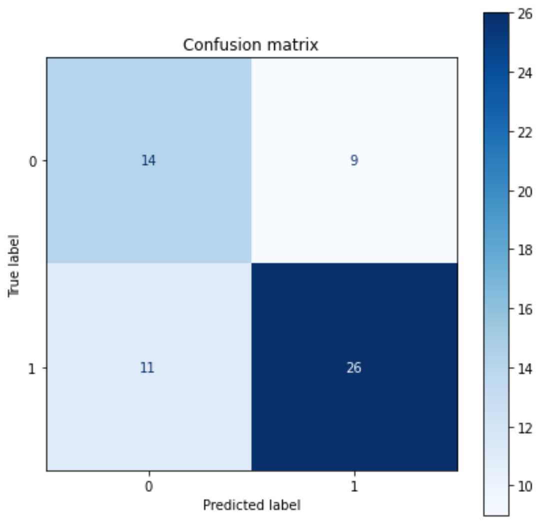
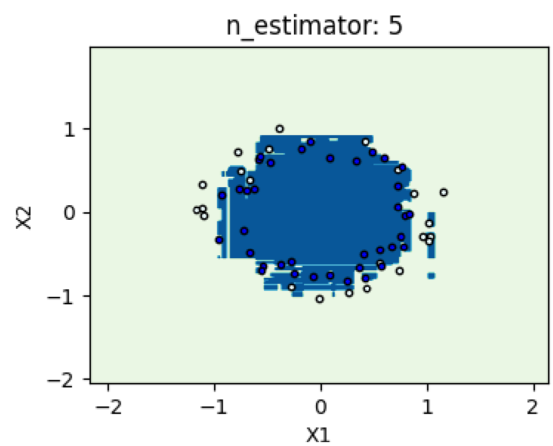
Estimators	Confusion Matrix	Decision Boundary
2 **Accuracy: 0.60**		
5 **Accuracy: 0.75**		

15 **Accuracy: 0.72**		
20 **Accuracy: 0.75**		

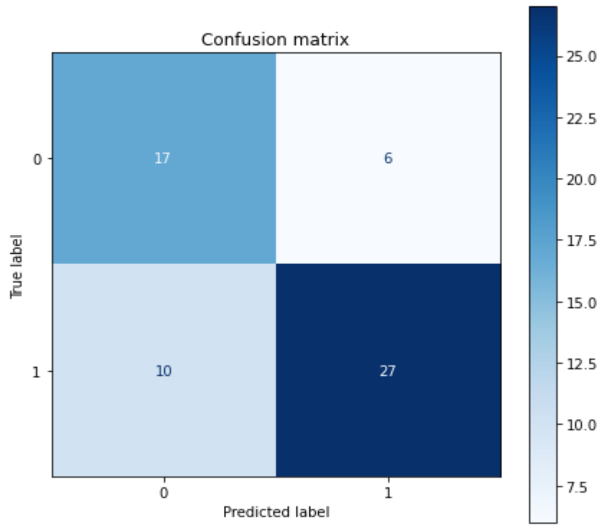
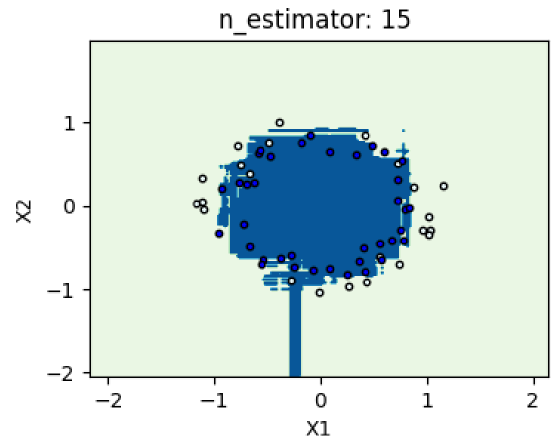
- **Q7**
  - Bagging refers to training the same model multiple times on different data sets(which are obtained by random sampling with replacement from the training set, we have **bootstrapping**). All the models are combined at the end, which leads to higher stability and a **lower variance** compared to the individual models, which lead to **reduce overfitting**.

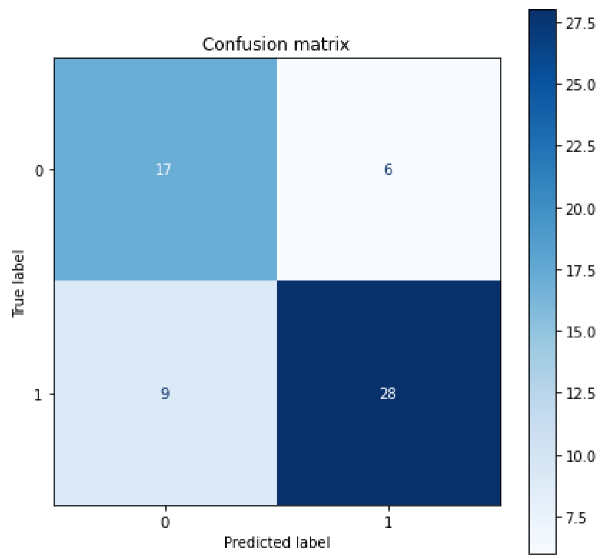
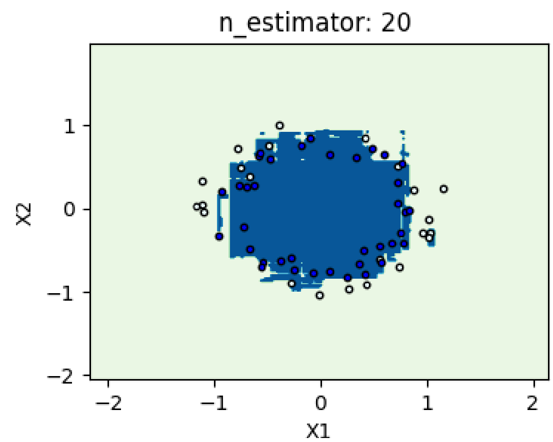
- **Random Forest**
  - **Q8**

```
from sklearn.ensemble import RandomForestClassifier as RF
n_estimators = [2, 5, 15, 20]
c = 1
for i in n_estimators:
    rf_model = RF(n_estimators = i)
    rf_model.fit(X_train_ci, y_train_ci)

    rf_pred = rf_model.predict(X_test_ci)

    print("\nModel: {}".format(c))
    title_rf = "n_estimator: {} ".format(i)
    plot_decision_boundary(X_test_ci, y_test_ci, rf_model, title_rf)
    print_accuracy(rf_model, y_test_ci, rf_pred, X_test_ci)

    print("-------------------------------------------------------")
    c += 1
```

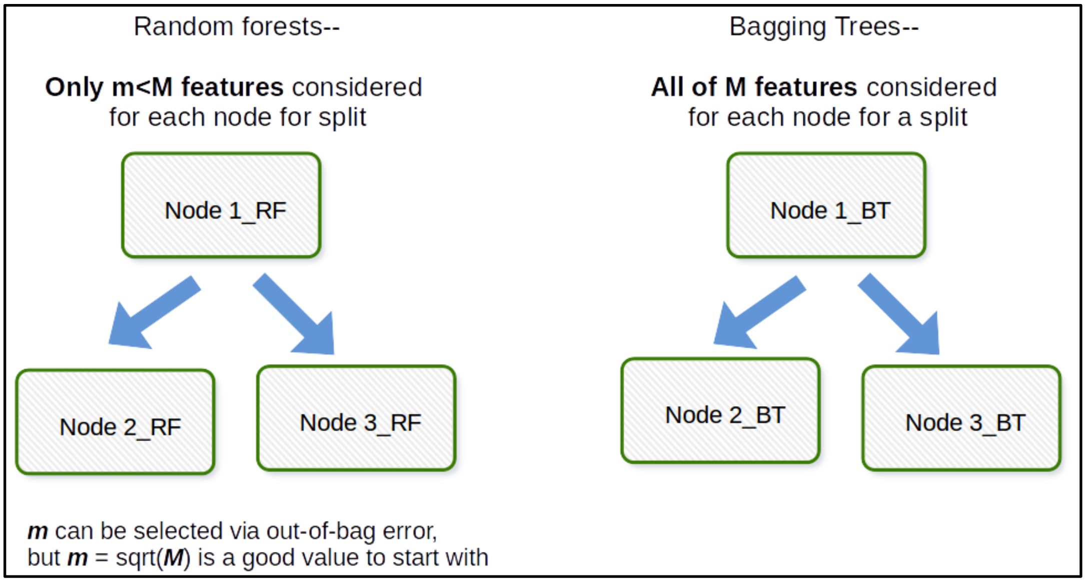
Estimators	Confusion Matrix	Decision Boundary
2 **Accuracy: 0.63**		
5 **Accuracy: 0.67**		

<table>
<tr>
<td>15

Accuracy: 0.73</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>20

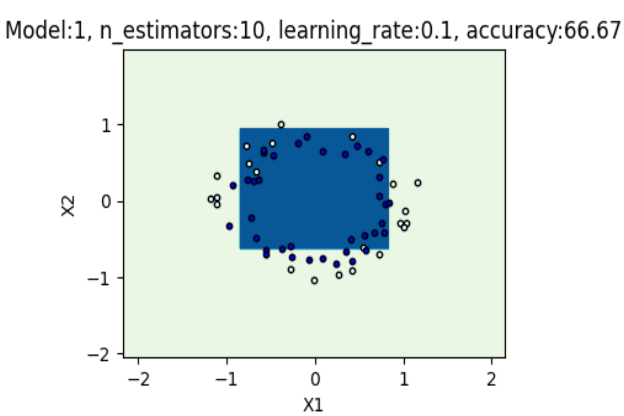
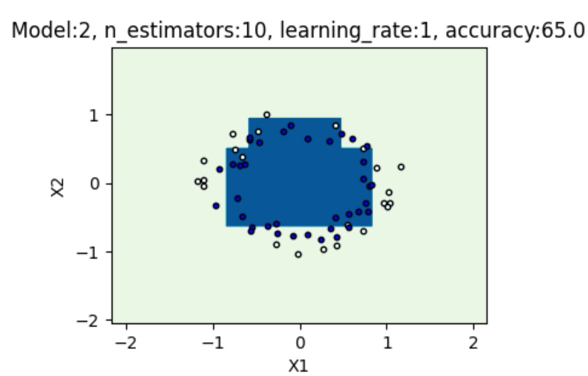
Accuracy: 0.75</td>
<td>
</td>
<td>
</td>
</tr>
</table>

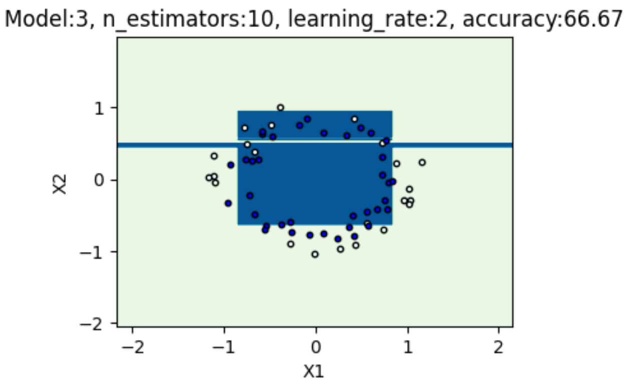
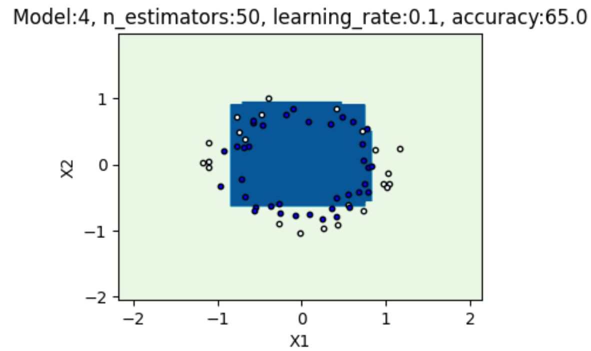
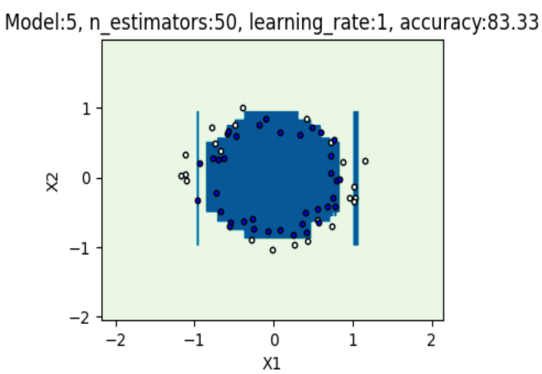
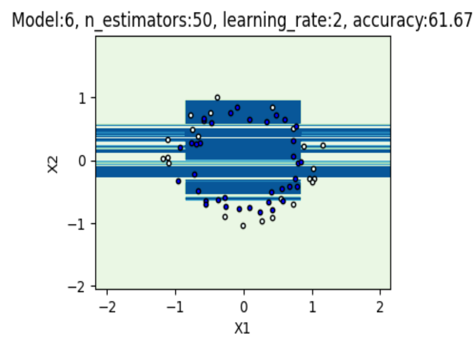
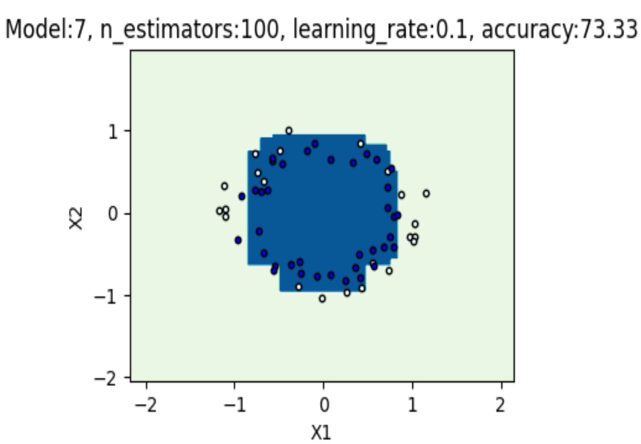
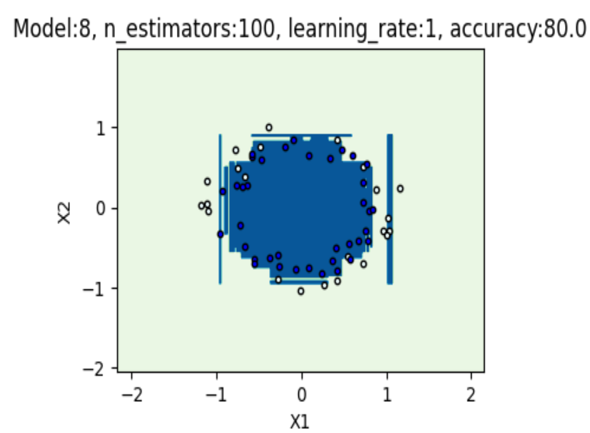
- **Q9**
    - From our 4 models of random forest based on different number of estimators, we see that the accuracy increases as the number of estimators increased, as we see the best model got accuracy with number of estimators equal 20
    - The difference between bagging and random forest is that
        - In Random forests, only a subset of features is selected at random out of the total and the best split feature from the subset is used to split each node in a tree,
        - unlike in bagging, where all features are considered for splitting a node.

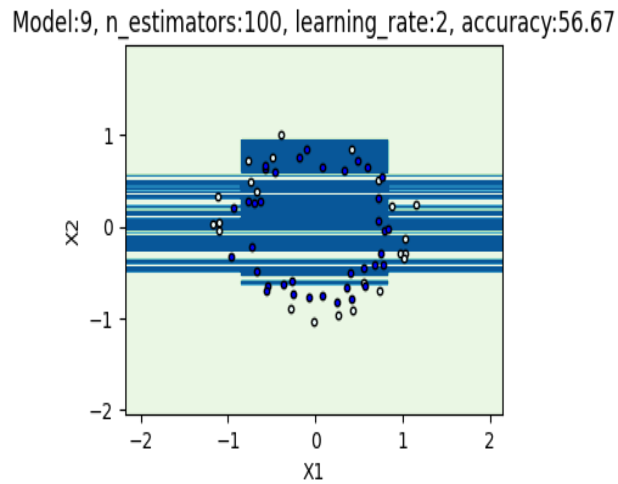
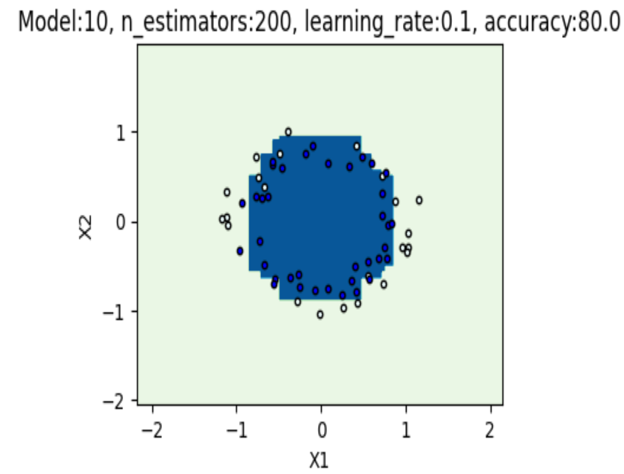
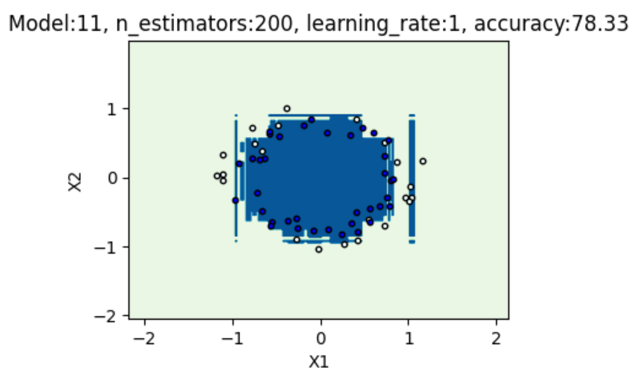
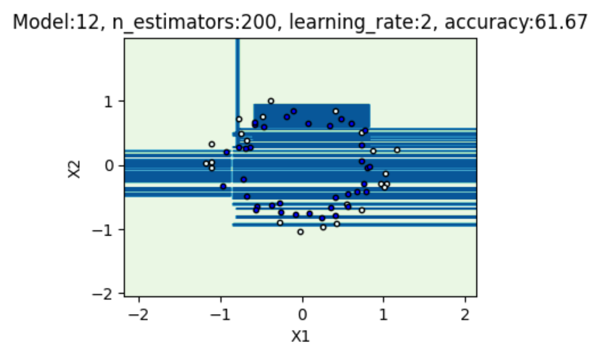


- **Boosting**
    - **Q10**

```
from sklearn.ensemble import AdaBoostClassifier

n_estimators = [10, 50, 100, 200]
lr = [0.1, 1, 2]

count = 1
for i in n_estimators:
    for j in lr:
        ada_model = AdaBoostClassifier(n_estimators= i, learning_rate= j)

        ada_model.fit(X_train_ci, y_train_ci)

        ada_pred = ada_model.predict(X_test_ci)
        acc = round(accuracy_score(y_test_ci, ada_pred) * 100, 2)

#         print_accuracy(ada_model, y_test_ci, ada_pred, X_test_ci)
        title_ada = "Model:{}, n_estimators:{}, learning_rate:{}, accuracy:{}".format(count, i, j, acc)
        plot_decision_boundary(X_test_ci, y_test_ci, ada_model, title_ada)

        count += 1
```

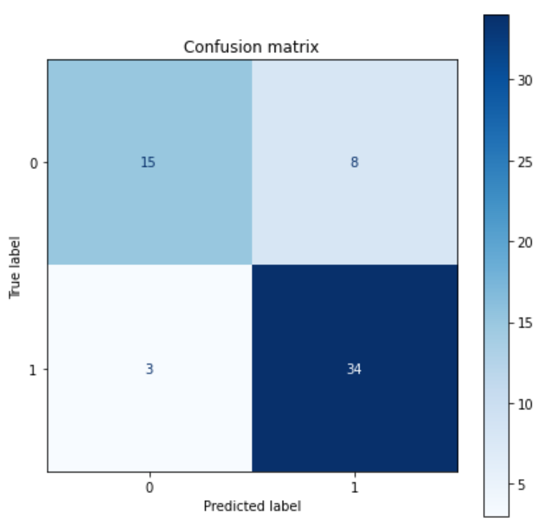
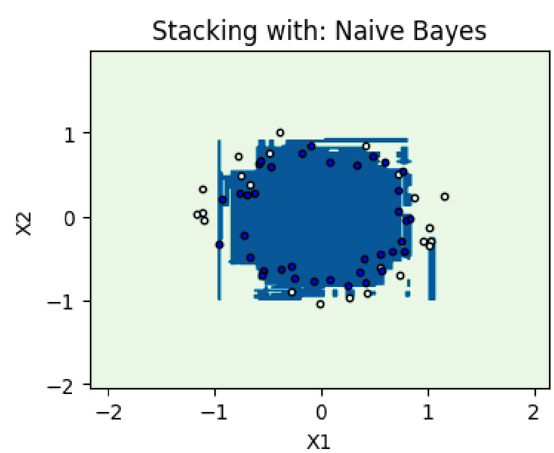
Model 1 accuracy: 66.67	**Model 2 accuracy: 65.0**
	

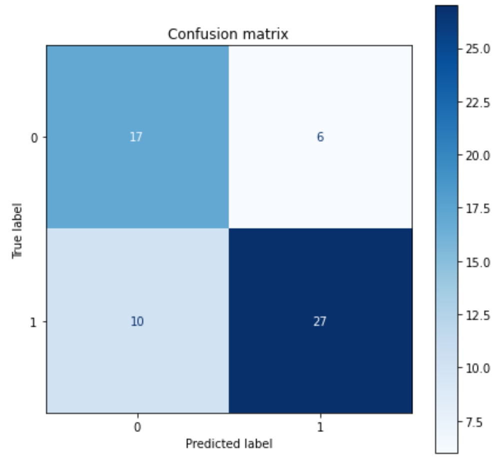
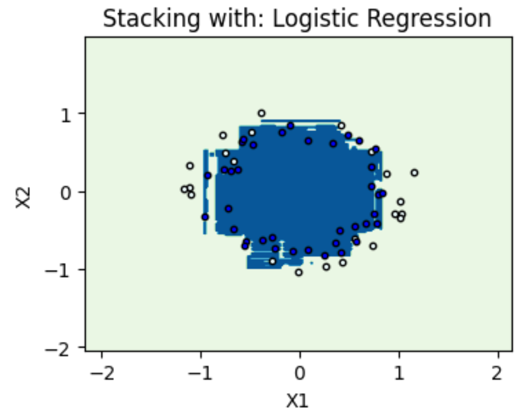
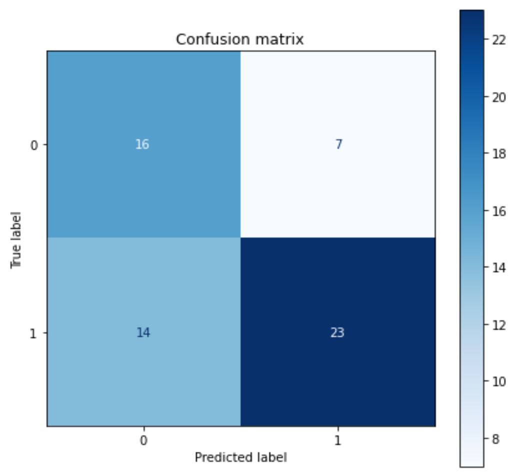
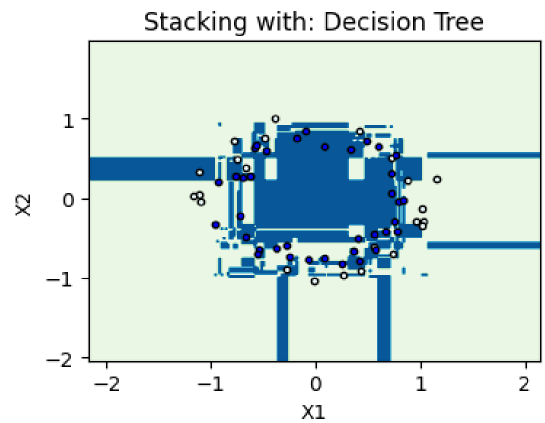
Model 3 accuracy: 66.67	Model 4 accuracy: 65.0
	
Model 5 accuracy: 83.33	**Model 6 accuracy: 61.67**
	
Model 7 accuracy: 73.33	**Model 8 accuracy: 80.0**
	
Model 9 accuracy: 56.67	**Model 10 accuracy: 80.0**

	
Model 11 accuracy: 78.33	**Model 12 accuracy: 61.67**
	

- **Stacking**
  - **Q11**

```
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB

estimators = [
     ('DT', DecisionTreeClassifier(random_state=0, criterion= 'entropy')),
     ('BC', BaggingClassifier(n_estimators = 5, random_state=0)),
     ('RF', RF(n_estimators = 2)),
     ('ADA', AdaBoostClassifier(n_estimators= 50, learning_rate= 1))]

aggregators = [
    DecisionTreeClassifier(random_state=0, criterion= 'entropy'),
    LogisticRegression(),
    GaussianNB()]

titles = ["Decision Tree", "Logistic Regression", "Naive Bayes"]

for i, j in enumerate(aggregators):
    stack_model = StackingClassifier(estimators= estimators, final_estimator=j)
    stack_model.fit(X_train_ci, y_train_ci)
    stack_pred = stack_model.predict(X_test_ci)

    acc = round(accuracy_score(y_test_ci, stack_pred) * 100, 3)
    print("\nStacking with: {}".format(titles[i]))
    stack_title = "\nStacking with: {}".format(titles[i])
    plot_decision_boundary(X_test_ci, y_test_ci, stack_model, stack_title)
    print_accuracy(stack_model, y_test_ci, stack_pred, X_test_ci)
    print("-----------------------------------------------------------")
```

Aggregators	Confusion Matrix	Decision Boundary
NB **Accuracy: 0.82**		

LR **Accuracy: 0.73**		
DT **Accuracy: 0.65**		

Conclusion

- In the numerical question the information gain and the gini index produced the same tree in the end.
- In this assignment, we have built some models based on different techniques as **decision tree**, **random forest**, **bagging**, **boosting**, and **stacking**
- We have learned the difference between bagging and random forest and how the models built with those techniques.
- Also, Boosting techniques based on sequential models, and we see that the model accuracy decreased as the learning rate and number of estimators increased, and best model got accuracy based on **50 number of estimators and 1 as Lr.**
- Additionally, in stacking, we see that the best aggregators here is **naive Bayes** which got **0.82 accuracy.**
- From all models created, we conclude that, the best models are the model that based on importance features as decision tree with cross validation, as it reach 99% sometimes.
- We also conclude that **cross validation** is very useful with little data.