ELG 5255[EG] Applied Machine Learning Summer

Assignment -Four (Decision tree and ensemble method)

(Group 34)

Ahmed Hallaba   Omar Sorour

Kamel El-Sehly

Part 1 numerical questions:

Q1: Please build a decision tree by using Gini Index (i.e., $Gini = 1 - \sum_{i=1}^{N_c} p_i^2$, where N_c is the number classes)

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

We first calculated the Gini index for each feature to determine the root

For Weather (F1):

Rainy (1Y,3N), Cloudy (2Y,1N), Sunny(2Y,1N)

Gini (Rainy) = 1 – $(1/4)^2$ - $(3/4)^2$= 0.375

Gini (Cloudy) = 1 – $(2/3)^2$ - $(1/3)^2$=0.444

Gini (Sunny) = 1 – $(2/3)^2$ - $(1/3)^2$=0.444

Weighted Gini (Weather) = 0.375(4/10) + 0.444(3/10) + 0.444(3/10) = 0.4167

For Temperature (F2):

Hot (2Y,2N), mild (3Y,2N), Cold(0Y,1N)

Gini (Hot) = 1 – $(2/4)^2$ - $(2/4)^2$= 0.5

Gini (Mild) = 1 – $(3/5)^2$ - $(2/5)^2$=0.48

Gini (Cold) = 1 – $(0/1)^2$ - $(1/1)^2$=0

Weighted Gini (Weather) = 0.5(4/10) + 0.48(5/10) + 0 (1/10) = 0.44

For Humidity (F3):

High (3Y,4N), Normal (2Y,1N)

Gini (High) = 1 – $(3/7)^2$ - $(4/7)^2$= 0.489

Gini (Normal) = 1 – $(2/3)^2$ - $(1/3)^2$=0.444

Weighted Gini (Humidity) = 0.489(7/10) + 0.4444(3/10) = 0.4761

For Wind (F4):

Weak (3Y,1N), Strong (2Y,4N)

Gini (Weak) = 1 – $(3/4)^2$ - $(1/4)^2$= 0.375

Gini (Normal) = 1 – $(2/6)^2$ - $(4/6)^2$=0.444

Weighted Gini (Humidity) = 0.375 (4/10) + 0.444(6/10) = 0.4167

Based on the previous we can see that the weather and the wind had the lowest hance we can select any of them to be the root node.

I chose the weather to be the root node and the rainy to be the condition.

For the left branch when Weather = rainy the corresponding table would be:

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

1. F2(Temperature): Hot has (0 Y, 0 N)
   Mild has (1 Y, 2 N)
   Cool has (0 Y, 1 N)
   Gini Index for Hot = $1 – (0)^2 – (0)^2 = 1$
   Gini Index for Mild = $1 – (1/3)^2 – (2/3)^2 = 0.444$
   Gini Index for Cool = $1 – (0)^2 – (1)^2 = 0$

   So, now we calculate the weighted Gini Index for **Temperature**

   = 1(0) + 0.444(3/4) + 0(1/10) = **0.333**

2. F3(Humidity):
   High has (1 Y, 2 N)
   Normal has (0 Y, 1 N)
   Gini Index for High = $1 – (1/3)^2 – (2/3)^2 = 0.444$
   Gini Index for Normal = $1 – (0)^2 – (1/1)^2 = 0$

   So, now we calculate the weighted Gini Index for **Humidity**

   = 0.444(3/4) + 0(1/4) = **0.333**

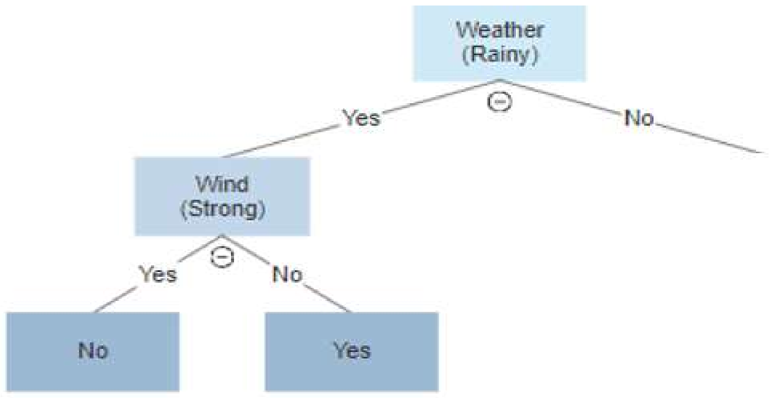
3. F4(Wind):
   Weak has (1 Y, 0 N)
   Strong has (0 Y, 3 N)
   Gini Index for Weak = $1 – (1/1)^2 – (0)^2 = 0$
   Gini Index for Strong = $1 – (0)^2 – (3/3)^2 = 0$

   So, now we calculate the weighted Gini Index for **Wind**

   = 0(1/4) + 0(3/4) = **0**

From the previous calculation we can see that the lowest Gini Index = 0. So, it will be the child of the Rainy condition.

The corresponding branch will be like the following:



Then we moved to the second branch where the weather is NOT rainy, the corresponding table was:

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

We calculated the Gini index for each feature as well:

1. F2(Temperature):
   Hot has (3 Y, 1 N)
   Mild has (2 Y, 0 N)
   Cool has (0 Y, 0 N)
   Gini Index for Hot = $1 – (3/4)^2 – (1/4)^2 = 0.375$
   Gini Index for Mild = $1 – (2/2)^2 – (0)^2 = 0$
   Gini Index for Cool = $1 – (0)^2 – (0)^2 = 1$

   we calculated the weighted Gini Index for **Temperature**

= 0.375(4/6) + 0(2/6) + 0(0) = **0.25**

2. F3(Humidity):
   High has (2 Y, 2 N)
   Normal has (2 Y, 0 N)
   Gini Index for High = 1 – $(2/4)^2$ – $(2/4)^2$ = 0.5
   Gini Index for Normal = 1 – $(2/2)^2$ – $(0)^2$ = 0

   we calculated the weighted Gini Index for **Humidity**

   = 0.5(4/6) + 0(2/6) = **0.333**

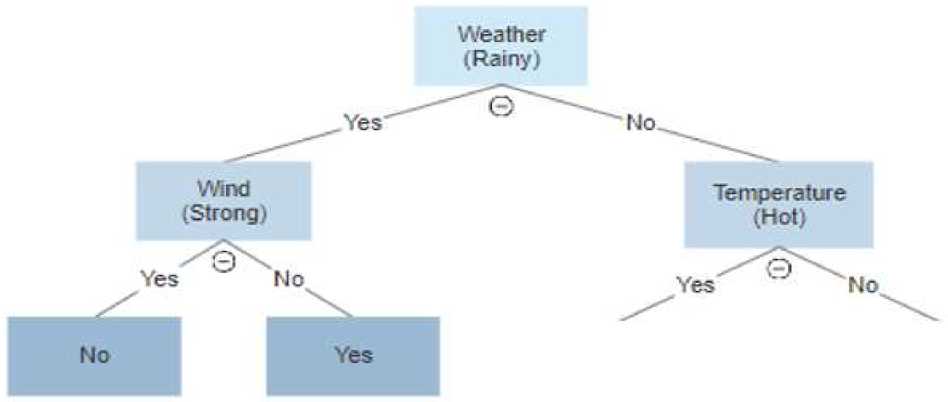
3. F4(Wind):
   Weak has (2 Yes, 1 No)
   Strong has (2 Yes, 1 No)
   Gini Index for Weak = 1 – $(2/3)^2$ – $(1/3)^2$ = 0.444
   Gini Index for Strong = 1 – $(2/3)^2$ – $(1/3)^2$ = 0.444

   So, now we calculate the weighted Gini Index for **Wind**

   = 0.444(3/6) + 0.444(3/6) = **0.444**

As we can see here Temperature has the lowest Gini, so it became the cild of the previous condition.

And since “hot” value has the lowest Gini it became the condition.



Working on the left branch where weather is not equal to rainy and Temperature is equal to Hot

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

1. F1(Weather):

   Cloudy has (1 Y, 1 N)

   Sunny has (1 Y, 1 N)

   Gini Index for Cloudy = $1 – (1/2)^2 – (1/2)^2 = 0.5$

   Gini Index for Sunny = $1 – (1/2)^2 – (1/2)^2 = 0.5$

   So, now we calculate the weighted Gini Index for **Weather**

   = 0.5(0.5) + 0.5(0.5) = **0.5**

2. F3(Humidity): it has High, and Normal.
   High has (1 Y, 2 N)
   Normal has (1 Y, 0 N)
   Gini Index for High = $1 – (1/3)^2 – (2/3)^2 = 0.444$
   Gini Index for Normal = $1 – (1)^2 – (0)^2 = 0$

   So, now we calculate the weighted Gini Index for **Humidity**

   = 0.444(3/4) + 0(1/4) = **0.333**

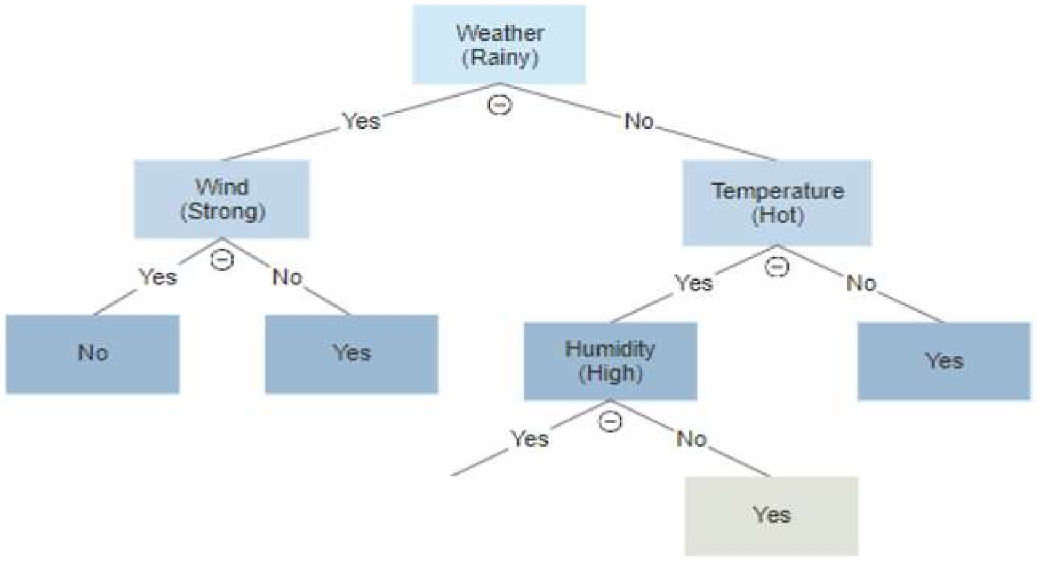
3. F4(Wind):
   Weak has (2 Y, 1 N)
   Strong has (0 Y, 1 N)
   Gini Index for Weak = $1 – (2/3)^2 – (1/3)^2 = 0.444$
   Gini Index for Strong = $1 – (0)^2 – (1)^2 = 0$

   So, now we calculate the weighted Gini Index for **Wind**

   = 0.444(3/4) + 0(1/4) = **0.333**

   So the next condition was Humidity with value equal to High

If the humidity is not High, we have only one results which is yes so it becomes a leaf node. On the other hand, we did further calculation to work on the left branch.



For the left branch where Humidity is high:

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

1. F1(Weather):

   Cloudy has (0 Y, 1 N)

   Sunny has (1 Yes, 1 No)

   Gini Index for Cloudy = 1 – $(0)^2$ – $(1)^2$ = 0

   Gini Index for Sunny = 1 – $(1/2)^2$ – $(1/2)^2$ = 0.5

   So, now we calculate the weighted Gini Index for **Weather**

   = 0(1/3) + 0.5(2/3) = **0.333**

2. F4(Wind):

Weak has (1 Y, 1 N)
Strong has (0 Y, 1 N)
Gini Index for Weak = 1 – $(1/2)^2$ – $(1/2)^2$ = 0.5
Gini Index for Strong = 1 – $(0)^2$ – $(1)^2$ = 0

So, now we calculate the weighted Gini Index for **Wind**

= 0.5(2/3) + 0(1/3) = **0.333**

We can select any feature of them to separate the data so for example we can select the weather with condition sunny

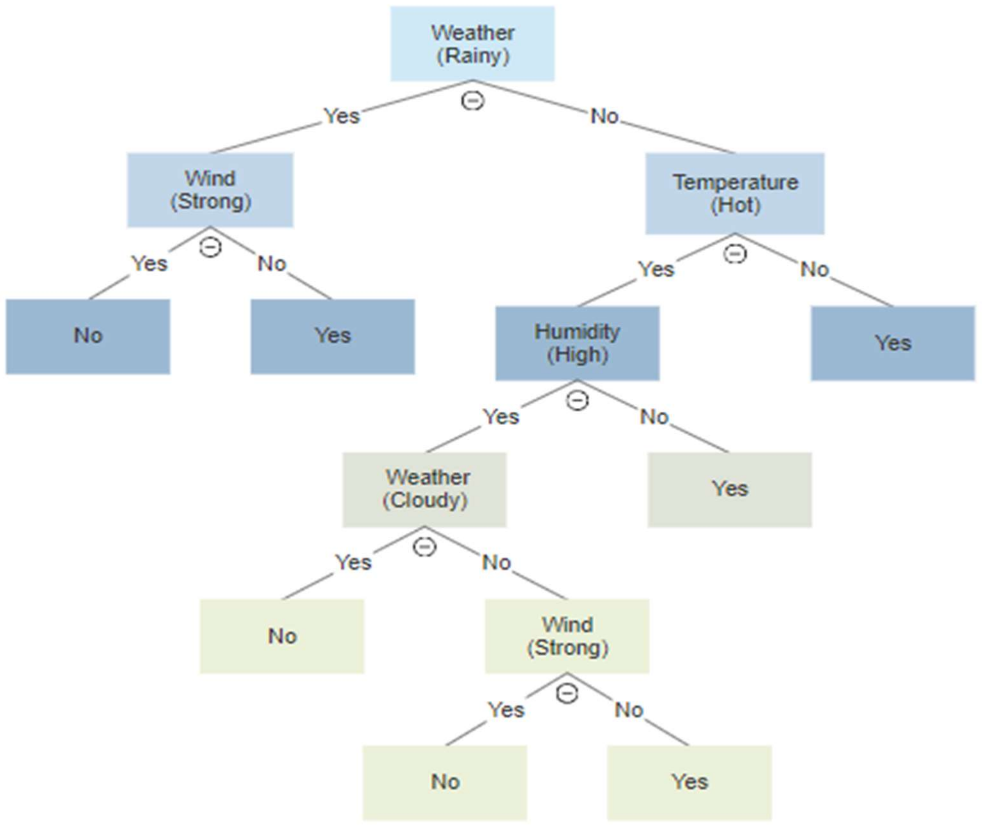
If the weather is sunny this is the corresponding table:

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

We can see that the label "hiking" can be easily separated with the wind attribute.
if the weather is not sunny

Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking
Cloudy	Hot	High	Weak	No

Hence the final decision tree was the following:



Q2: Please build a decision tree by using Information Gain (i.e., *IG(t,a)* $= Entropy(T) - Entropy(T|a)$),

Date / / Object

Decision Tree with information Gain

$E(\text{Hiking}) = E(5,5) = -\left(\frac{5}{10}\log_2\frac{5}{10}\right) - \left(\frac{5}{10}\log_2\frac{5}{10}\right)$
$= \boxed{1}$

		Hiking	
		yes	No
weather	cloudy	2	1
	sunny	2	1
	rainy	1	3

$E(\text{Hiking, weather}) = p(\text{cloudy}) * E(2,1) + p(\text{Sunny}) * E(2,1) + p(\text{rainy}) * E(1,3)$
$= \frac{3}{10} * 0,92 + \frac{4}{10} * 0,81 + \frac{3}{10} * 0,92$
$= 0,876$

$G(\text{Hiking, weather}) = E(\text{Hiking}) - E(\text{Hiking, weather})$
$= 1 - 0,876 = \boxed{0,124}$ ✓

		Hiking	
		yes	No
Temprature	Hot	2	2
	Mild	3	2
	Cool	0	1

$E(\text{Hiking, temprature}) = \frac{4}{10} * 1 + \frac{5}{10} * 0,97 + \frac{1}{10} * 0$
$= 0,885$

$G(\text{Hiking, temprature}) = 1 - 0,885 = \boxed{0,115}$

K.M.S.

Date / / Object

		Hiking yes	Hiking No
Humedity	High	3	4
	Normal	2	1

E(Hiking, Humedity) = $\frac{7}{10} * 0{,}99 + \frac{3}{10} * 0{,}92$
= 0,969

G(Hiking, Humedity) = 1 − 0,969 = $\boxed{0{,}031}$

		Hiking yes	Hiking No
wind	Strong	2	4
	weak	3	1

E(Hiking, wind) = $\frac{6}{10} * 0{,}92 + \frac{4}{10} * 0{,}81$
= 0,876

G(Hiking, wind) = 1 − 0,876 = $\boxed{0{,}124}$

$$\begin{tikzcd} & \boxed{\text{Weather}} \arrow[dl, "\text{yes}"'] \arrow[dr, "\text{No}"] \arrow[d, phantom, "\text{Rainy}"] & \\ {} & {} & {} \end{tikzcd}$$

E(Hiking, weather(Rainy)) = E(1,3) = 0,81

		Hiking | weather(rainy) yes	No
Temprature	Mild	1	2
	Cool	0	1

E(Hiking, temprature) = $\frac{3}{4} * 0{,}92 + \frac{1}{4} * 0 = 0{,}69$

G(Hiking, temprature) = 0,81 − 0,69 = $\boxed{0{,}12}$

K.M.S

$\boxed{\text{HiKing / weather(rainy)}}$

$\boxed{\text{Humidity}}$		yes	No
	High	1	2
	Normal	0	1

$E(\text{HiKing, Humidity}) = \frac{3}{4} * 0{,}92 + \frac{1}{4} * 0 = 0{,}69$

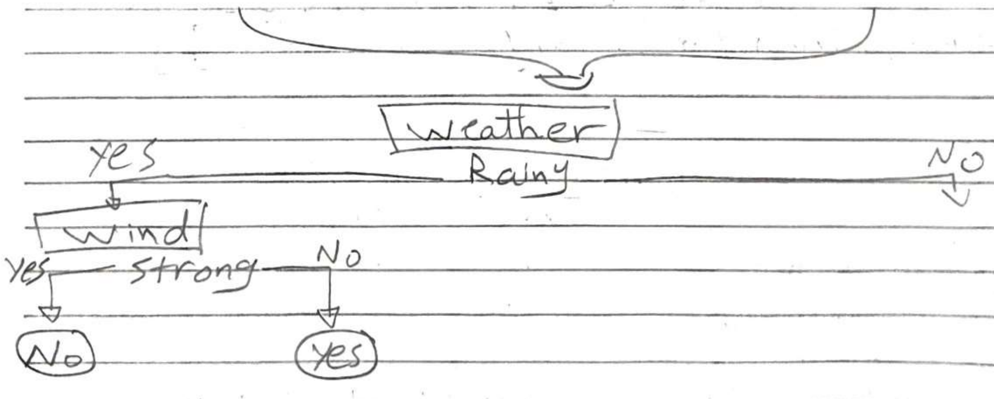
$G(\text{HiKing, Humedity}) = 0{,}81 - 0{,}69 = \boxed{0{,}12}$

$\boxed{\text{HiKing / weather(rainy)}}$

$\boxed{\text{wind}}$		yes	No
	Strong	0	3
	weak	1	0

$E(\text{HiKing, wind}) = \frac{3}{4} * 0 + \frac{1}{4} * 0 = 0$

$G(\text{HiKing, wind}) = 0{,}81 - 0 = \boxed{0{,}81}$ ✓



Date / / Object

E (HiKing , weather (Not rainy) = E (4, 2) = 0,92

Temprature	HiKing | weather (NoT rainy) yes	No
HoT	2	2
Mild	2	0

E(HiKing, temprature) = $4/6 * 1 + 2/4 * 0 = 0,6$
G(HiKing, temprature) = $0,92 - 0,6 = \boxed{0,32}$ ✓

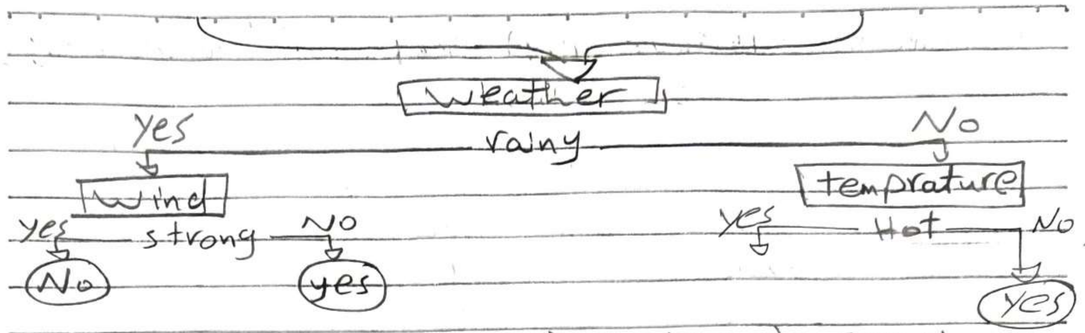
Humidity	HiKing | weather (NOT rainy) yes	No
High	2	2
Normal	2	0

E(HiKing, temprature) = $4/6 * 1 + 2/4 * 0 = 0,6$
G(HiKing, temprature) = $0,92 - 0,6 = \boxed{0,32}$

Wind	HiKing | weather (NOT rainy) yes	No
weak	2	1
strong	2	1

E(HiKing, wind) = $3/6 * 0,92 + 3/6 * 0,92 = 0,92$
G(HiKing, wind) = $0,92 - 0,92 = \boxed{0}$

K.M.S



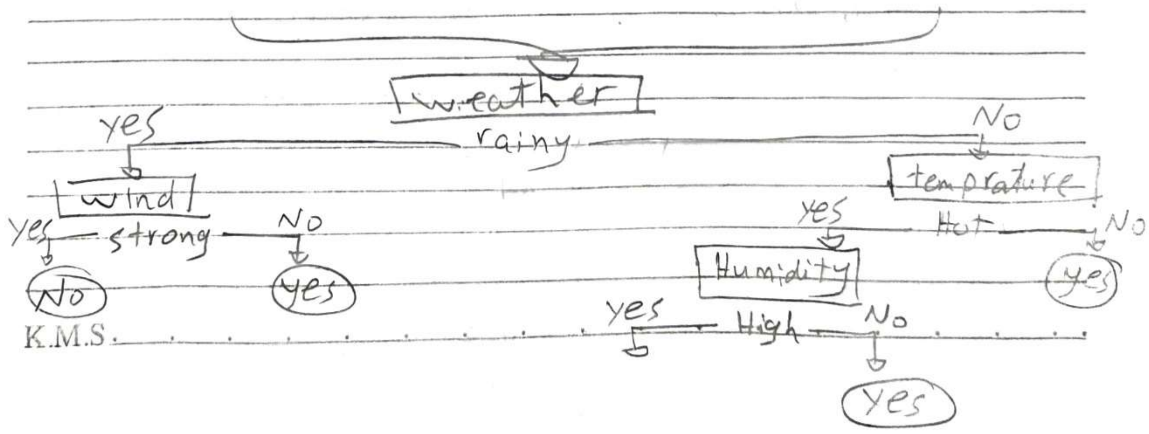
E (HiKing, weather(Notraing), temp(Hot) = E(2,2) = 1

$\boxed{\text{HiKing} \mid \text{weather(Not rainy)} \mid \text{temp(HoT)}}$		yes	No
$\boxed{\text{Humidity}}$	High	1	2
	Normal	1	0

E(HiKing, Humidity) = 3/4 * 0,92 + 1/4 * 0 = 0,69
G(HiKing, Humidity) = 1 - 0,69 = $\boxed{0,31}$ ✓

$\boxed{\text{HiKing} \mid \text{weather(NoTrainy)} \mid \text{temp(Hot)}}$		yes	No
$\boxed{\text{wind}}$	weak	2	1
	strong	0	1

E(HiKing, wind) = 3/4 * 0,92 + 1/4 * 0 = 0,69
G(HiKing, wind) = 1 - 0,69 = $\boxed{0,31}$



Date / / Object

E(HiKing, weather(Notrainy) | temp(Hot) | Humidity(high))
= E(1,2) = 0,92

		HiKing	
		yes	No
Weather	cloudy	0	1
	Sunny	1	1

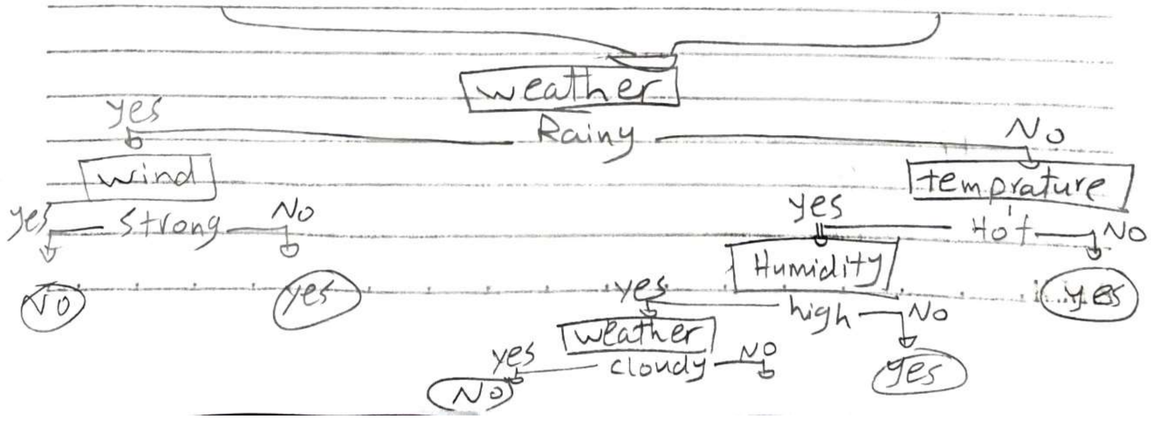
E(HiKing, weather) = $\frac{1}{3} * 0 + \frac{2}{3} * 1 = 0,6$
G(HiKing, weather) = $0,92 - 0,6 = \boxed{0,32}$ ✓

		HiKing	
		yes	No
Temprature	Hot	1	2

E(HiKing, temprature) = $\frac{3}{3} * 0,92 = 0,92$
G(HiKing, temprature) = $0,92 - 0,92 = \boxed{0}$

		HiKing	
		yes	No
wind	weak	1	1
	Strong	0	1

E(HiKing, wind) = $\frac{2}{3} * 1 + \frac{1}{3} * 0 = 0,6$
G(HiKing, wind) = $0,92 - 0,6 = \boxed{0,32}$



Date / / Object

E(Hiking, weather(Notraindy) | temp(Hot) | Humidity(high) | Weather(Not cloudy)) = E(1,1) = 1

Temprature		Hiking yes	Hiking No
	Hot	1	1

E(Hiking, temprature) = 2/2 * 1 = 1
G(Hiking, temprature) = 1 - 1 = $\boxed{0}$

Humidity		Hiking yes	Hiking No
	High	1	1

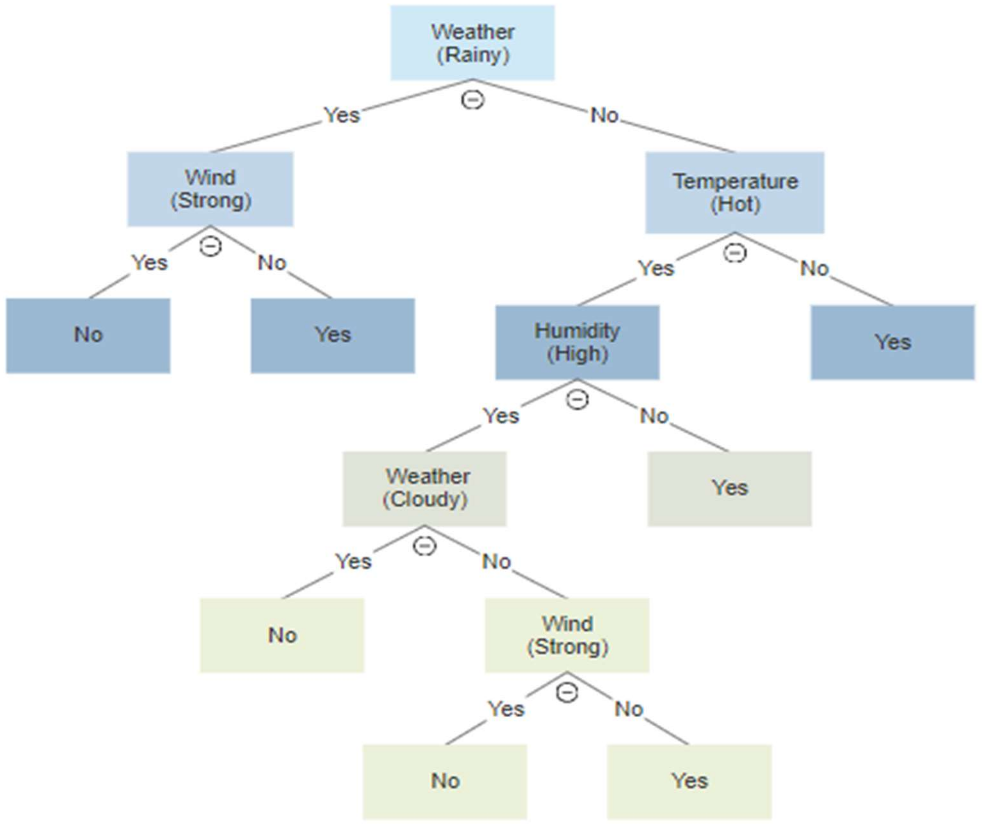
E(Hiking, Humidity) = 2/2 * 1 = 1
G(Hiking, Humidity) = 1 - 1 = $\boxed{0}$

		Hiking yes	Hiking No
Wind	weak	1	0
	strong	0	1

E(Hiking, Wind) = 1/2 * 0 + 1/2 * 0 = 0
G(Hiking, wind) = 1 - 0 = $\boxed{1}$ ✓

K.M.S.

The final tree was the same as the resulted from the Gini index



Q3: Please compare the advantages and disadvantages between Gini Index and Information Gain.

Gini impurities tell us the probability of misclassification of an observation, while Information gain is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees. Information gain is calculated by comparing the entropy of the dataset before and after a transformation. Hence, we can summarize the advantages and disadvantages of the both approaches in the following points:

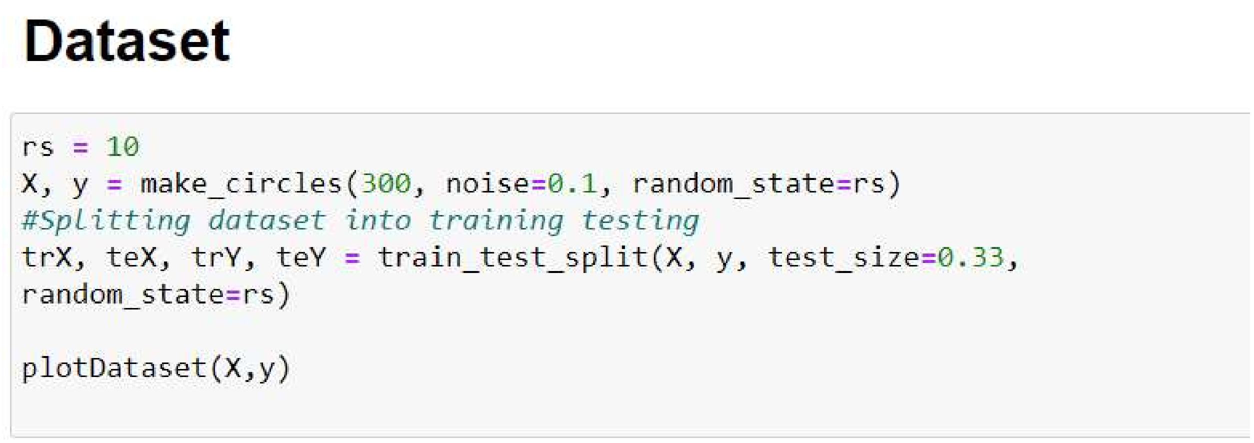
1. Gini index is simpler, faster, and computationally efficient especially with the large dataset as the information gain does require higher computation power for calculating the log calculations in the entropy.
2. Gini's maximum impurity is 0.5 and maximum purity is 0 while Entropy's maximum impurity is 1 and maximum purity is 0. This makes information gain is a little bit easier to interpret.

Part 2: Programming Questions

Q4 (5 Marks): Apply decision tree to classify testing set, get the accuracy of the result, and plot the decision boundary as we showed in the lab.

This step has been broken up into three steps:

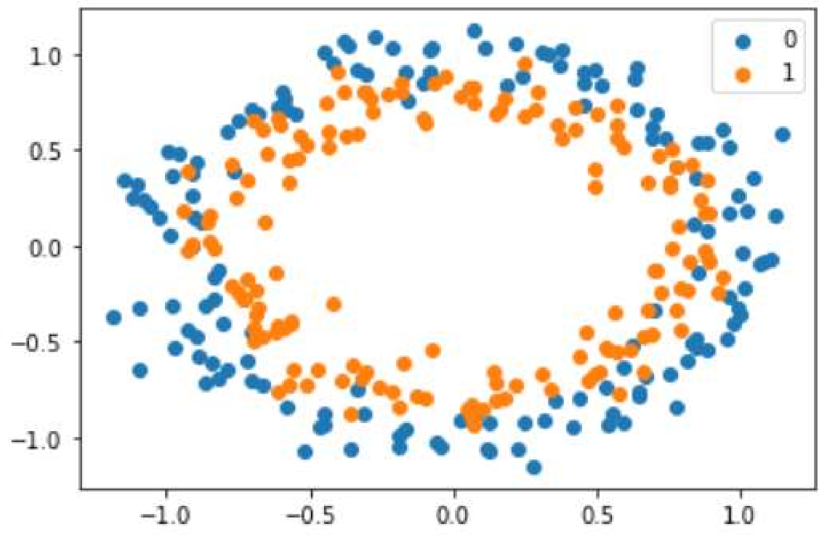
The first one is loading the dataset:



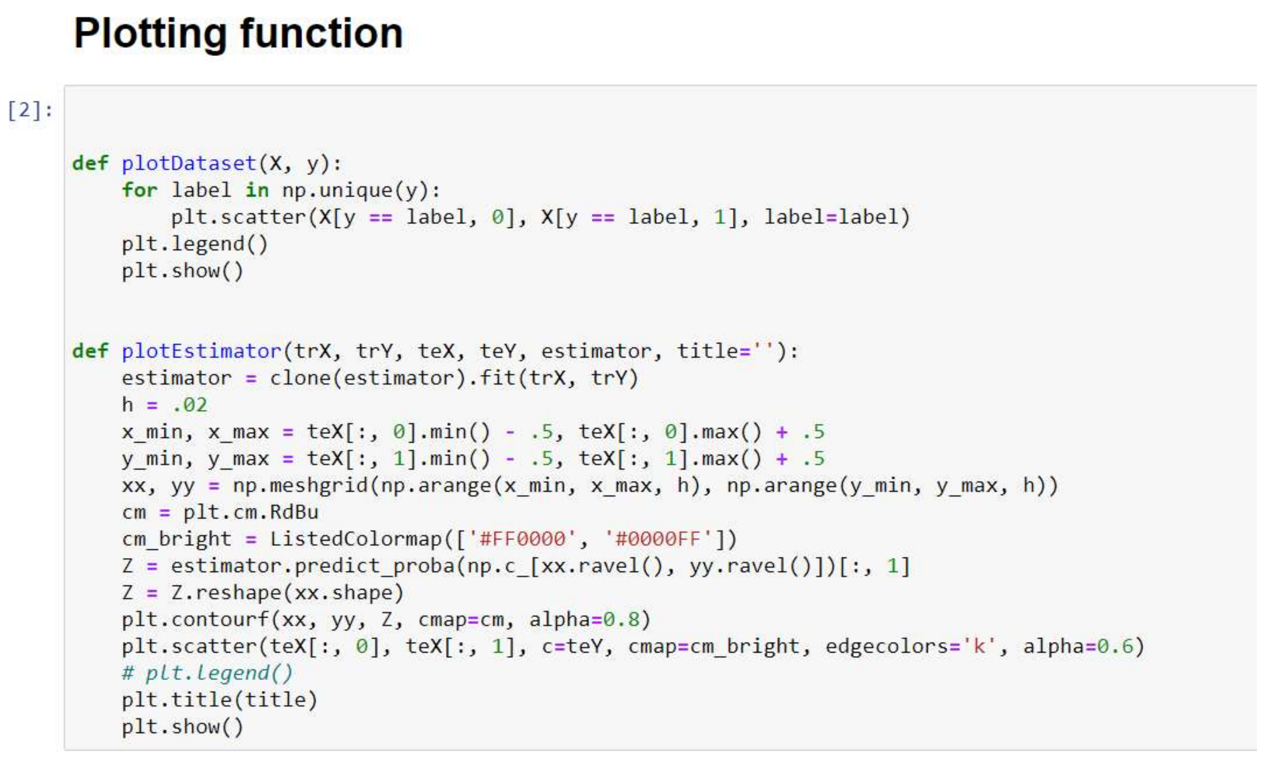
Dataset

```
rs = 10
X, y = make_circles(300, noise=0.1, random_state=rs)
#Splitting dataset into training testing
trX, teX, trY, teY = train_test_split(X, y, test_size=0.33,
random_state=rs)

plotDataset(X,y)
```



The second step is to define the plotting function:



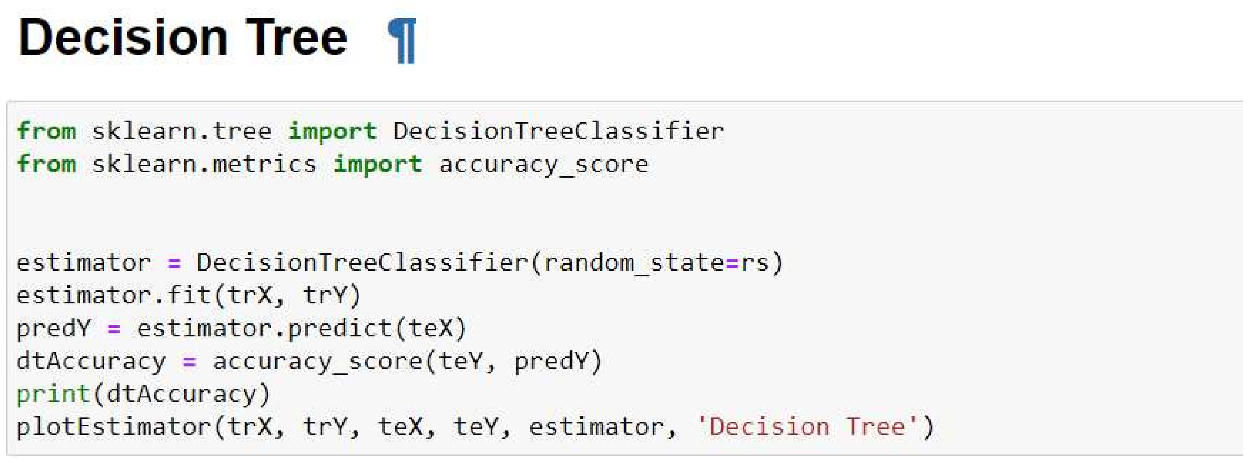
Plotting function

```
[2]:

def plotDataset(X, y):
    for label in np.unique(y):
        plt.scatter(X[y == label, 0], X[y == label, 1], label=label)
    plt.legend()
    plt.show()


def plotEstimator(trX, trY, teX, teY, estimator, title=''):
    estimator = clone(estimator).fit(trX, trY)
    h = .02
    x_min, x_max = teX[:, 0].min() - .5, teX[:, 0].max() + .5
    y_min, y_max = teX[:, 1].min() - .5, teX[:, 1].max() + .5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    cm = plt.cm.RdBu
    cm_bright = ListedColormap(['#FF0000', '#0000FF'])
    Z = estimator.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=cm, alpha=0.8)
    plt.scatter(teX[:, 0], teX[:, 1], c=teY, cmap=cm_bright, edgecolors='k', alpha=0.6)
    # plt.legend()
    plt.title(title)
    plt.show()
```

The third step is to train the decision tree and to plot the decision boundaries using the decision boundaries plotting function previously defined.



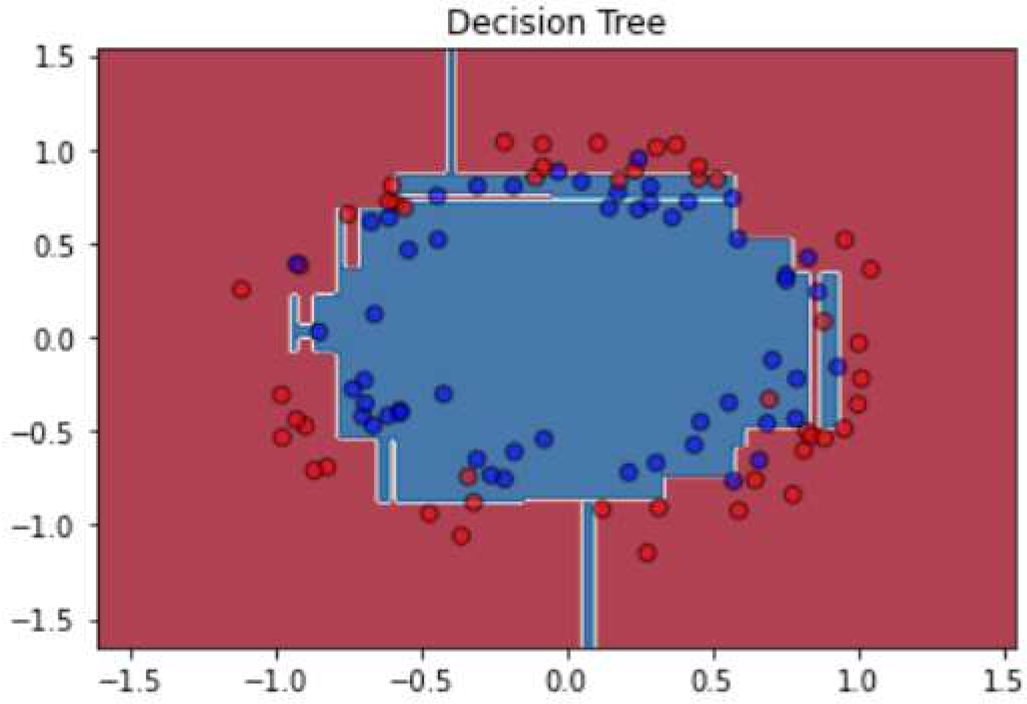
Decision Tree ¶

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score


estimator = DecisionTreeClassifier(random_state=rs)
estimator.fit(trX, trY)
predY = estimator.predict(teX)
dtAccuracy = accuracy_score(teY, predY)
print(dtAccuracy)
plotEstimator(trX, trY, teX, teY, estimator, 'Decision Tree')
```

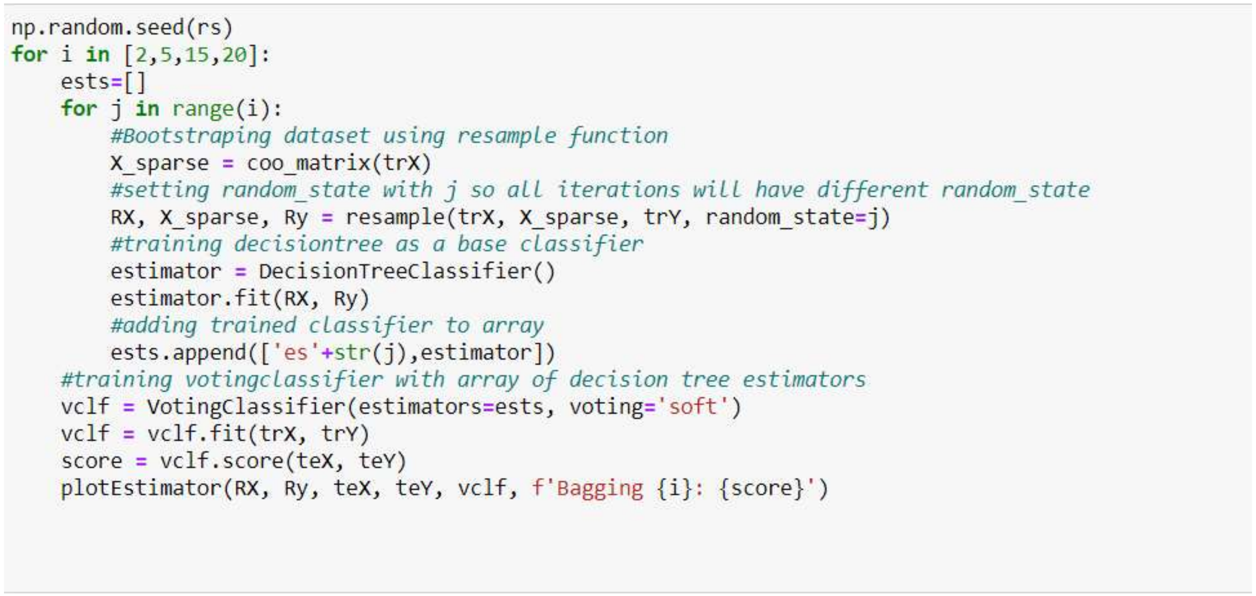
The resulted accuracy was 0.8181818181818182

And here was the resulted decision boundaries plot



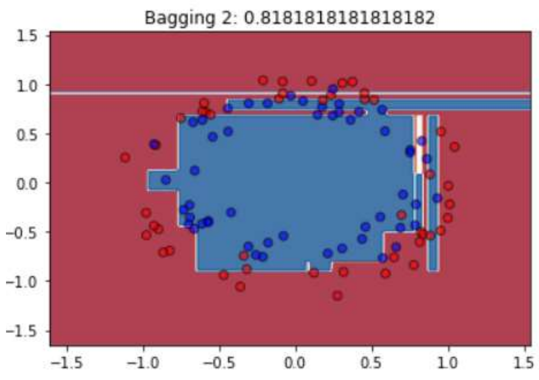
Q4: Using decision tree as base-estimator and write bagging algorithm from scratch, set the number of estimators as 2, 5, 15, 20 respectively, and generate the results accordingly (i.e., accuracy and decision boundary)

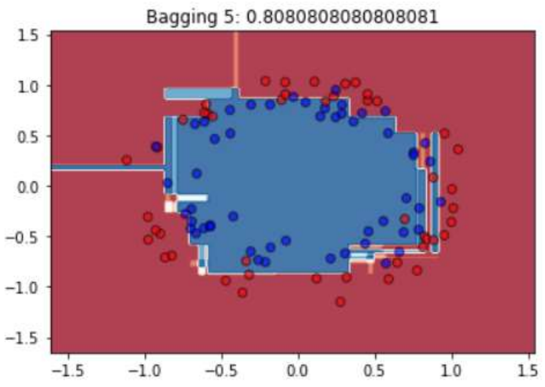
We first created different subset of the data the trained models with each subset, appended the resulted models in an array then we applied the voting classifier to aggregate the results from all models created. This has been implemented in the following lines of code:

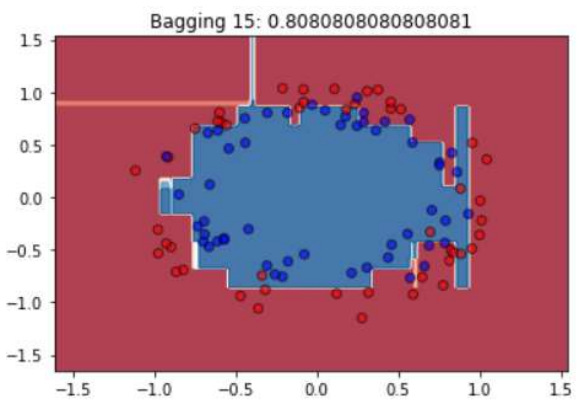


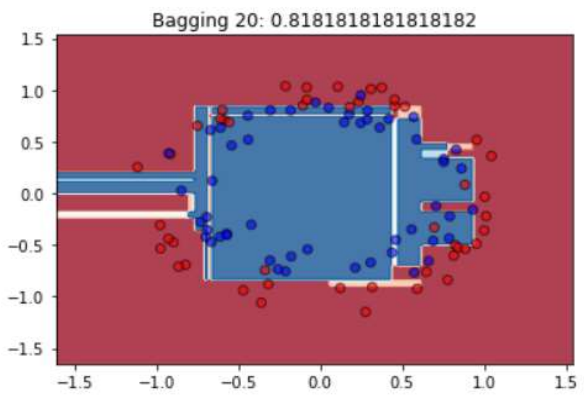
```
np.random.seed(rs)
for i in [2,5,15,20]:
    ests=[]
    for j in range(i):
        #Bootstraping dataset using resample function
        X_sparse = coo_matrix(trX)
        #setting random_state with j so all iterations will have different random_state
        RX, X_sparse, Ry = resample(trX, X_sparse, trY, random_state=j)
        #training decisiontree as a base classifier
        estimator = DecisionTreeClassifier()
        estimator.fit(RX, Ry)
        #adding trained classifier to array
        ests.append(['es'+str(j),estimator])
    #training votingclassifier with array of decision tree estimators
    vclf = VotingClassifier(estimators=ests, voting='soft')
    vclf = vclf.fit(trX, trY)
    score = vclf.score(teX, teY)
    plotEstimator(RX, Ry, teX, teY, vclf, f'Bagging {i}: {score}')
```

These were the results obtained:









Q5: Explain why bagging can reduce the variance and mitigate the overfitting problem

Bagging is an approach of the ensemble method in which predictions from multiple separate models are combined together. Each model is trained on a random subset of the data so each models has its own bias and variance. Since these models are very strong learners, their bias is very low, yet their variance is very high. To solve this problem, voting on the final prediction on each separate model is being applied so that the final predictions are more generalized.

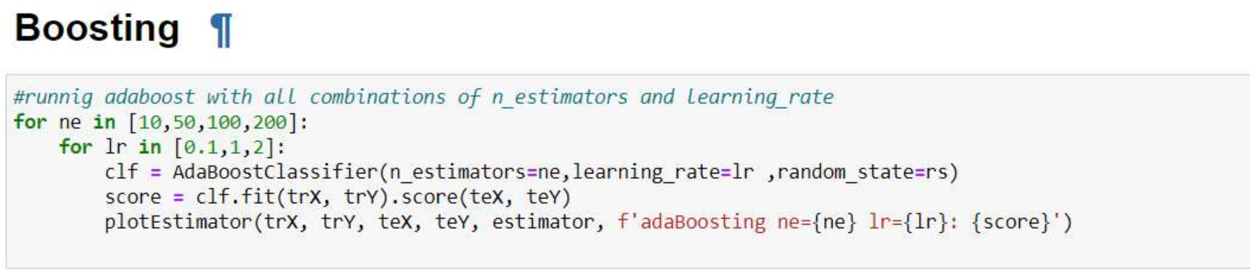
Bagging, as an ensemble method, attempts to reduce the chance of overfitting complex models.

- It trains a large number of “strong” learners in parallel on different subset of the data selected with replacement.
- A strong learner is a model that’s relatively unconstrained.
- Bagging then combines all the strong learners together in order to “smooth out” their predictions then achieve more generalization and reduce the variance.

Q6 (15 Marks): There are 2 important hyperparameters in AdaBoost, i.e., the number of estimators (ne), and learning rate (lr). Please plot 12 subfigures as the following table's setup. Each figure should plot the decision boundary and each of their title should be the same format as {n_estimaotrs}, {learning_rate}, {accuracy}

ne=10; lr=0.1	ne=50; lr=0.1	ne=100; lr=0.1	ne=200; lr=0.1
ne=10; lr=1	ne=50; lr=1	ne=100; lr=1	ne=200; lr=1
ne=10; lr=2	ne=50; lr=2	ne=100; lr=2	ne=200; lr=2

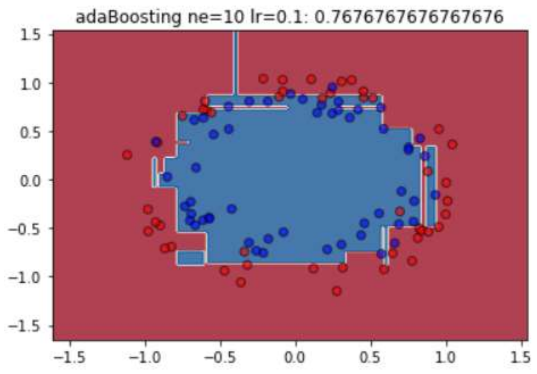
This has been implemented using the following lines of code:

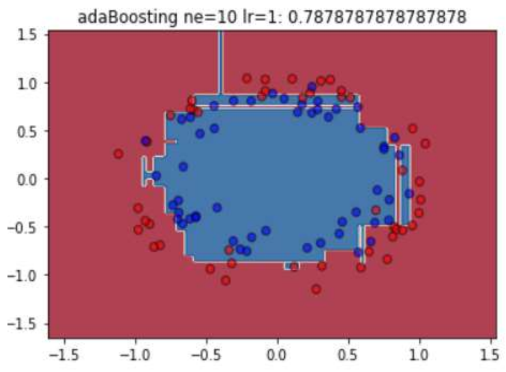


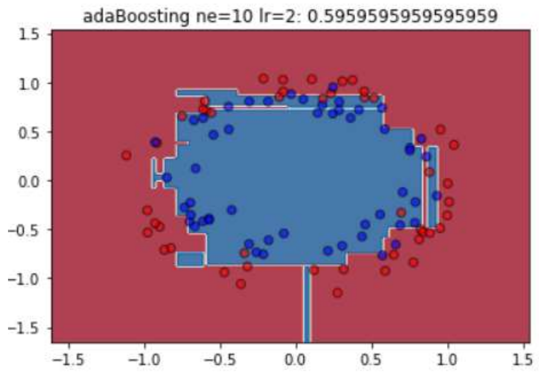
Boosting ¶

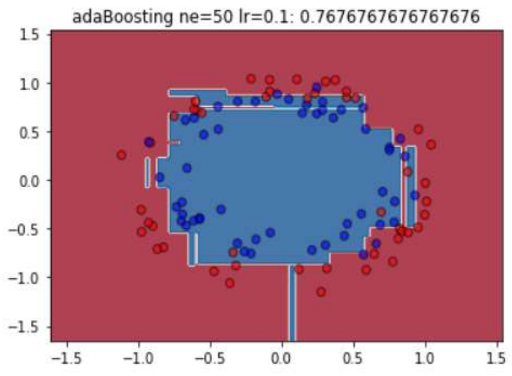
```python
#runnig adaboost with all combinations of n_estimators and learning_rate
for ne in [10,50,100,200]:
    for lr in [0.1,1,2]:
        clf = AdaBoostClassifier(n_estimators=ne,learning_rate=lr ,random_state=rs)
        score = clf.fit(trX, trY).score(teX, teY)
        plotEstimator(trX, trY, teX, teY, estimator, f'adaBoosting ne={ne} lr={lr}: {score}')
```

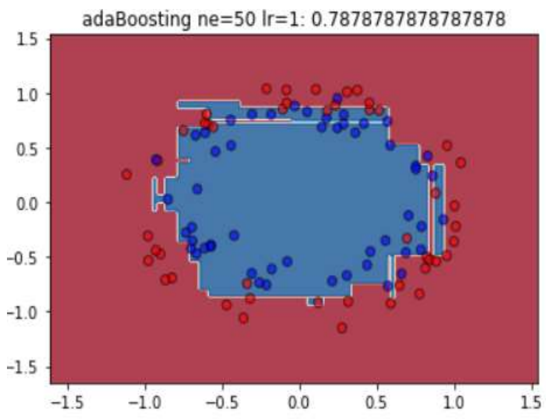
These were results:

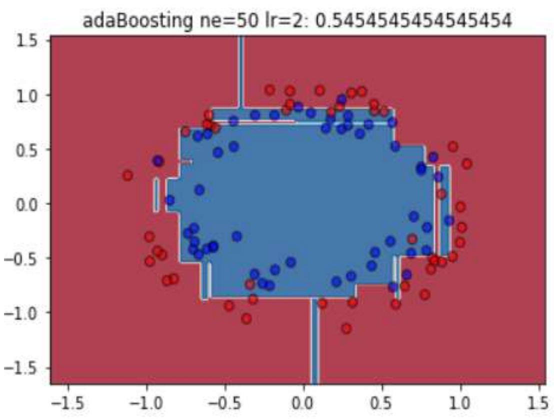


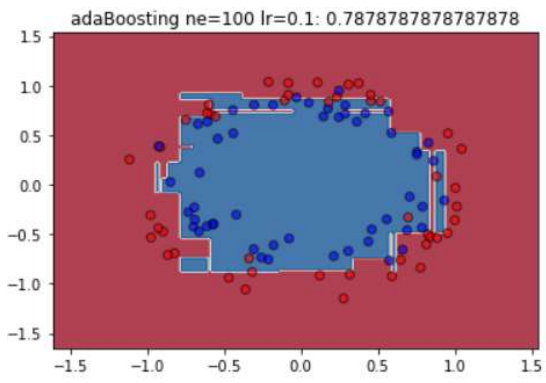


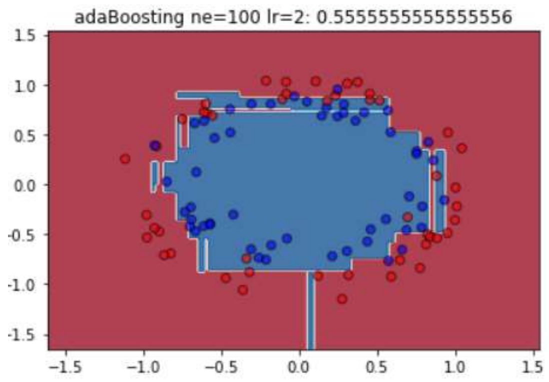


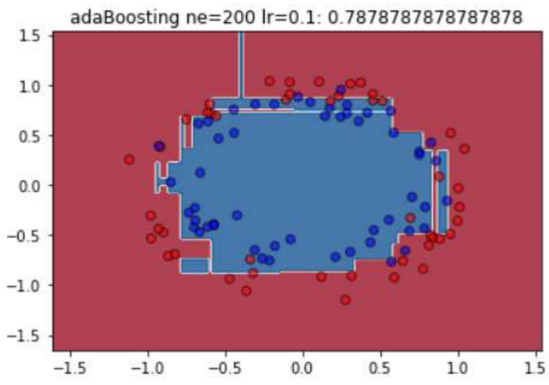


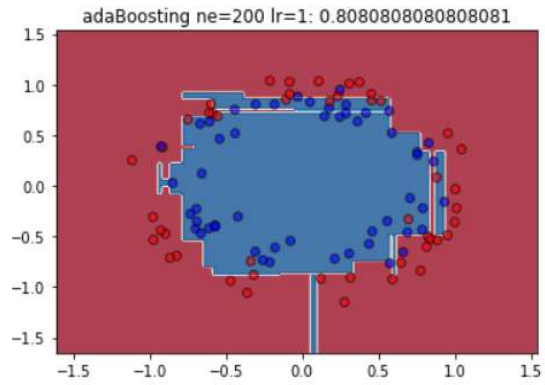

adaBoosting ne=50 lr=1: 0.7878787878787878

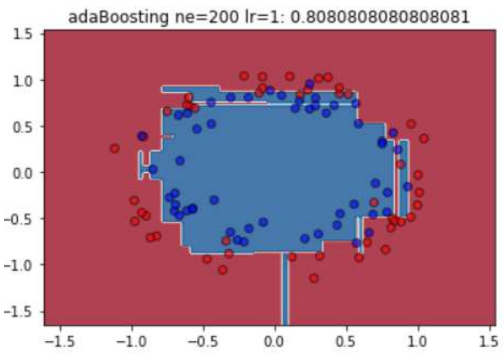

adaBoosting ne=50 lr=2: 0.5454545454545454

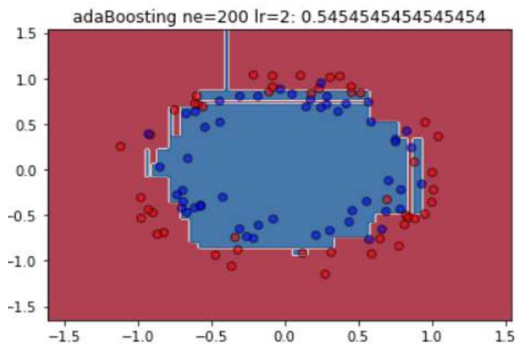

adaBoosting ne=100 lr=0.1: 0.7878787878787878


adaBoosting ne=100 lr=2: 0.5555555555555556


adaBoosting ne=200 lr=0.1: 0.7878787878787878


adaBoosting ne=200 lr=1: 0.8080808080808081


adaBoosting ne=200 lr=1: 0.8080808080808081
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


adaBoosting ne=200 lr=2: 0.5454545454545454
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5