Decision Tree and Ensemble Learning Assignment 4

Applied Machine Learning ELG5255 [EG]

Group name: Group 6

Ahmed Fares Saad Eldin Khalifa

Abdallah Medhat Mohamed Rashed

1. Part 1

1.1 Building a decision tree by using Gini.

Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Label)
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

GINI (t) =1- $\sum_j^n (p(j\backslash t))^2$

GINI (split) = $\sum_{i=1}^{k} (\mathbf{n(i)/n})\mathbf{GINI(i)}$

Calculation of Gini Index for Weather:

Weather (F1)	Yes	No	counts
Cloudy	2	1	3
Sunny	2	1	3
Rainy	1	3	4

Gini (Weather=Cloudy) =1 - (2/3) ² - (1/3) ² = 0.444

Gini (Weather=Sunny) =1 - (2/3) ² - (1/3) ² = 0.444

Gini (Weather=Rainy) =1 - (1/4) ² - (3/4) ²= 0.375

Now weighted sum of the Gini Indices can be calculated:

Gini (Weather) = (3/10)*0.444 + (3/10)*0.444 + (4/10)*0.375 = 0.4164

Calculation of Gini Index for Temperature:

Temperature(F2)	Yes	No	counts
Hot	2	2	4
Mild	3	2	5
Cool	0	1	1

Gini (Temperature =Hot) = 1 - (2/4) ²- (2/4) ²= 0.5

Gini (Temperature =Mild) = 1 - (3/5) ²- (2/5) ²= 0.48

Gini (Temperature =Cool) = 1 - (0/1) ²- (1/1) ²= 0

Now weighted sum of the Gini Indices can be calculated:

Gini (Temperature) = (4/10)*0.5 + (5/10)*0.48 + (1/10)*0 = 0.44

Calculation of Gini Index for Humidity:

Humidity(F3)	Yes	No	Counts
High	3	4	7
Normal	2	1	3

Gini (Humidity=High) = 1-(3/7) ² - (4/7) ² = 0.49

Gini (Humidity=Normal) = 1-(2/3) ² - (1/3) ² = 0.444

Now weighted sum of the Gini Indices can be calculated:

Gini (Humidity) = (7/10)*0.49 + (3/10)*0.444=0.4762

Calculation of Gini Index for Wind:

Wind(F4)	Yes	No	counts
Strong	2	4	6
Weak	3	1	4

Gini (Wind=Strong) = 1 - (2/6) ² - (4/6) ² = 0.444

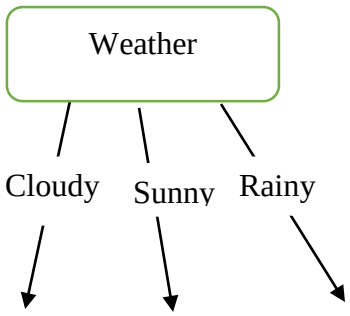
Gini (Wind= Weak) = 1 - (3/4) ² - (1/4) ² = 0.375

Now weighted sum of the Gini Indices can be calculated:

Gini (Wind) = (6/10)*0.444 + (4/10)*0.375 =0.4164

Feature	Gini Index
Weather (F1)	0.4164
Temperature(F2)	0.44
Humidity(F3)	0.4762
Wind(F4)	0.4164

We obtain (Weather and wind) have the lowest Gini Index and we will select (Weather) as the root node.



Calculation of Gini Index for the (cloudy) branch of Weather.

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

Calculation of Gini Index for Temperature:

Temperature (F2)	Yes	NO	Counts
Hot	1	1	2
Mild	1	0	1

Gini (Temperature =Hot) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Temperature =Mild) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Humidity:

Humidity	Yes	NO	Counts
High	1	1	2
Normal	1	0	1

Gini (Humidity =High) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Humidity =Normal) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Wind:

Wind	Yes	NO	Counts
Weak	1	1	2
Strong	1	0	1

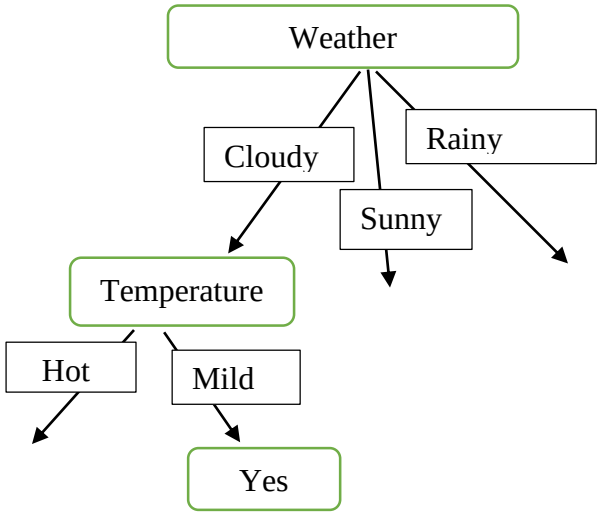
Gini (Wind =Weak) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Wind =Strong) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

We obtain (Temperature, Humidity and wind) have the lowest Gini Index and we will select (Temperature). We will split the (Weather = Cloudy) node by using the (Temperature) feature.



Calculation of Gini Index for the (Hot) branch of Temperature.

Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Label)
cloudy	Hot	High	Weak	No
cloudy	Hot	Normal	Weak	Yes

Calculation of Gini Index for Humidity:

Humidity	Yes	NO	Counts
High	0	1	1
Normal	1	0	1

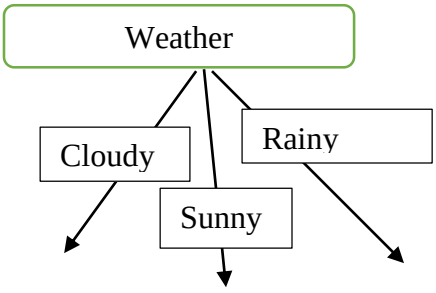
Gini (Humidity =High) =1 - (0) ² - (1) ² = 0

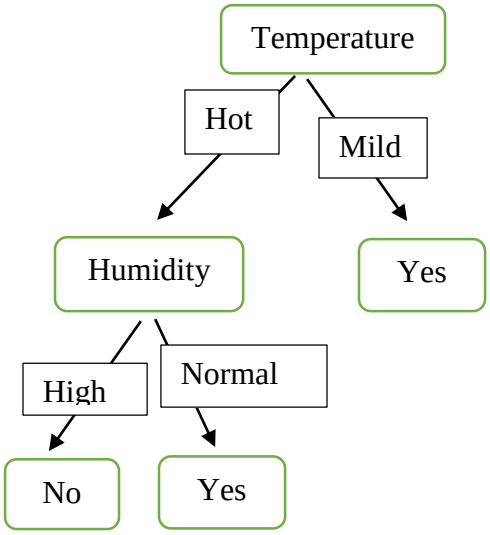
Gini (Humidity =Normal) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

Gini (Humidity) = (1/2)* 0 + (1/2)*0 = 0

We obtain Gini (Humidity) =0 is the least and the Humidity is in closer to the left than the wind. We select the Humidity. We will split the (Weather = Cloudy and Temperature=Hot) node by using the (Humidity) feature.





Calculation of Gini Index for the (Sunny) branch of Weather.

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

Calculation of Gini Index for Temperature:

Temperature (F2)	Yes	NO	Counts
Hot	1	1	2
Mild	1	0	1

Gini (Temperature =Hot) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Temperature =Mild) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Humidity:

Humidity	Yes	NO	Counts
High	1	1	2
Normal	1	0	1

Gini (Humidity =High) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Humidity =Normal) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Wind:

Wind	Yes	NO	Counts
Weak	1	0	1
Strong	1	1	2

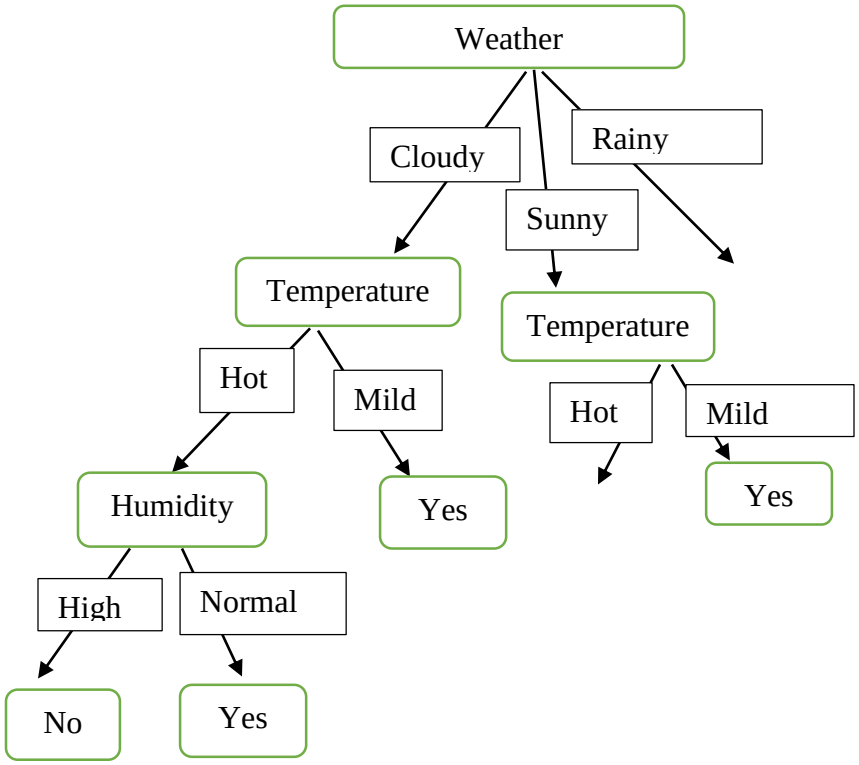
Gini (Wind =Weak) =1 - (1) ² - (0) ² = 0

Gini (Wind =Strong) =1 - (1/2) ² - (1/2) ² = 0.5

Now weighted sum of the Gini Indices can be calculated:

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

We obtain (Temperature, Humidity and wind) have the lowest Gini Index and we will select (Temperature). We will split the (Weather = Sunny) node by using the (Temperature) feature.



Calculation of Gini Index for the (Hot) branch of Temperature.

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

Calculation of Gini Index for Humidity:

Humidity	Yes	NO	Counts

High	1	1	2
Normal	0	0	0

Gini (Humidity =High) =1 - (1/2) ² - (1/2) ² = 0.5

Gini (Humidity =Normal) =1 - (0) ² - (0) ² = 1

Now weighted sum of the Gini Indices can be calculated:

Gini (Humidity) = (2/2)* 0.5 + 0 = 0.5

Calculation of Gini Index for Wind:

Wind	Yes	NO	Counts
Weak	1	0	1
Strong	0	1	1

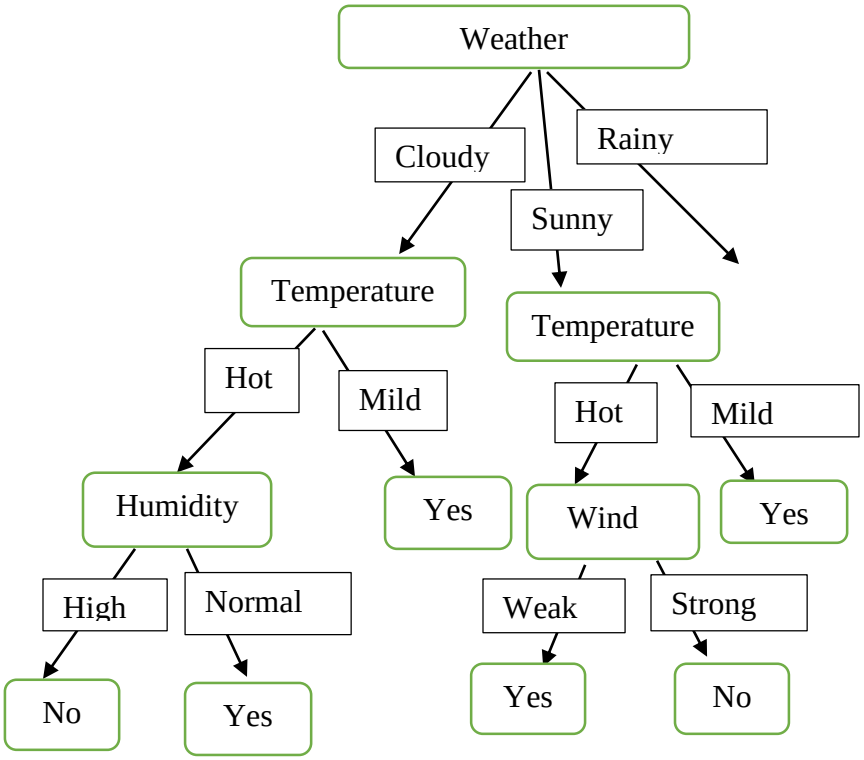
Gini (Wind =Weak) =1 - (1/1) ² - (0) ² = 0

Gini (Wind =Strong) =1 - (0) ² - (1/1) ² = 0

Now weighted sum of the Gini Indices can be calculated:

Gini (Wind) = 0 + 0 = 0

We obtain Gini (Wind) =0 is the least. We select the Wind. We will split the (Weather = Sunny and Temperature=Hot) node by using the (Wind) feature.



Calculation of Gini Index for the (Rainy) branch of Weather.

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

Calculation of Gini Index for Temperature:

Temperature (F2)	Yes	NO	Counts
Mild	1	2	3
Cool	0	1	1

Gini (Temperature =Mild) =1 - (1/3) ² - (2/3) ² = 0.444

Gini (Temperature =Mild) =1 - (0) ² - (1) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Humidity:

Humidity	Yes	NO	Counts
High	1	2	3
Normal	0	1	1

Gini (Humidity =High) =1 - (1/3) ² - (2/3) ² = 0.444

Gini (Humidity =Normal) =1 - (0) ² - (1) ² = 0

Now weighted sum of the Gini Indices can be calculated:

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

Calculation of Gini Index for Wind:

Wind	Yes	NO	Counts
Strong	0	3	3
Weak	1	0	1

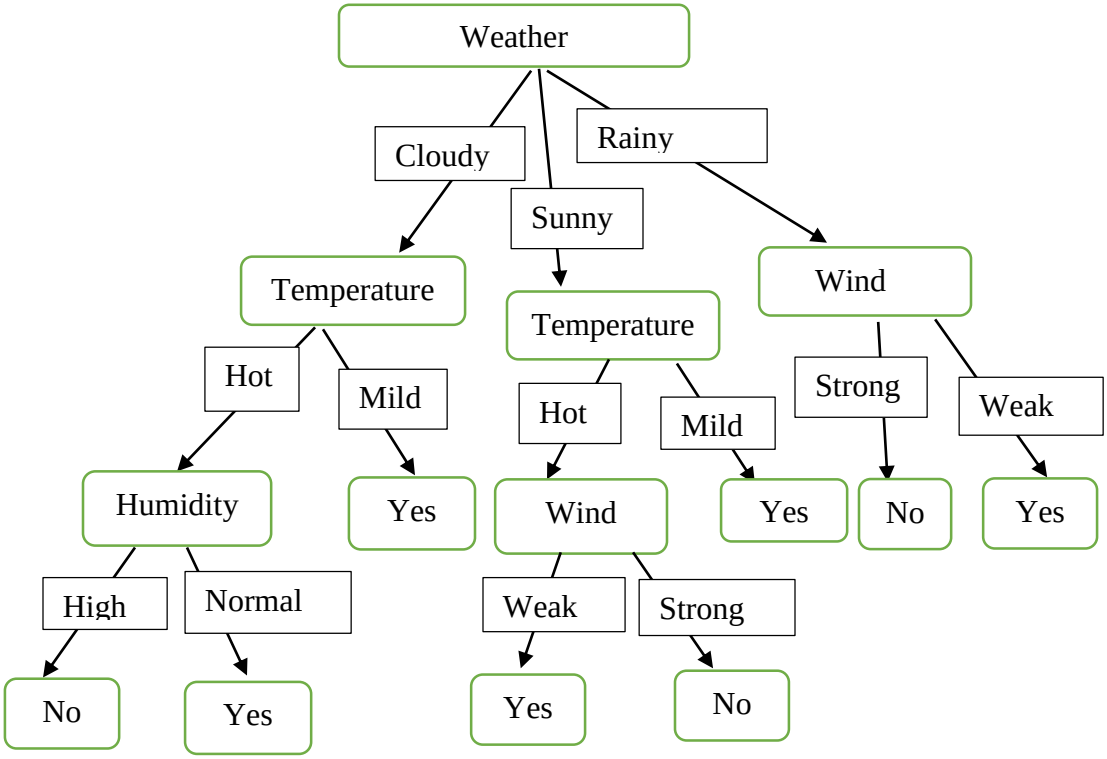
Gini (Wind =Weak) =1 - (0) ² - (1) ² = 0

Gini (Wind =Strong) =1 - (1) ² - (0) ² = 0

Now weighted sum of the Gini Indices can be calculated:

Gini (Wind) = 0 + 0 = 0

We obtain (wind) have the lowest Gini Index and we will select (wind). We will split the (Weather = Rainy) node by using the (Wind) feature.



1.2 Building a decision tree by using Information Gain.

Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Label)
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

Entropy= $-\sum P(j|t) \log 2 P(j|t)$

Entropy (Hiking (label)) = - 5/10*$\log_2$ (5/10) -5/10*$\log_2$ (5/10) =1

Information gain:

GAIN (split) =Entropy (p)- $\sum_{i=1}^{k}(\mathbf{n}(\mathbf{i})/\mathbf{n})\mathbf{Entropy}(\mathbf{i})$

Calculation of Information gain for Weather:

IG (Hiking (label), weather) = 1-(3/10) (-1/3*$\log_2$ (1/3)-2/3*$\log_2$ (2/3)) -(3/10) (-1/3*$\log_2$ (1/3)-2/3*$\log_2$ (2/3)) -(4/10) (-1/4*$\log_2$ (1/4)-3/4*$\log_2$ (3/4)) = 0.1245

Calculation of Information gain for Temperature:

IG (Hiking (label), Temperature) = 1-(4/10) (-2/4*$\log_2$ (2/4)-2/4*$\log_2$ (2/4)) - (5/10) (-3/5*$\log_2$ (3/5)-2/5*$\log_2$ (2/5)) - (1/10) (-1/1*$\log_2$ (1/1)) = 0.1145

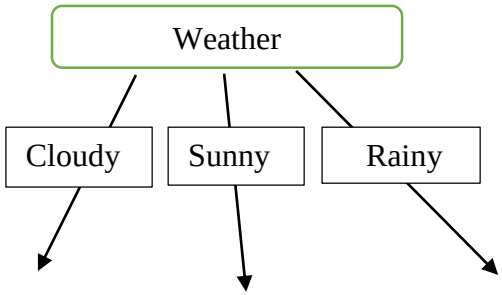
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 1-(7/10) (-3/7*$\log_2$ (3/7)-4/7*$\log_2$ (4/7)) - (3/10) (-1/3*$\log_2$ (1/3)-2/3*$\log_2$ (2/3)) = 0.03485

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 1-(6/10) (-4/6*$\log_2$ (4/6)-2/6*$\log_2$ (2/6)) - (4/10) (-3/4*$\log_2$ (3/4)-1/4*$\log_2$ (1/4)) = 0.1245

We obtain (Weather and wind) have the highest Information gain and we will select (Weather) as the root node.



Calculation of Information gain for the (cloudy) branch of Weather.

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

Entropy (Hiking (label)) = - 1/3*$\log_2$ (1/3) -2/3*$\log_2$ (2/3) = 0.918

Calculation of Information gain for Temperature:

IG (Hiking (label), Temperature) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1) = 0.251

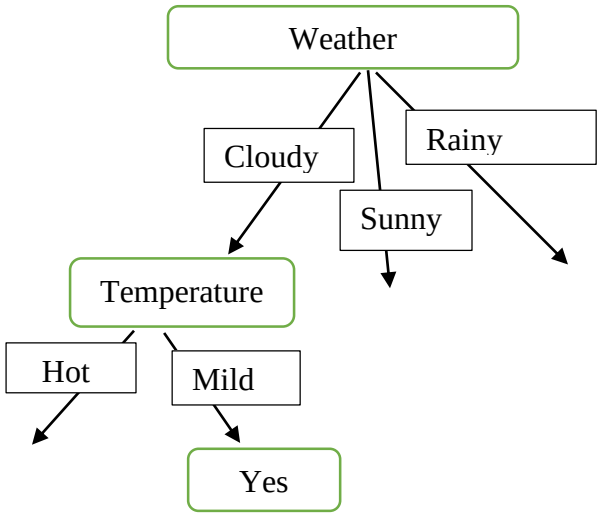
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1) = 0.251

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1) = 0.251

We obtain (Temperature, Humidity and wind) have the highest Information gain and we will select (Temperature). We will split the (Weather = Cloudy) node by using the (Temperature) feature.



Calculation of Information gain for the (Hot) branch of Temperature.

Weather (F1)	Temperature (F2)	Humidity (F3)	Wind (F4)	Hiking (Label)
cloudy	Hot	High	Weak	No
cloudy	Hot	Normal	Weak	Yes

Entropy (Hiking (label)) = - 1/2*$\log_2$ (1/2) -1/2*$\log_2$ (1/2) = 1

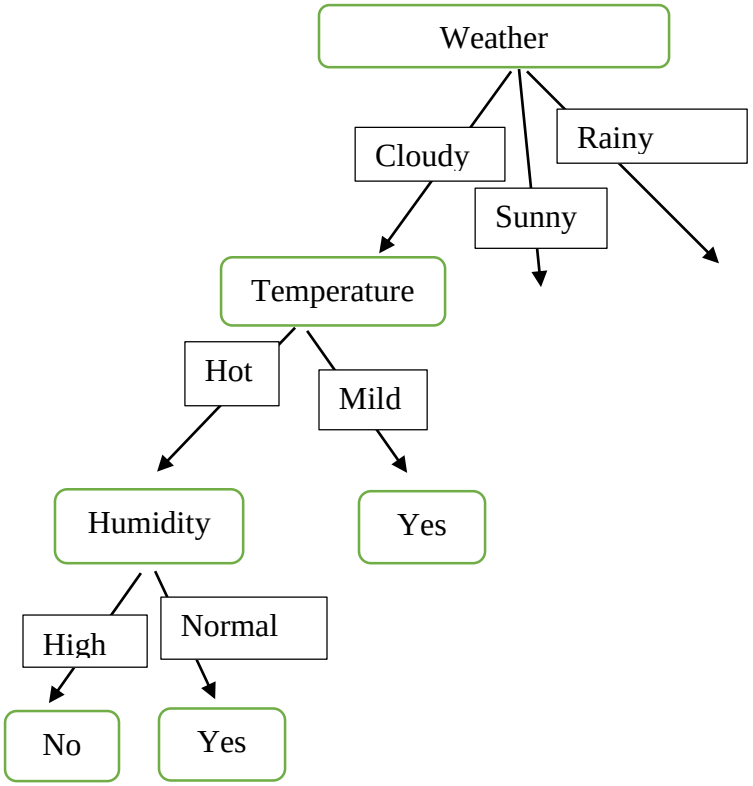
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 1-(1/2) (-1/1*$\log_2$ (1/1)) - (1/2) (-1/1*$\log_2$ (1/1)) = 1

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 1-(2/2) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) = 0

We obtain (Humidity) have the highest Information gain and we will select (Humidity). We will split the (Weather = Cloudy, Temperature=Hot) node by using the (Humidity) feature.



Calculation of Information gain for the (Sunny) branch of Weather.

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

Entropy (Hiking (label)) = - 1/3*$\log_2$ (1/3) -2/3*$\log_2$ (2/3) = 0.918

Calculation of Information gain for Temperature:

IG (Hiking (label), Temperature) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1)) = 0.251

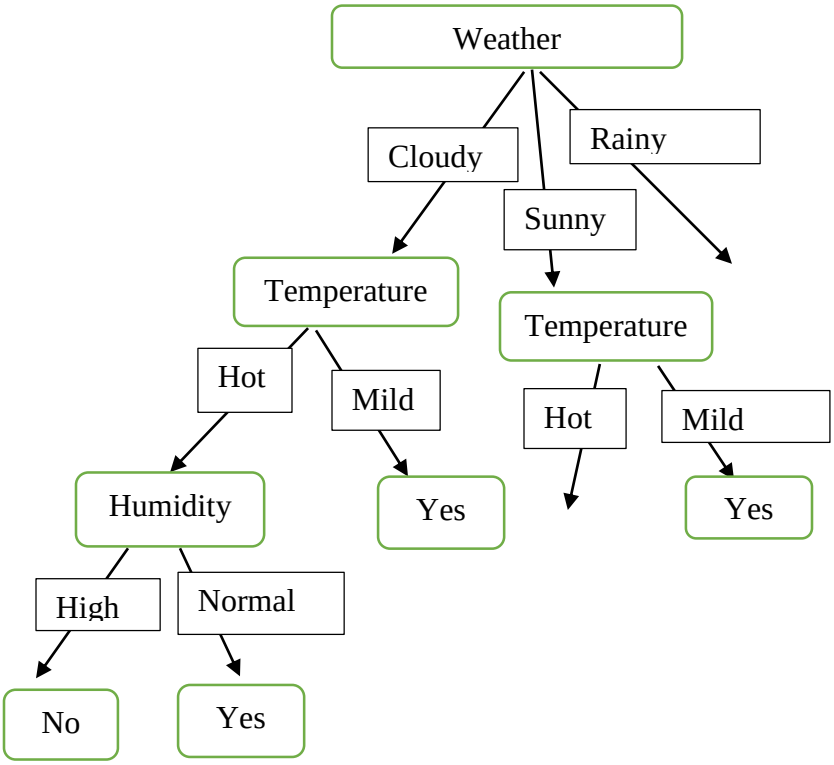
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1) = 0.251

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 0.918-(2/3) (-1/2*$\log_2$ (1/2)-1/2*$\log_2$ (1/2)) - (1/3) (-1/1*$\log_2$ (1/1) = 0.251

We obtain (Temperature, Humidity and wind) have the highest Information gain and we will select (Temperature). We will split the (Weather = Sunny) node by using the (Temperature) feature.



Calculation of Information gain for the (Hot) branch of Temperature.

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

Entropy (Hiking (label)) = - 1/2*$\log_2$ (1/2) -1/2*$\log_2$ (1/2) = 1

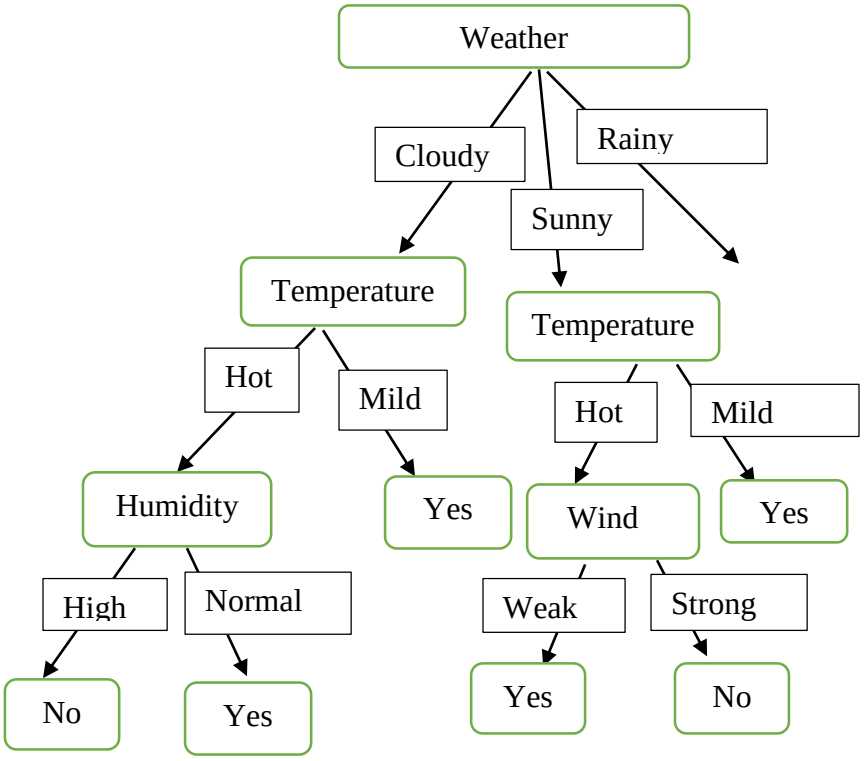
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 1-(2/2) (-1/2*$\log_2$ (1/2)-(1/2) *$\log_2$ (1/2)) = 0

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 1-(1/2) (-1/1*$\log_2$ (1/1))-(1/2) (-1/1*$\log_2$ (1/1)) = 1

We obtain (Wind) have the highest Information gain and we will select (Wind). We will split the (Weather = Sunny, Temperature=Hot) node by using the (Wind) feature.



Calculation of Information gain for the (Rainy) branch of Weather.

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

Entropy (Hiking (label)) = - 3/4*$\log_2$ (3/4) -1/4*$\log_2$ (1/4) = 0.811

Calculation of Information gain for Temperature:

IG (Hiking (label), Temperature) = 0.811-(3/4) (-2/3*$\log_2$ (2/3)-1/3*$\log_2$ (1/3)) - (1/4) (-1/1*$\log_2$ (1/1)) = 0.122

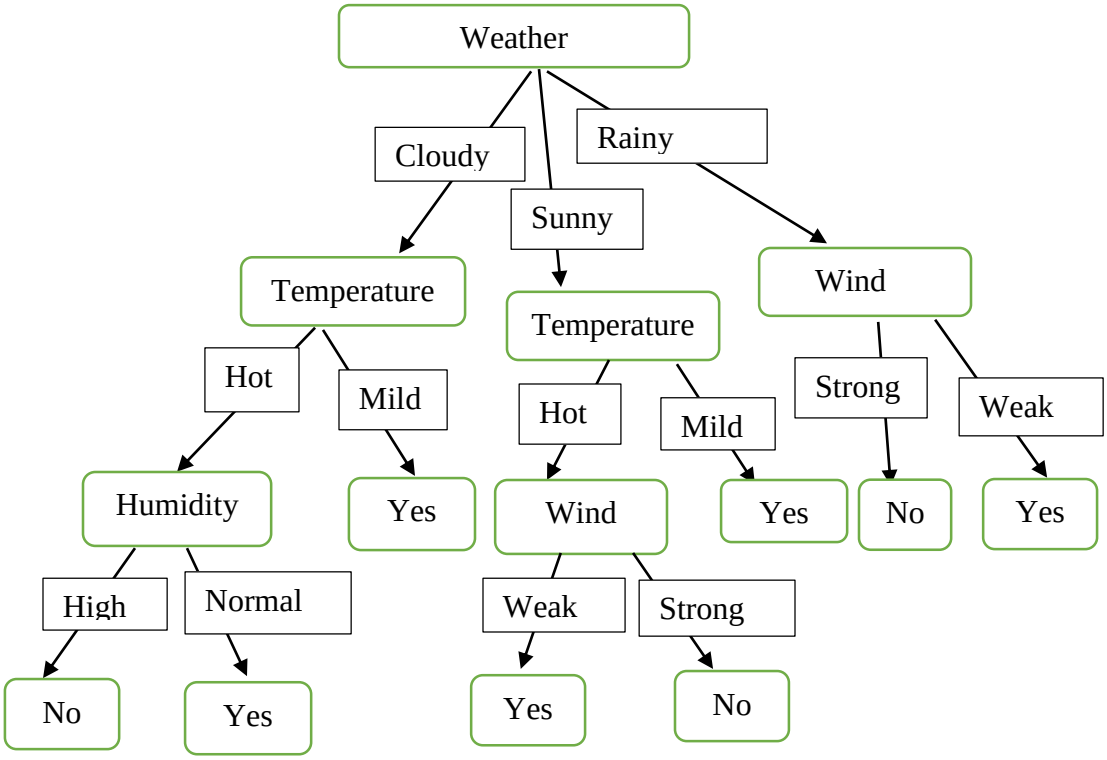
Calculation of Information gain for Humidity:

IG (Hiking (label), Humidity) = 0.811-(3/4) (-2/3*$\log_2$ (2/3)-1/3*$\log_2$ (1/3)) - (1/4) (-1/1*$\log_2$ (1/1)) = 0.122

Calculation of Information gain for Wind:

IG (Hiking (label), Wind) = 0.811-(3/4) (-3/3*$\log_2$ (3/3)) - (1/4) (-1/1*$\log_2$ (1/1) = 0.811

We obtain (wind) have the highest Information gain and we will select (wind). We will split the (Weather = Rainy) node by using the (Wind) feature.



In our example, we obtain the same graph in decision tree because of Gini Index and Information gain based on the different strategy.

1.3 Compare the advantages and disadvantages between Gini Index and Information Gain

Gini index and Information Gain are used for the analysis of the real-time scenario, and data is real that is captured from the real-time analysis.

In many definitions, it has also been mentioned as (impurity of data) or (how data is distributed). We can calculate which data is taking less Gini or more information gain for decision making in decision tree.

1. Gini index favours larger partitions (distributions) and is very easy to implement whereas information gain supports smaller partitions (distributions) with various distinct values, i.e. there is a need to perform an experiment with data and splitting criterion.

2. While working on categorical data variables, gini index gives results either in “success” or “failure” and performs binary splitting only, in contrast to this, information gain measures the entropy differences before and after splitting and depicts the impurity in class variables.
3. The gini index approach is used by CART algorithms, in opposite to that, information gain is deployed in ID3, C4.5 algorithms.
4. Gini index is measured by subtracting the sum of squared probabilities of each class from one, in opposite of it, information gain is obtained by multiplying the probability of the class by log (base= 2) of that class probability.

2. Part 2

2.1 Apply decision tree on the Circle Dataset, set criterion as gini and entropy.

Making Circle Dataset and plotting circle Dataset (training and testing) in 2D by function (plot Dataset).

```
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()
```

```
rs = 0
X, y = make_circles(300, noise=0.1, random_state=rs)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,random_state=rs)
plotDataset(X_train, y_train)
plotDataset(X_test, y_test)
```

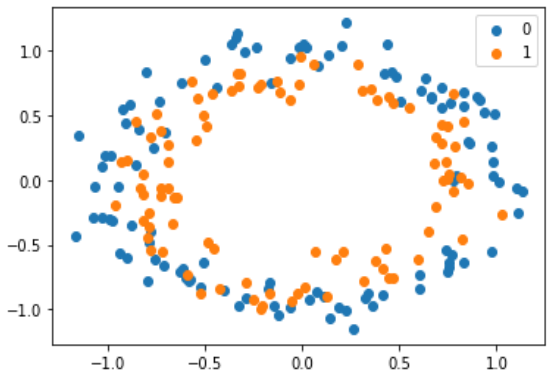


Figure 1: Training data

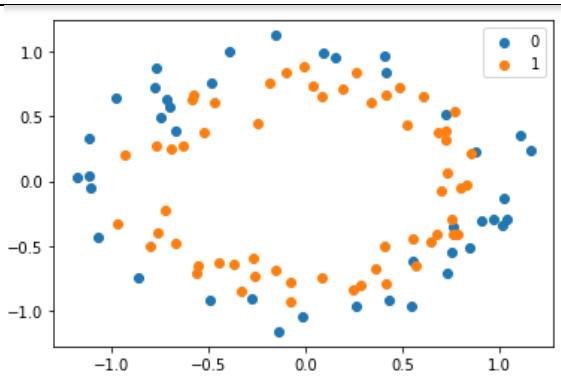


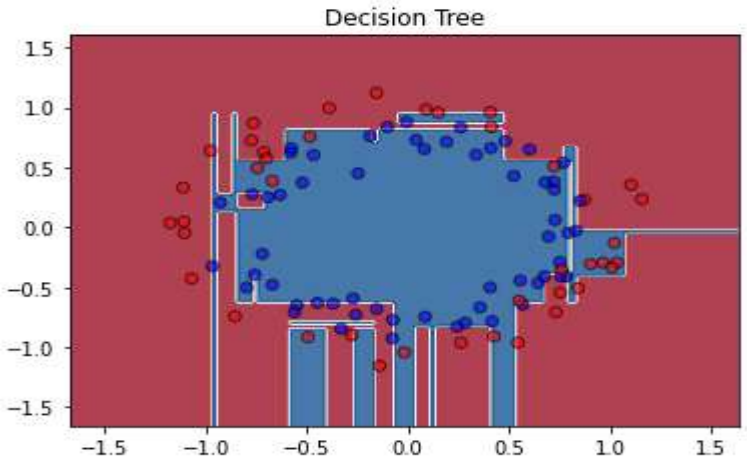
Figure 2: Testing data

Applying decision tree on the training data with two criterion (gini and entropy) and predict testing data to obtain accuracy and plotting decision boundaries by function (plotEstimator).

```
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()
```

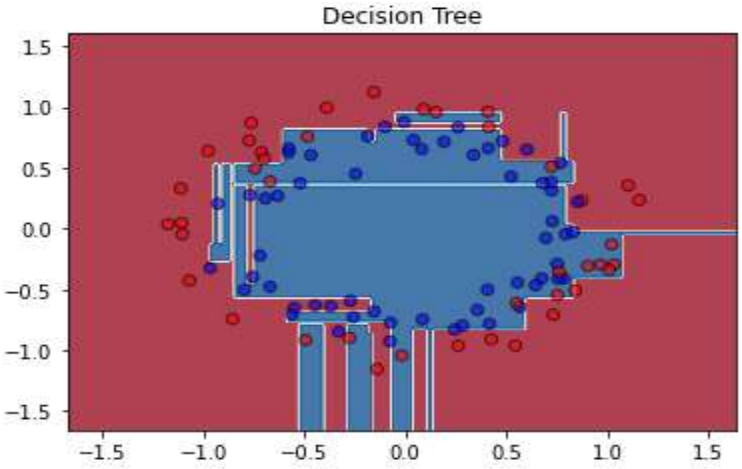
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
estimator = DecisionTreeClassifier(criterion="gini",random_state=rs)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
dtAccuracy = accuracy_score(y_test, y_pred)*100
print(f'Accuracy Score:{dtAccuracy:.1f}%')
plotEstimator(X_train, y_train, X_test, y_test, estimator, 'Decision Tree')
```

```
Accuracy Score:60.6%
```



```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
estimator = DecisionTreeClassifier(criterion="entropy",random_state=rs)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
dtAccuracy = accuracy_score(y_test, y_pred)*100
print(f'Accuracy Score:{dtAccuracy:.1f}%')
plotEstimator(X_train, y_train, X_test, y_test, estimator, 'Decision Tree')
```

```
Accuracy Score:66.7%
```



After applying decision tree with two criterion (gini and entropy). We obtain the accuracy of entropy is higher than the gini.

The difference between these criterions:

Gini index and Information Gain are used for the analysis of the real-time scenario, and data is real that is captured from the real-time analysis.

In many definitions, it has also been mentioned as (impurity of data) or (how data is distributed). We can calculate which data is taking less Gini or more information gain for decision making in decision tree.

1. Gini index favours larger partitions (distributions) and is very easy to implement whereas information gain supports smaller partitions (distributions) with various distinct values, i.e. there is a need to perform an experiment with data and splitting criterion.
2. While working on categorical data variables, gini index gives results either in “success” or “failure” and performs binary splitting only, in contrast to this, information gain

measures the entropy differences before and after splitting and depicts the impurity in class variables.

3. The gini index approach is used by CART algorithms, in opposite to that, information gain is deployed in ID3, C4.5 algorithms.
4. Gini index is measured by subtracting the sum of squared probabilities of each class from one, in opposite of it, information gain is obtained by multiplying the probability of the class by log (base= 2) of that class probability.

2.2 Plot Validation Accuracy (y-axis) vs Top K Important Feature (x-axis) curve; where 4-fold cross validation should be used, and also plot Test Accuracy vs Top K Important Feature curve.

Making classification Dataset and plotting classification Dataset (training and testing) in 2D by function (plot Dataset).

```
from sklearn.datasets import make_classification
rs = 0
X, y = make_classification(300, random_state=rs)
trX, teX, trY, teY = train_test_split(X, y, test_size=0.2,random_state=rs)
plotDataset(trX, trY)
plotDataset(teX,teY)
```

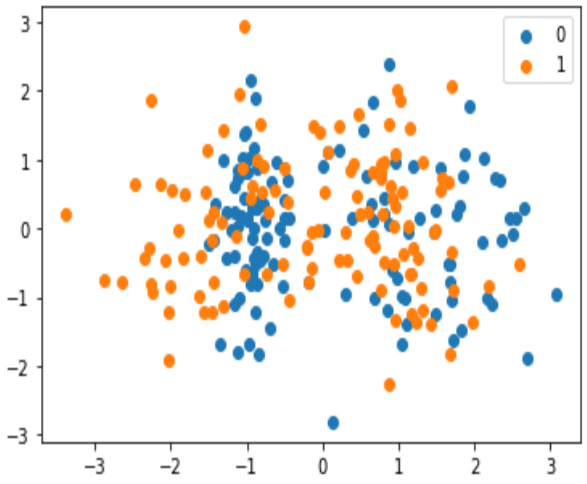


Figure 3Training data.

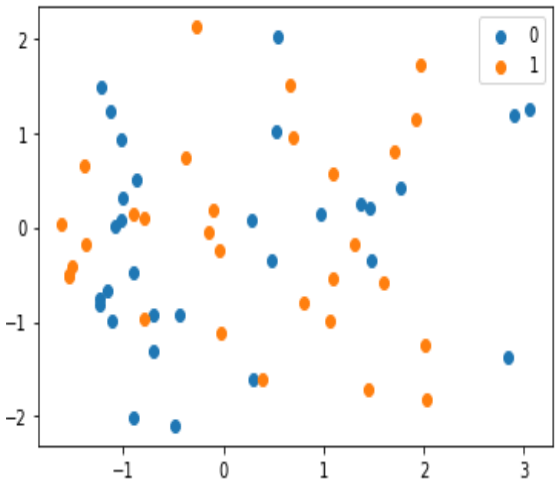


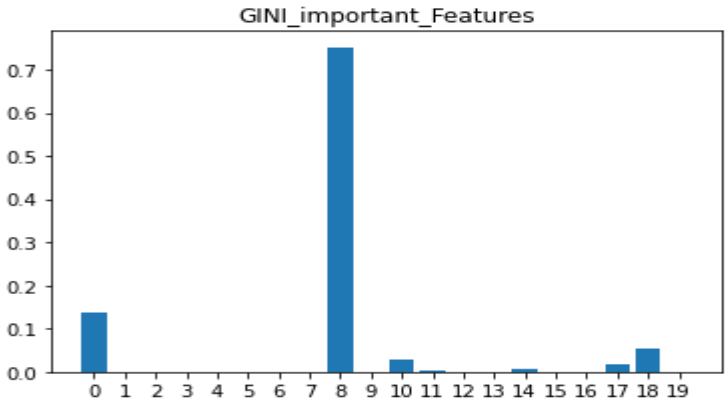
Figure 4Testing data.

Apply decision tree with gini to obtain the important features and plotting the all features to show the highest important of features by function (`plot_importance`) then sort the features based on index by descending.

```
estimator_class = DecisionTreeClassifier(criterion="gini",random_state=rs)
estimator_class.fit(trX, trY)
y_pred_class = estimator_class.predict(teX)
dtre_Accuracy = accuracy_score(teY, y_pred_class)
important_feature = estimator_class.feature_importances_
```

```
def plot_importance(importance, title):
    plt.bar([x for x in range(len(importance))], importance)
    plt.xticks([x for x in range(len(importance))])
    plt.title(title)
    plt.show()
```

```
plot_importance(important_feature, 'GINI_important_Features')
```



Obtain 7 features are very important (8, 0, 18, 10, 17, 14, 11).

```
sorted_importance_feature = np.argsort(important_feature)[::-1]
sorted_importance_feature
```

```
array([ 8,  0, 18, 10, 17, 14, 11,  6,  1,  2,  3,  4,  5, 19,  7, 12, 13,
       15, 16,  9])
```

```
imp = sorted_importance_feature[:7]
l1_im,l2_im,l3_im,l4_im,l5_im,l6_im,l7_im = imp[0:1],imp[0:2],imp[0:3],imp[0:4],imp[0:5],imp[0:6],imp
lst = [l1_im, l2_im, l3_im, l4_im, l5_im, l6_im, l7_im]
lst
```

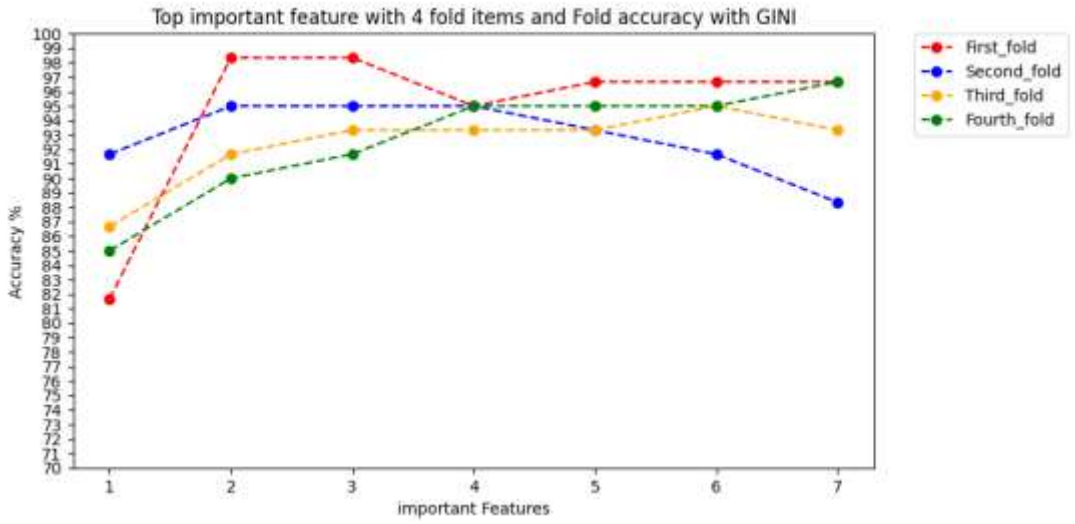
```
[array([8]),
 array([8, 0]),
 array([ 8,  0, 18]),
 array([ 8,  0, 18, 10]),
 array([ 8,  0, 18, 10, 17]),
 array([ 8,  0, 18, 10, 17, 14]),
 array([ 8,  0, 18, 10, 17, 14, 11])]
```

Applying cross validation (k=4) on the important features from classification dataset to plot Validation Accuracy with important features by function (plot_importance_accuracy) and fitting model decision tree on the important features from classification dataset to plot Test Accuracy with important features by function (plot_importance_testaccuracy).

```
from sklearn.model_selection import cross_validate
fold_accuracy = []
test_accuracy = []
estimator_tree = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
for i in lst:
    cv_results = cross_validate(estimator_tree, trX[:, i], trY, cv = 4)
    fold_accuracy.append(cv_results['test_score']*100)
    #fit the model to obtain the test accuracy
    estimator_tree.fit(trX[:, i], trY)
    y_preds = estimator_tree.predict(teX[:, i])
    accuracy =  accuracy_score(teY, y_preds)
    test_accuracy.append(accuracy* 100)
accuracies_fold = list(map(list, zip(*fold_accuracy)))
```

```
def plot_importance_accuracy(values, accuracies_values, title):
    plt.figure(figsize=(9,5), dpi=100)
    plt.plot(values, accuracies_values[0], color='red', marker='o',
                     linestyle='dashed', label = 'First_fold')
    plt.plot(values, accuracies_values[1], color='blue', marker='o',
                     linestyle='dashed', label = 'Second_fold')
    plt.plot(values, accuracies_values[2], color='orange', marker='o',
                     linestyle='dashed', label = 'Third_fold')
    plt.plot(values, accuracies_values[3], color='green', marker='o',
                     linestyle='dashed', label = 'Fourth_fold')
    plt.title(title)
    plt.xlabel('important Features')
    plt.ylabel('Accuracy %')
    plt.xticks(values)
    plt.yticks(range(70,101))
    plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
    plt.show
```

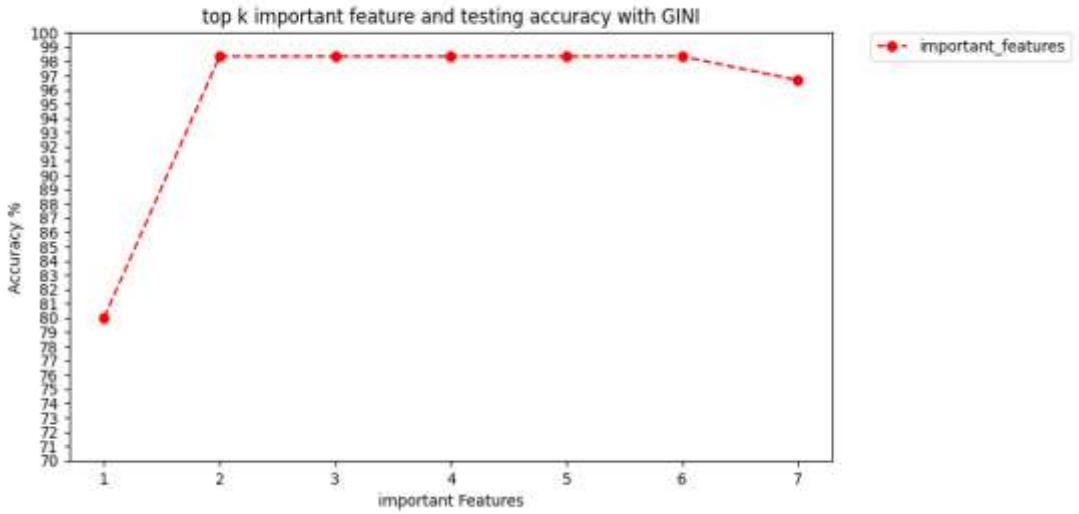
```
plot_importance_accuracy(range(1,8), accuracies_fold, "Top important feature with 4 fold items and Fold accuracy with GINI")
```



After applying cross validation (k=4) on the important features. We obtain the first fold is the highest accuracy (98%) with number of features =2, 3.

```
def plot_importance_testaccuracy(values, test_accuracy_values, title):
    plt.figure(figsize=(9,5), dpi=100)
    plt.plot(values, test_accuracy_values, color='red', marker='o',
                     linestyle='dashed', label = 'important_features')
    plt.title(title)
    plt.xlabel('important Features')
    plt.ylabel('Accuracy %')
    plt.xticks(values)
    plt.yticks(range(70,101))
    plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
    plt.show
```

```
plot_importance_testaccuracy(range(1,8), test_accuracy, "top k important feature and testing accuracy with GINI")
```

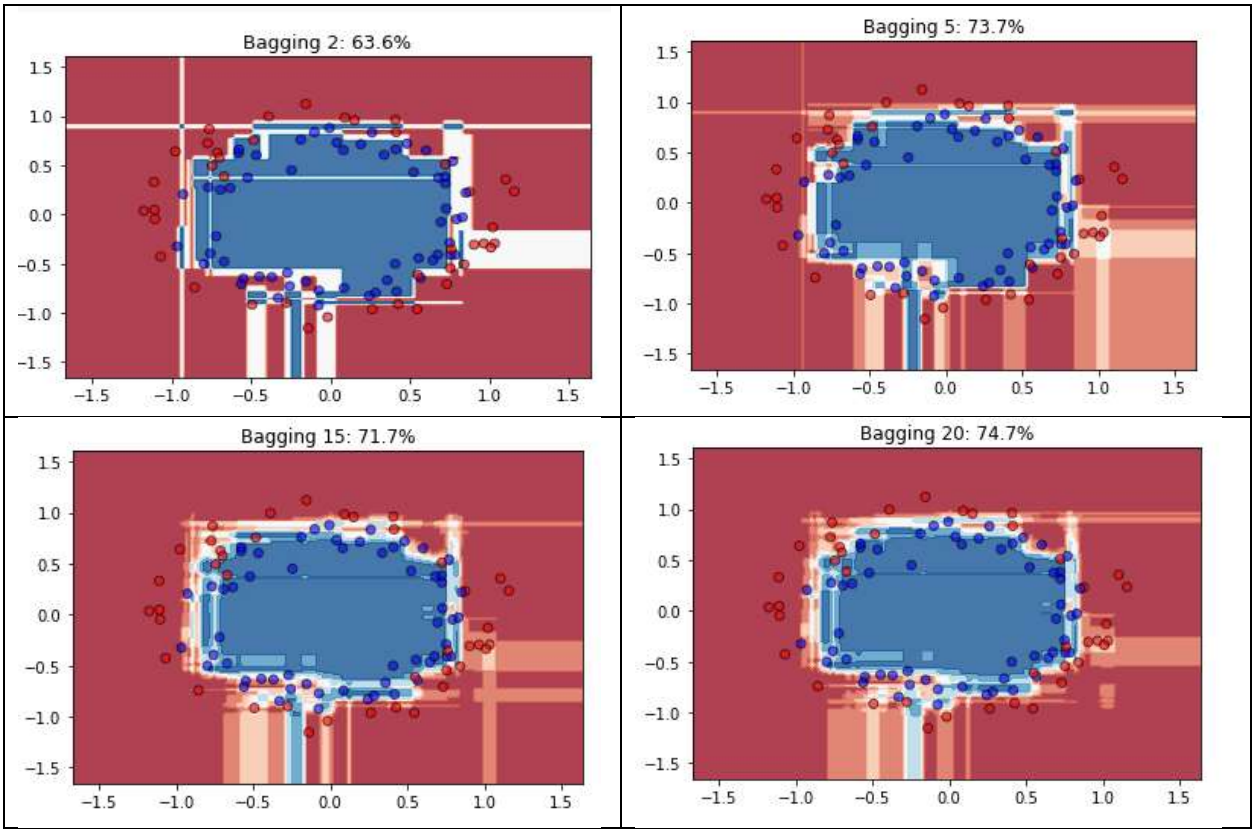


After fitting model (decision tree) and plotting accuracy with the number of importance features. We obtain the highest accuracy (98%) with number of features = 2, 3, 4, 5, 6

2.3 Use Circle Dataset. Set the number of estimators as 2, 5, 15, 20 respectively, and generate the results accordingly (i.e., accuracy and decision boundary)

```
from sklearn.ensemble import BaggingClassifier

for nEst in [2,5,15,20]:
    estimator = BaggingClassifier(n_estimators=nEst, random_state=rs)
    score = estimator.fit(X_train, y_train).score(X_test, y_test)*100
    plotEstimator(X_train, y_train, X_test, y_test, estimator, f'Bagging {nEst}: {score:.1f}%')
```



2.4 Explain why bagging can reduce the variance and mitigate the overfitting problem

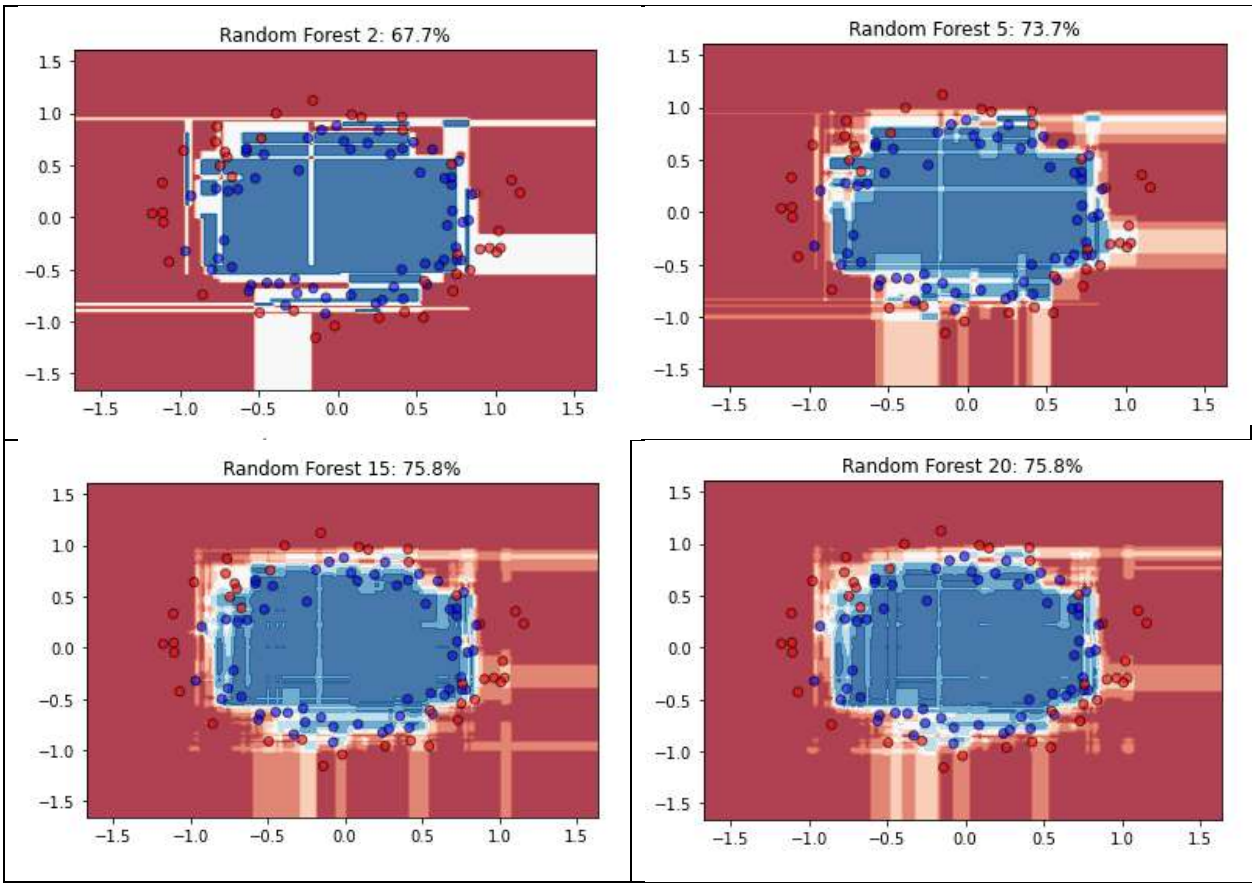
Bootstrap aggregation, or "Bagging," in machine learning decreases variance through building more advanced models of complex data sets. Specifically, the bagging approach creates subsets which are often overlapping to model the data in a more involved way. One interesting and straightforward notion of how to apply bagging is to take a set of random samples and extract the simple mean. Then, using the same set of samples, create dozens of subsets built as decision trees to manipulate the eventual results. The second mean should show a truer picture of how those individual samples relate to each other in terms of value. The same idea can be applied to

any property of any set of data points. Since this approach consolidates discovery into more defined boundaries, it decreases variance and helps with overfitting.

2.5 Use Circle Dataset. Set the number of estimators as 2, 5, 15, 20 respectively, and generate the results accordingly (i.e., accuracy and decision boundary)

```
from sklearn.ensemble import RandomForestClassifier

for nEst in [2,5,15,20]:
    estimator = RandomForestClassifier(n_estimators=nEst, random_state=rs)
    score = estimator.fit(X_train, y_train).score(X_test, y_test)*100
    plotEstimator(X_train, y_train, X_test, y_test, estimator, f'Random Forest {nEst}: {score:.1f}%')
```



2.6 Compare with bagging results and explain the difference between Bagging and Random Forest

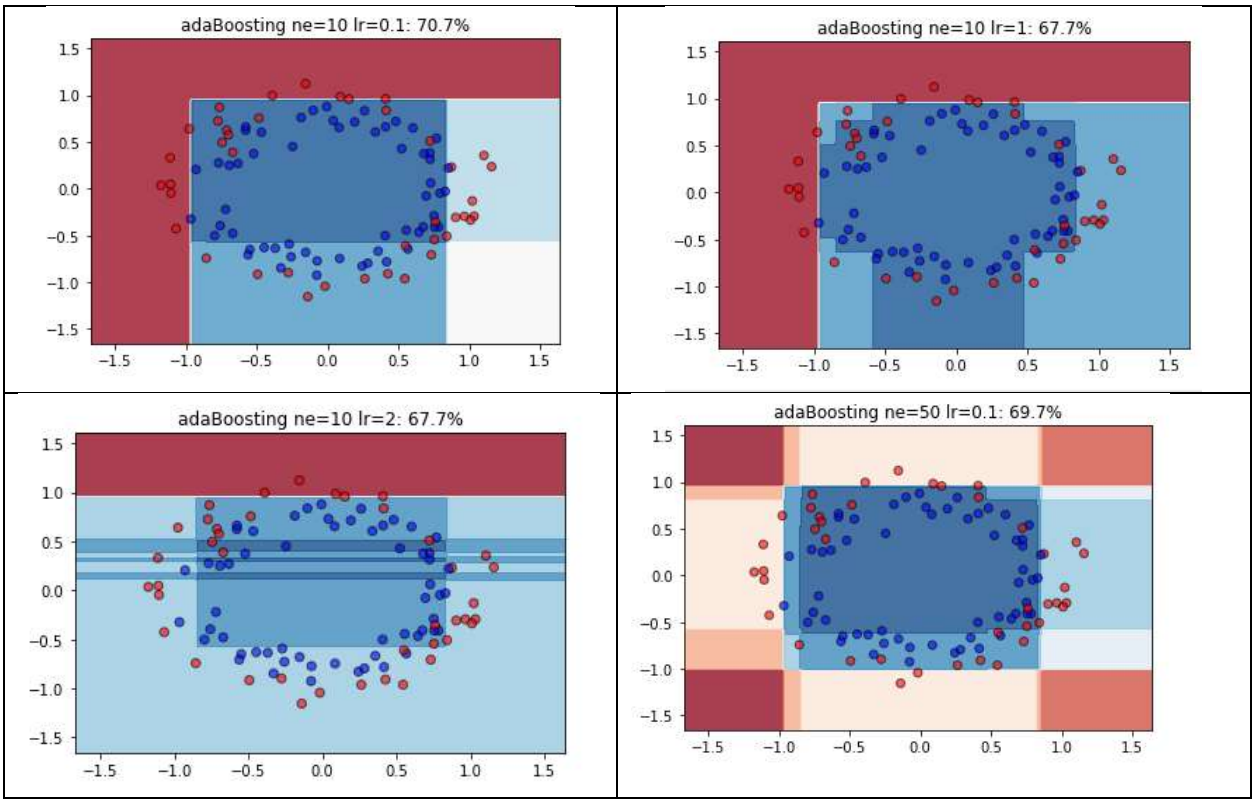
After we apply Bagging classifier with multiple numbers of estimators, we found that the highest accuracy equal 74.7 % with number of estimator equal 20. In Random Forest we get the highest accuracy score with number of estimators 15 and 20 but we choose the lowest number of estimators that gives us the highest accuracy which is 15.

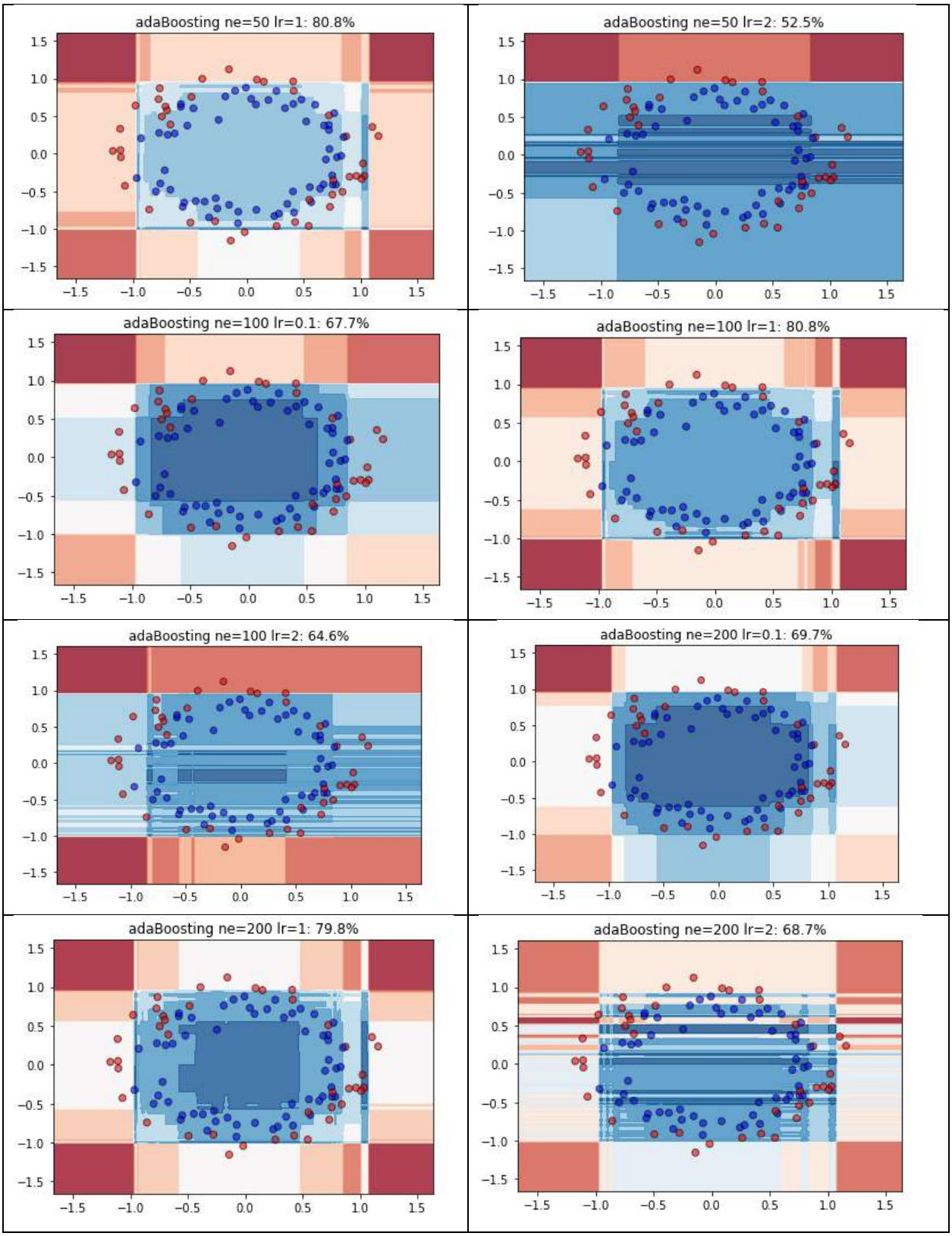
Random forests differ from bagged trees by forcing the tree to use only a subset of its available predictors to split on in the growing phase. All the decision trees that make up a random forest are different because each tree is built on a different random subset of data. Because it minimizes overfitting, it tends to be more accurate than a single decision tree. Bagging can be used with any classifier but Random forest depend on decision tree.

2.7 Use Circle Dataset. 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.

```
from sklearn.ensemble import AdaBoostClassifier

for estimator in [10,50,100,200]:
    for learning_rate in [0.1,1,2]:
        classifier = AdaBoostClassifier(n_estimators=estimator,
                                        learning_rate=learning_rate,
                                        random_state=rs)
        score = classifier.fit(X_train, y_train).score(X_test, y_test)*100
        plotEstimator(X_train, y_train, X_test, y_test, classifier,
                      f'adaBoosting ne={estimator} lr={learning_rate}: {score:.1f}%')
```




adaBoosting ne=50 lr=1: 80.8%
adaBoosting ne=50 lr=2: 52.5%
adaBoosting ne=100 lr=0.1: 67.7%
adaBoosting ne=100 lr=1: 80.8%
adaBoosting ne=100 lr=2: 64.6%
adaBoosting ne=200 lr=0.1: 69.7%
adaBoosting ne=200 lr=1: 79.8%
adaBoosting ne=200 lr=2: 68.7%

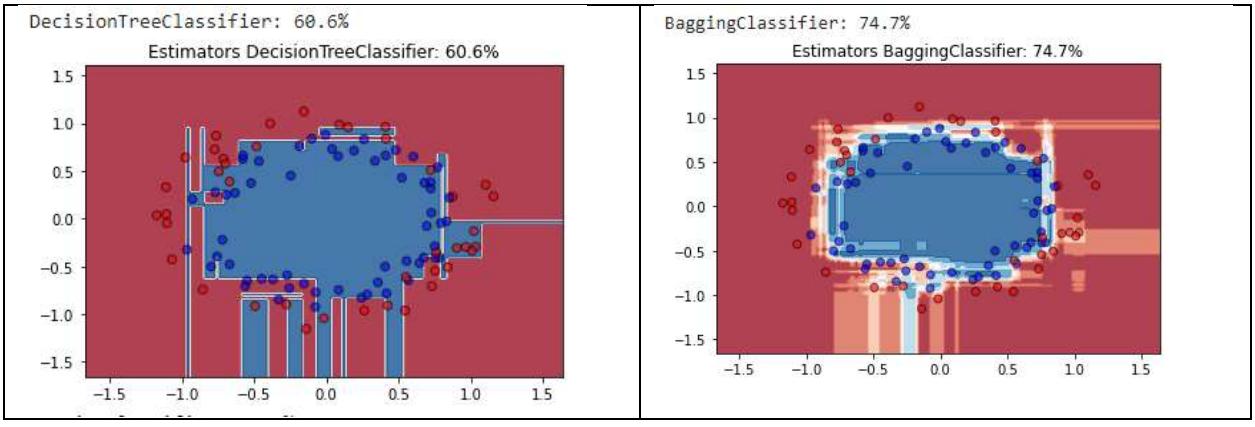
After applying hyperparameters on learning rate and number of estimators in AdaBoost, we obtain n_estimaotrs = 50 and =100 with learning _rate = 1 have the highest accuracy but we choose the lowest number of estimators with learning rate that gives us the highest accuracy which is (50,1).

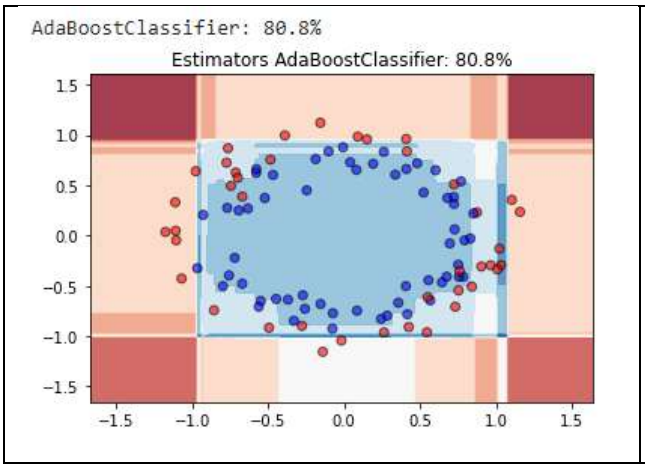
2.8 We have tuned the Decision Tree, Bagging, Random Forest, and AdaBoost in the previous section. Use these fine-tuned model as base estimators and use Naive Bayes, Logistic Regression, and Decision Tree as aggregators to generate the results accordingly (i.e., accuracy and decision boundary)

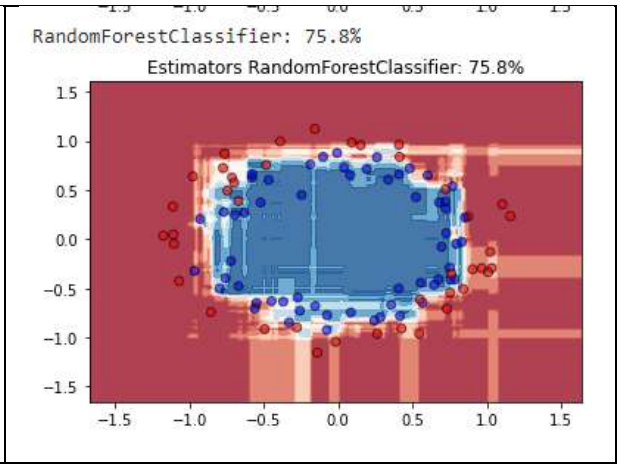
```
rs = 0

estimators = [DecisionTreeClassifier(random_state=rs),
              BaggingClassifier(n_estimators=20, random_state=rs),
              AdaBoostClassifier(n_estimators=50,learning_rate=1,random_state=rs),
              RandomForestClassifier(n_estimators=15, random_state=rs)]
estimators = {estimator.__class__.__name__: estimator for estimator in estimators}

print(f'Baselines Score:')
for estName, est in estimators.items():
    est = clone(est).fit(X_train, y_train)
    predY = est.predict(X_test)
    acc = accuracy_score(y_test, predY)*100
    print(f'{estName}: {acc:.1f}%')
    plotEstimator(X_train, y_train, X_test, y_test, est, f'Estimators {estName}: {acc:.1f}%')
```



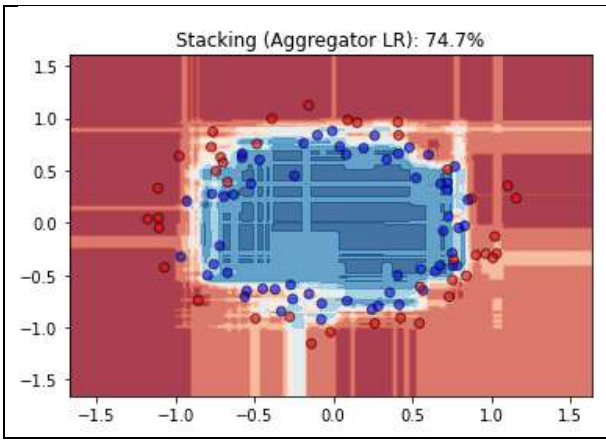


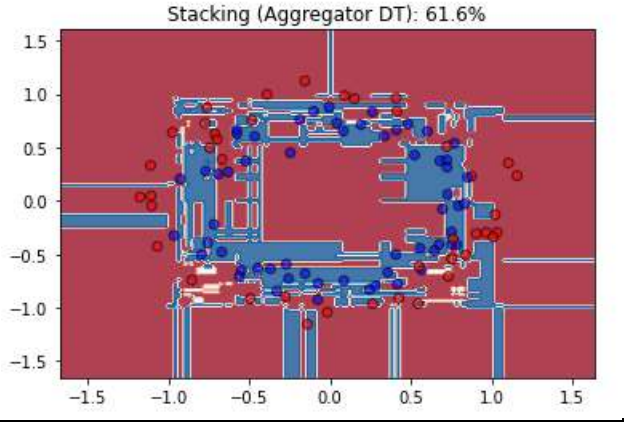


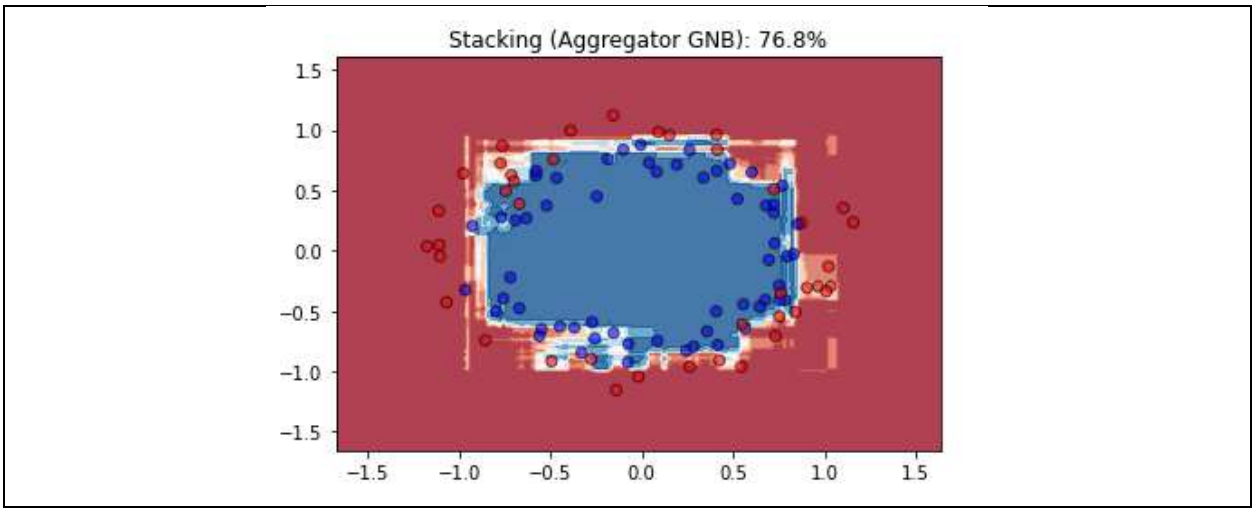

```
stacking = StackingClassifier([(estName, clone(est)) for estName, est in estimators.items()], LogisticRegression(random_state=rs), n_jobs=-1)
stacking.fit(X_train, y_train)
predY = stacking.predict(X_test)
acc = accuracy_score(y_test, predY)*100
plotEstimator(X_train, y_train, X_test, y_test, stacking, f'Stacking (Aggregator LR): {acc:.1f}%')

stacking = StackingClassifier([(estName, clone(est)) for estName, est in estimators.items()], DecisionTreeClassifier(random_state=rs), n_jobs=-1)
stacking.fit(X_train, y_train)
predY = stacking.predict(X_test)
acc = accuracy_score(y_test, predY)*100
plotEstimator(X_train, y_train, X_test, y_test, stacking,f'Stacking (Aggregator DT): {acc:.1f}%')

stacking = StackingClassifier([(estName, clone(est)) for estName, est in estimators.items()], GaussianNB(), n_jobs=-1)
stacking.fit(X_train, y_train)
predY = stacking.predict(X_test)
acc = accuracy_score(y_test, predY)*100
plotEstimator(X_train, y_train, X_test, y_test, stacking,f'Stacking (Aggregator GNB): {acc:.1f}%')
```







3. Conclusion

we learn from this assignment, making circles and classification Dataset, after we applied decision tree with entropy and Gini, we obtain that the entropy in decision tree has higher accuracy than the Gini. Then when we applied Bagging on circles dataset with different number of estimators, we get the highest accuracy when number of estimators = 20. After applying Random Forest on circles dataset with different number of estimators, we get higher accuracy when number of estimators = 15 and =20 and we choose the lowest number of estimators which is 15. We applied AdaBoost on circles dataset with different number of estimators and learning rate, we get the highest accuracy when number of estimators = 50 with learning rate = 1. Finally, we get highest accuracy when we applied Stacking by base estimator aggregation by Naive Bayes.