Decision Tree Analysis

Assignment 4_G4

Ahmed Yousry **Bassel Hamshary**
Mohamed El-Namoury

uOttawa

Faculté de génie
Faculty of Engineering

JULY 13, 2021
UNIVERSITY OF OTTAWA
Ottawa, Canada

Table of Contents

List of Figures

1. Implementation

1.1. Part one (Numerical)

1.1.1. Gini Index

Weather	Temperature	Humidity	Wind
Cloudy = $1 – (2/3)^2 – (1/3)^2$ = 0.444 Sunny = $1 – (2/3)^2 – (1/3)^2$ = 0.444 Rainy = $1 – (1/4)^2 – (3/4)^2$ = 0.375	Hot = $1 – (1/2)^2 – (1/2)^2$ = 0.5 Mild = $1 – (3/5)^2 – (2/5)^2$ = 0.48 Cool = $1 – (0)^2 – (1)^2 = 0$	High = $1 – (3/7)^2 – (4/7)^2$ = 0.4898 Normal = $1 – (2/3)^2 – (1/3)^2$ = 0.444	Weak = $1 – (3/4)^2 – (1/4)^2$ = 0.375 Strong = $1 – (2/6)^2 – (4/6)^2$ = 0.444

After getting the Gini number for each attribute in features, we should calculate the Gini for the whole feature:

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

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

Humidity: 0.4898(7/10) + 0.444(3/10) = 0.476 **Wind**: 0.375(4/10) + 0.444(6/10) = 0.4164

At this moment we have 2 features with same Gini index number (“Weather”, “Wind”). So, we can choose any one of the it will make no different. We will start with weather. After choosing weather we should use “Rainy” attribute because it has the least Gini number.

Rainy	Mild	High	Strong	No
Rainy	Cool	Normal	Strong	No
Rainy	Mild	High	Weak	Yes
Rainy	Mild	High	Strong	No

After this repeat the pervious steps. But this time on the rows that contain “Rainy Only”.

Temperature	Humidity	Wind
Hot = $1 – (0)^2 – (0)^2 = 1$ Mild = $1 – (1/3)^2 – (2/3)^2 = 0.444$ Cool = $1 – (0)^2 – (1)^2 = 0$	High = $1 – (1/3)^2 – (2/3)^2 = 0.444$ Normal = $1 – (0)^2 – (1)^2 = 0$	Weak = $1 – (1)^2 – (0)^2 = 0$ Strong = $1 – (0)^2 – (1)^2 = 0$

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

Wind: 0(1/4) + 0(3/4) = 0

Wind has zero index. This mean it is a child for the weather when its “Rainy”. Now we must decide which feature will be a child when it is “Not Rainy”. So, we will select the other rows for our table that do not contain “Rainy” In Weather feature. And repeat the pervious steps to get decision when its “Rainy”.

Temperature	Humidity	Wind
Gini Index for Hot = $1 – (3/4)^2 – (1/4)^2 = 0.375$ Gini Index for Mild = $1 – (1)^2 – (0)^2 = 0$ Gini Index for Cool = $1 – (0)^2 – (0)^2 = 1$	Gini Index for High = $1 – (2/4)^2 – (2/4)^2 = 0.5$ Gini Index for Normal = $1 – (1)^2 – (0)^2 = 0$	Gini Index for Weak = $1 – (2/3)^2 – (1/3)^2 = 0.444$ Gini Index for Strong = $1 – (2/3)^2 – (1/3)^2 = 0.444$

Temperature: 0.375(4/6) + 0(2/6) + 0(0) = 0.25 **Humidity**: 0.5(4/6) + 0(2/6) = 0.333

Wind: 0(1/4) + 0(3/4) = 0.444(3/6) + 0.444(3/6) = 0.444

Least Gini index is "Temperature". This means after checking it is "Not Rainy" the tree will go for "Temperature" to check.

Now we are done from "Weather" lets repeat the pervious steps with "Temperature". In "Temperature" we see Mild as minimum Gini index but we cannot choose it because the "Mild" always classified as "Yes" so we cannot choose it because the tree will be overfitted so we will choose "Hot" because it has the lowest Gini index after "Mild".

Weather	Temperature	Humidity	Wind	Hiking
Cloudy	Hot	High	Weak	No
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No
Cloudy	Hot	Normal	Weak	Yes

Now calculate the Gini index as pervious

Weather	Humidity	Wind
Cloudy = $1 – (1/2)^2 – (1/2)^2 = 0.5$ Sunny = $1 – (1/2)^2 – (1/2)^2 = 0.5$	High = $1 – (1/3)^2 – (2/3)^2 = 0.444$ Normal = $1 – (1)^2 – (0)^2 = 0$	Gini Index for Weak = $1 – (2/3)^2 – (1/3)^2 = 0.444$ Gini Index for Strong = $1 – (0)^2 – (1)^2 = 0$

Weather: 0.5(0.5) + 0.5(0.5) = 0.5 Humidity: 0.444(3/4) + 0(1/4) = 0.333

Wind: 0.444(3/4) + 0(1/4) = 0.333

The same situation we saw before. We have 2 similar Gini index. So, we can choose any one of them it will make no different in this case. We will go throw "Humidity" in this case with attribute high. hence "Normal" has lower Gini index but don't have a lot of test cases on normal so we will choose "High".

Cloudy	Hot	High	Weak	No
Sunny	Hot	High	Weak	Yes
Sunny	Hot	High	Strong	No

Weather	Wind
Cloudy = $1 - (0)^2 - (1)^2 = 0$ Sunny = $1 - (1/2)^2 - (1/2)^2 = 0.5$	Weak = $1 - (1/2)^2 - (1/2)^2 = 0.5$ Strong = $1 - (0)^2 - (1)^2 = 0$

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

Same Gini index makes no different between them. At the end, the final 2 branches will go for "Weather" then "Wind"

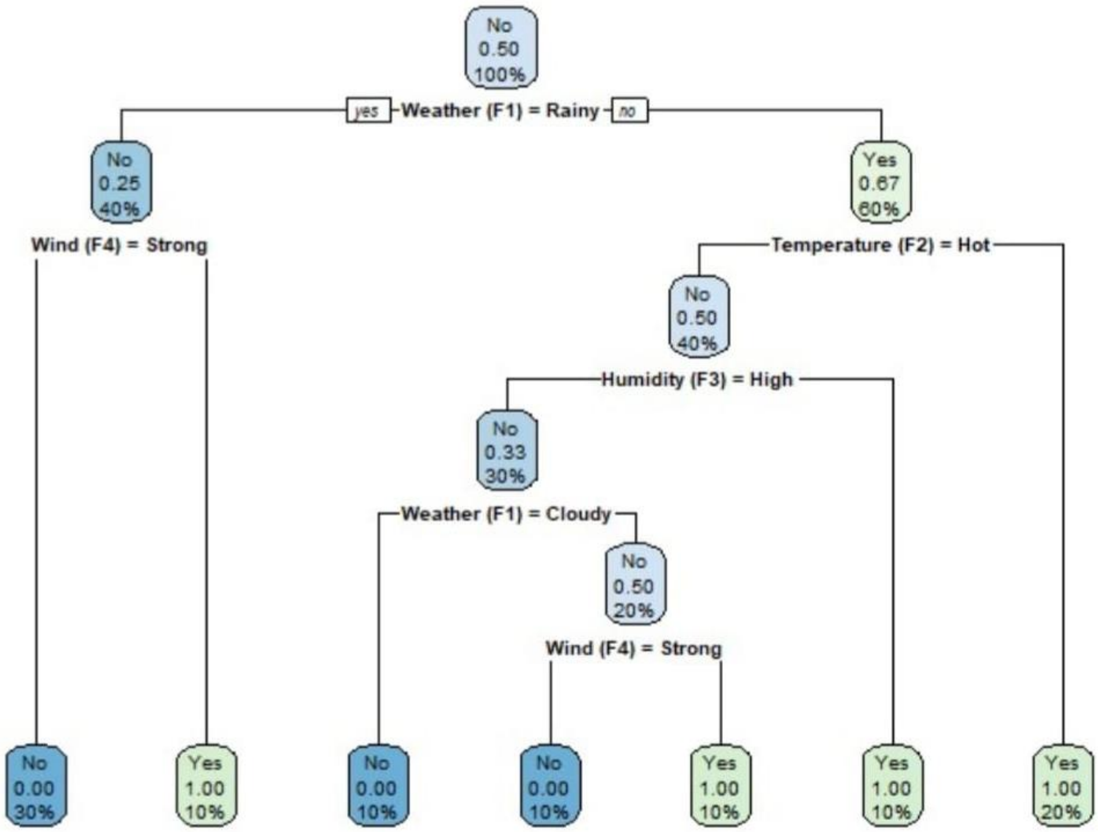


Figure 1: Gini Index Decision Tree

1.1.2. Information Gain

Information Gain
Decision Tree

$$Entropy(S) = -\frac{5}{10} \log_2 \frac{5}{10} - \frac{5}{10} \log_2 \frac{5}{10} = \boxed{1}$$

$$Gain(S, Weather) = 1 - \frac{S_{cloudy}}{10} E(S_{cloudy}) - \frac{S_{sunny}}{10} E(S_{sunny}) - \frac{S_{rainy}}{10} E(S_{rainy})$$

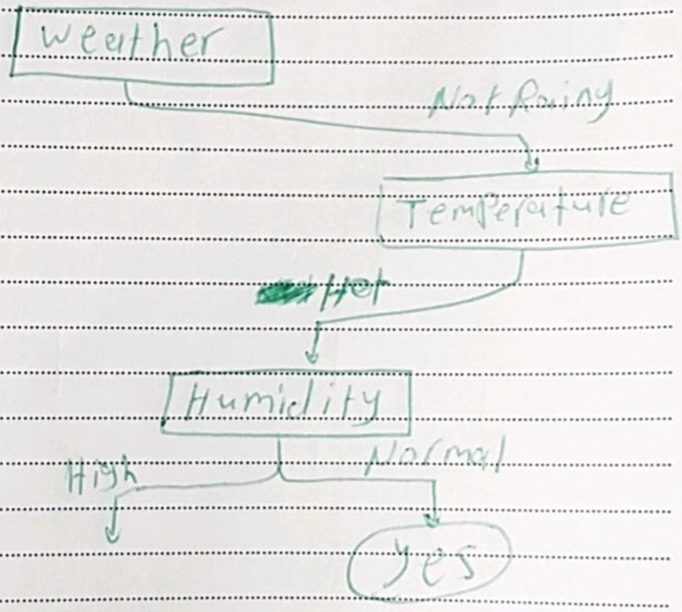
$$= 1 - \frac{3}{10}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{3}{10}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{4}{10}\left(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right) = \boxed{0.124}$$

$$Gain(S, \cancel{Weath}\ temp) = 1 - \frac{4}{10}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{5}{10}\left(\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) - \frac{1}{10}\left(\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.115}$$

$$Gain(S, Hum.) = 1 - \frac{7}{10}\left(-\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7}\right) - \frac{3}{10}\left(\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) = \boxed{0.031}$$

$$\text{Gain}(S, \text{NotRainy}, \text{Hot}, \text{Hum.}) = 1 - \frac{3}{4}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.31}$$

$$\text{Gain}(S, \text{NotRainy}, \text{Hot}, \text{Wind}) = 1 - \frac{3}{4}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.31}$$



$$E(S, \text{Not Rainy}, \text{Hot}, \text{High H.}) = E(2,1) = \boxed{0.92}$$

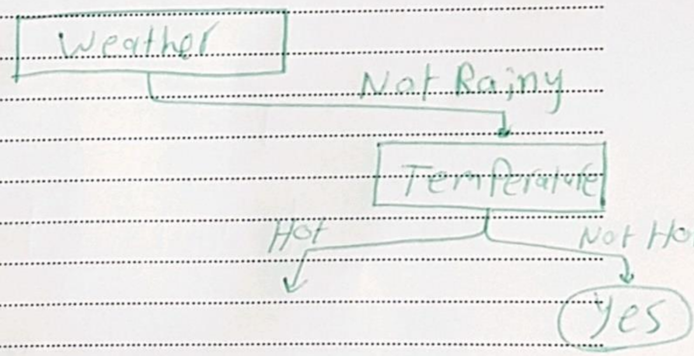
If Not Rainy

$$E(S, NotRainy) = E(2,4) = \boxed{0.92}$$

$$Gain(S, NotRainy, temp.) = 0.92 - \frac{4}{6}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{2}{6}\left(\frac{2}{2}\log_2\frac{2}{2}\right) = \boxed{0.253}$$

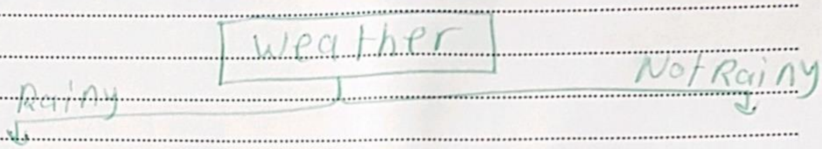
$$Gain(S, NotRainy, Hum.) = 0.92 - \frac{4}{6}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{2}{6}\left(\frac{2}{2}\log_2\frac{2}{2}\right) = \boxed{0.253}$$

$$Gain(S, NotRainy, wind) = 0.92 - \frac{3}{6}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{3}{6}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) = \boxed{0}$$



$$E(S, NotRainy, Hot) = E(2,2) = \boxed{1}$$

$$Gain(S, wind) = 1 - \frac{4}{10}\left(\frac{-1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right) - \frac{6}{10}\left(\frac{-4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right) = \boxed{0.124}$$



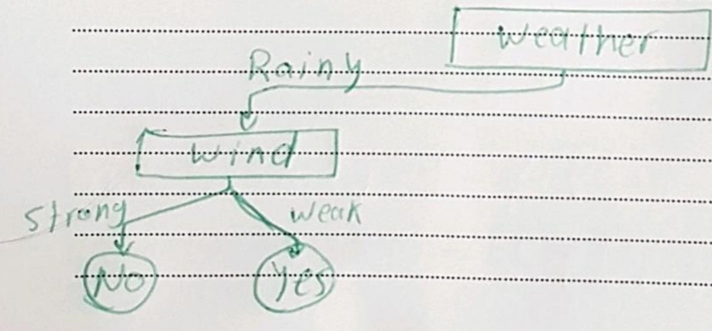
If Raining

$$Entropy(S, weather(Rainy)) = E(1,3) = \boxed{0.81}$$

$$Gain(S, Rainy, temp.) = 0.81 - \frac{3}{4}\left[\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right] - \frac{1}{4}\left(-\frac{1}{1}\log\frac{1}{1}\right) = \boxed{0.121}$$

$$Gain(S, Rainy, Hum.) = 0.81 - \frac{3}{4}\left(\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) - \frac{1}{4}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.121}$$

$$Gain(S, Rainy, wind) = 0.81 - \frac{3}{4}\left(\frac{-3}{3}\log_2\frac{3}{3}\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.81}$$



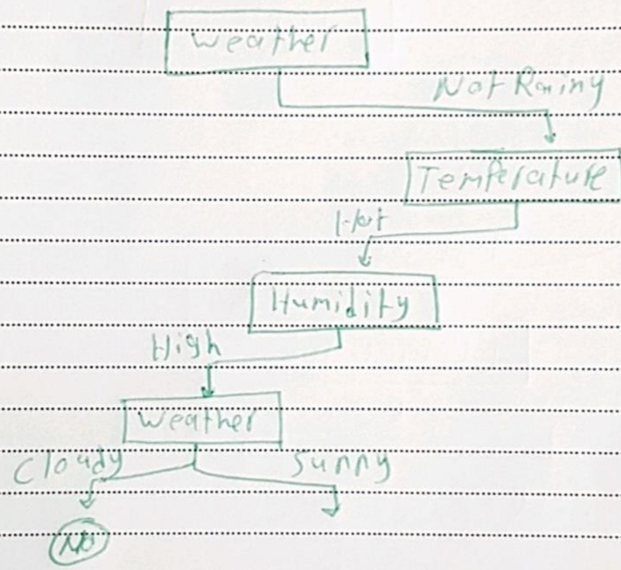
sunny, cloudy

Gain(S, Not Rainy, Hot, High H., Weather) =

$$0.92 - \frac{2}{3}\left(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{1}{3}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.253}$$

Gain(S, Not Rainy, Hot, High H., wind) =

$$0.92 - \frac{2}{3}\left(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{1}{3}\left(-\frac{1}{1}\log_2\frac{1}{1}\right) = \boxed{0.253}$$



E(S, Not Rainy, Hot, High H., Sunny) = E(1,1) = 1

G(S, Not Rainy, Hot, High H., Sunny, wind) = $\boxed{1}$

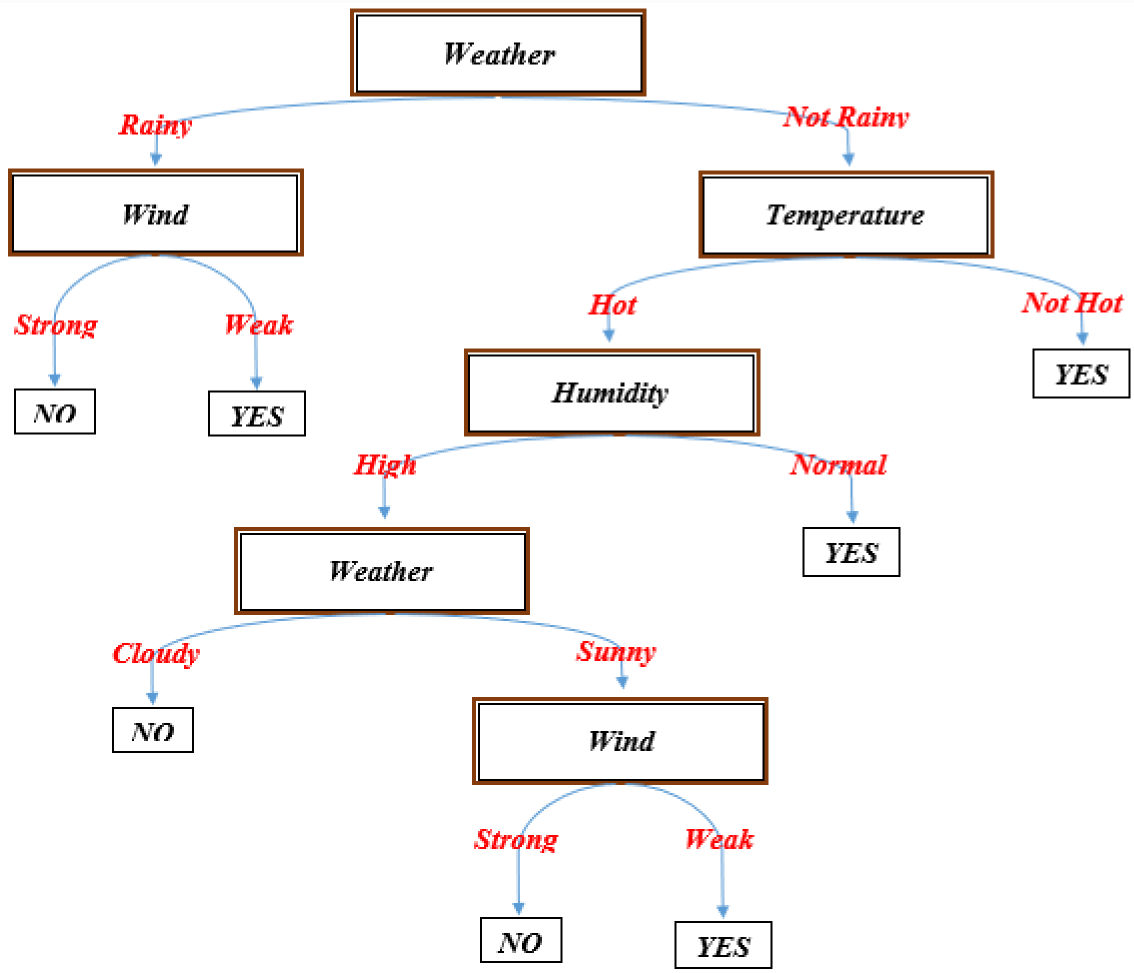
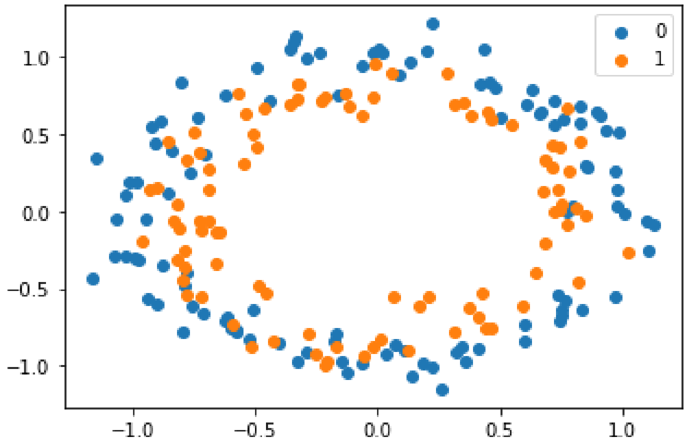
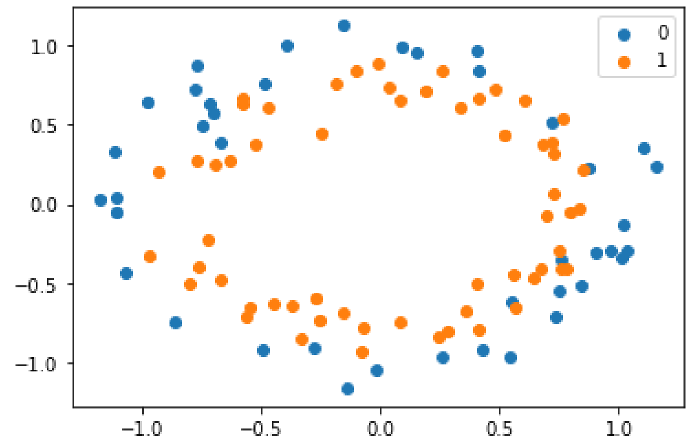


Figure 2: Information Gain Decision Tree

1.1.3. Comparison between Gini Index & Information Gain

	Advantages	Dis-Advantages
Gini Index	• Facilitates the bigger distributions so easy to implement • It deals with inequality. So, it can judge the distribution pattern better	• Operates on the categorical target variables in terms of "success" or "failure" and performs only binary split • Gini index can be dependent on sample size • The Gini index is sometimes prone to random and systematic data errors. If there is any inaccurate data, it can create problems with the index value • Gini index value can be the same for different distributions. So, it creates degeneracy
Information Gain	• Computes the difference between entropy before and after the split and indicates the impurity in classes of elements • leaves with a small number of instances are assigned less weight and it favors dividing data into bigger but homogeneous groups.	• Favors lesser distributions having small count with multiple specific values • Accuracy is usually problematic with unbalanced data • One of the drawbacks is that it tends to use the feature that has more unique values.

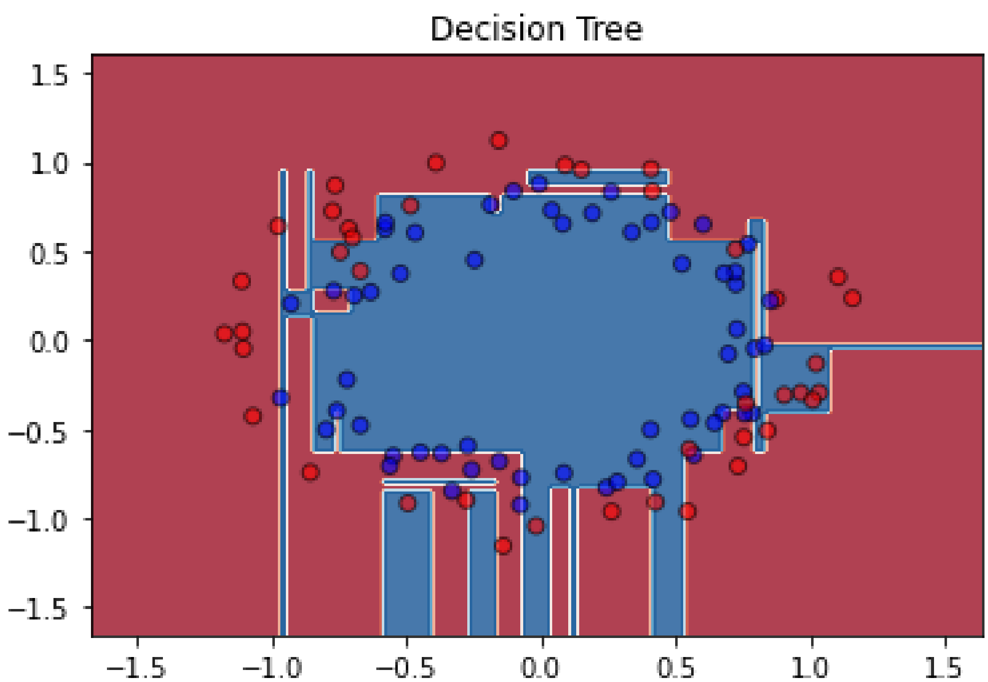
1.2. Part Two (Programming)

Training set	Testing set
	

1.2.1. Decision Tree

```
estimator = DecisionTreeClassifier(random_state=rs)
estimator.fit(X_train, y_train)
y_pred = estimator.predict(X_test)
dtAccuracy = accuracy_score(y_test, y_pred)
print(dtAccuracy)
plotEstimator(X_train, y_train, X_test, y_test, estimator, 'Decision Tree')
```

```
0.6060606060606061
```



1.2.2. Bagging

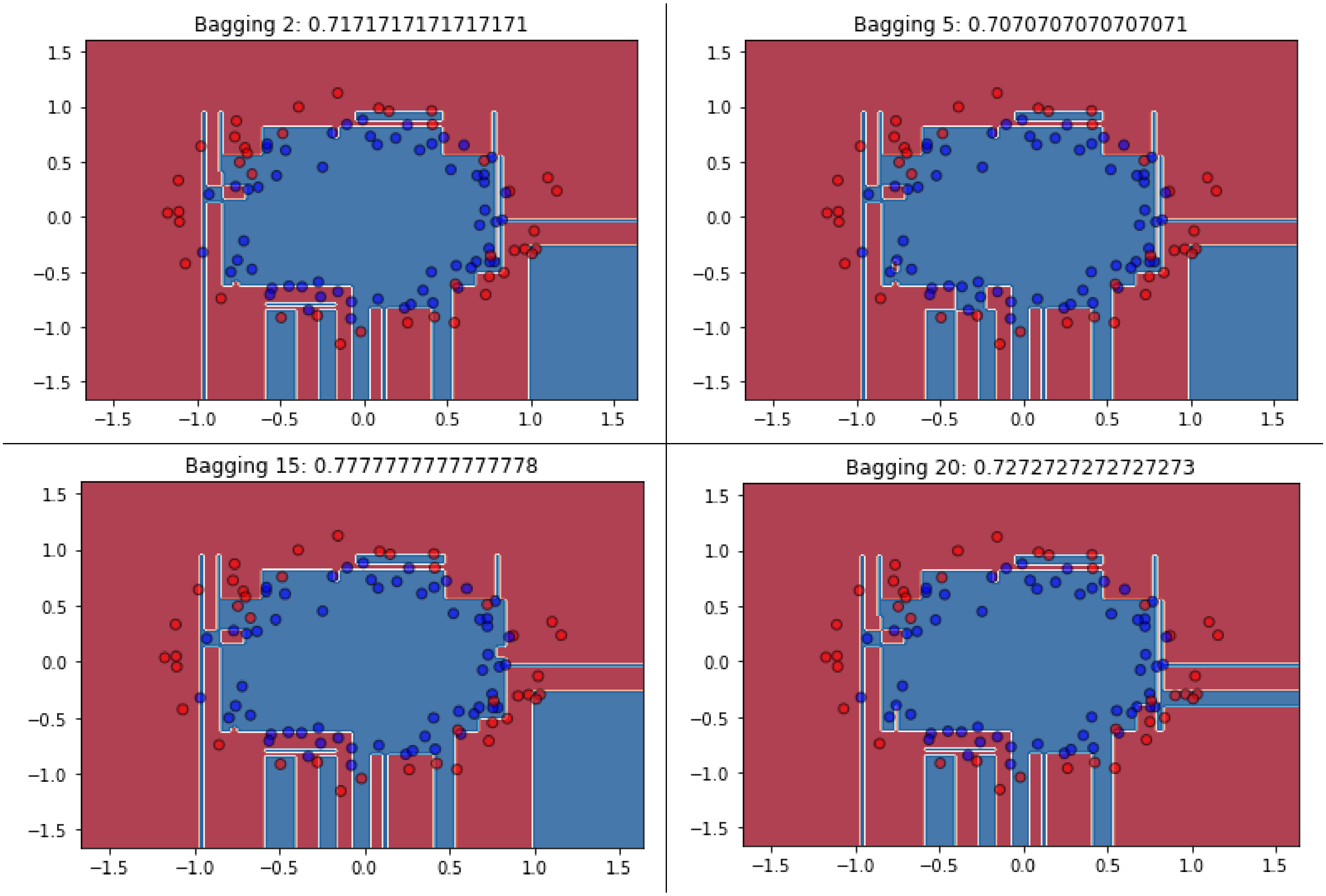
From Scratch

```
class Bagging():
    def __init__(self, n_estimators=10, max_depth = None):

        """
        Intializing the class with it's default paramters
        defalut number of bootstraped trees is 10
        maximum depth if not passed will be equal to none
        """
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self.trees = []

    def bagged_data(self, X, y):
        """
        boostrapping dataset by randomizing the the index of the rows
        as it will be the input of the decision tree
        """
        index = np.random.choice(np.arange(len(X)),len(X))
        return X[index], y[index]
```

```
def fit(self, X, y):
    for i in range(self.n_estimators):
        X_bagged, y_bagged = self.bagged_data(X,y)
        new_tree = DecisionTreeClassifier(random_state=rs)
        new_tree.fit(X_bagged,y_bagged)
        self.trees.append(new_tree)
def predict(self, X):
    """
    Uses the list of tree models built in the fit,
     The final prediction uses the mode of all the trees predictions.
    """
    self.predicts = []
    for tree in self.trees:
        self.predicts.append(tree.predict(X))
    self.pred_by_row = np.array(self.predicts).T

    predictions = []
    for row in self.pred_by_row:
        predictions.append(collections.Counter(row).most_common(1)[0][0])
    return predictions
```

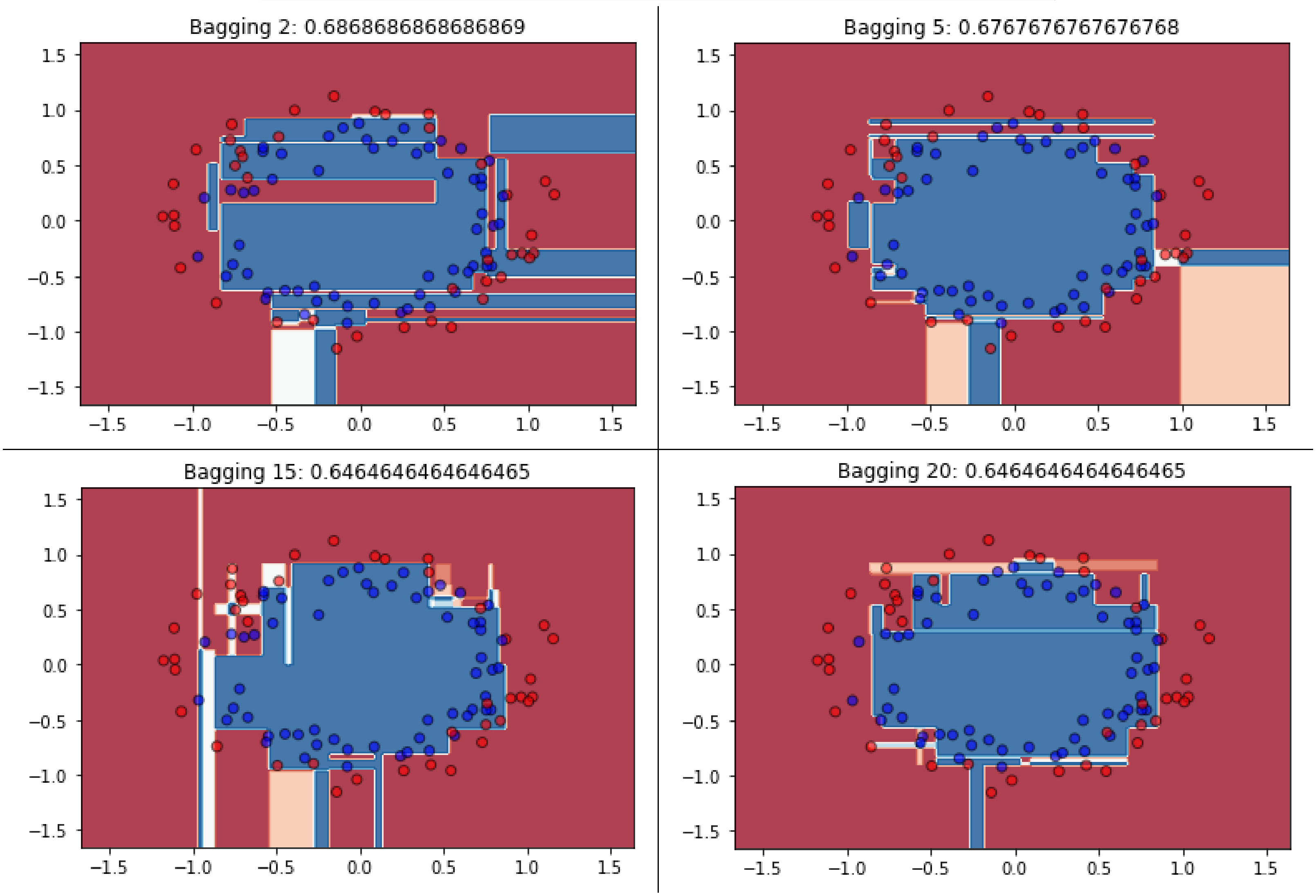
```
def score(self, X, y):
    """
    Uses the predict method to measure the accuracy of the model.
    """
    pred = self.predict(X)
    correct = 0
    for i,j in zip(y,pred):
        if i == j:
            correct+=1
    return float(correct)/float(len(y))
```

```
estimators = [2,5,15,20]
np.random.seed(rs)
for nEst in estimators:
    estimator_maded = Bagging()
    estimator_maded.fit(X_train, y_train)
    score = estimator_maded.score(X_test, y_test)
    plotEstimator(X_train, y_train, X_test, y_test, estimator,
                  f'Bagging {nEst}: {score}')
```



Another Technique

```
np.random.seed(rs)
for i in [2,5,15,20]:
    estimators=[]
    for j in range(i):
        #Bootstraping dataset using resample function
        X_strapped = coo_matrix(X_train)
        resampled_X, X_strapped, resampled_y = resample(X_train, X_strapped,
                                                        y_train, random_state=j)
        #Building Bagging on decisiontree
        estimator = DecisionTreeClassifier()
        estimator.fit(resampled_X, resampled_y)
        #array of estimators
        estimators.append(['estimator'+str(j),estimator])
    #Voting Classifier
    voting_clf = VotingClassifier(estimators=estimators, voting='soft')
    voting_clf = voting_clf.fit(X_train, y_train)
    score = voting_clf.score(X_test, y_test)
    plotEstimator(resampled_X, resampled_y, X_test, y_test, voting_clf,
                  f'Bagging {i}: {score}')
```



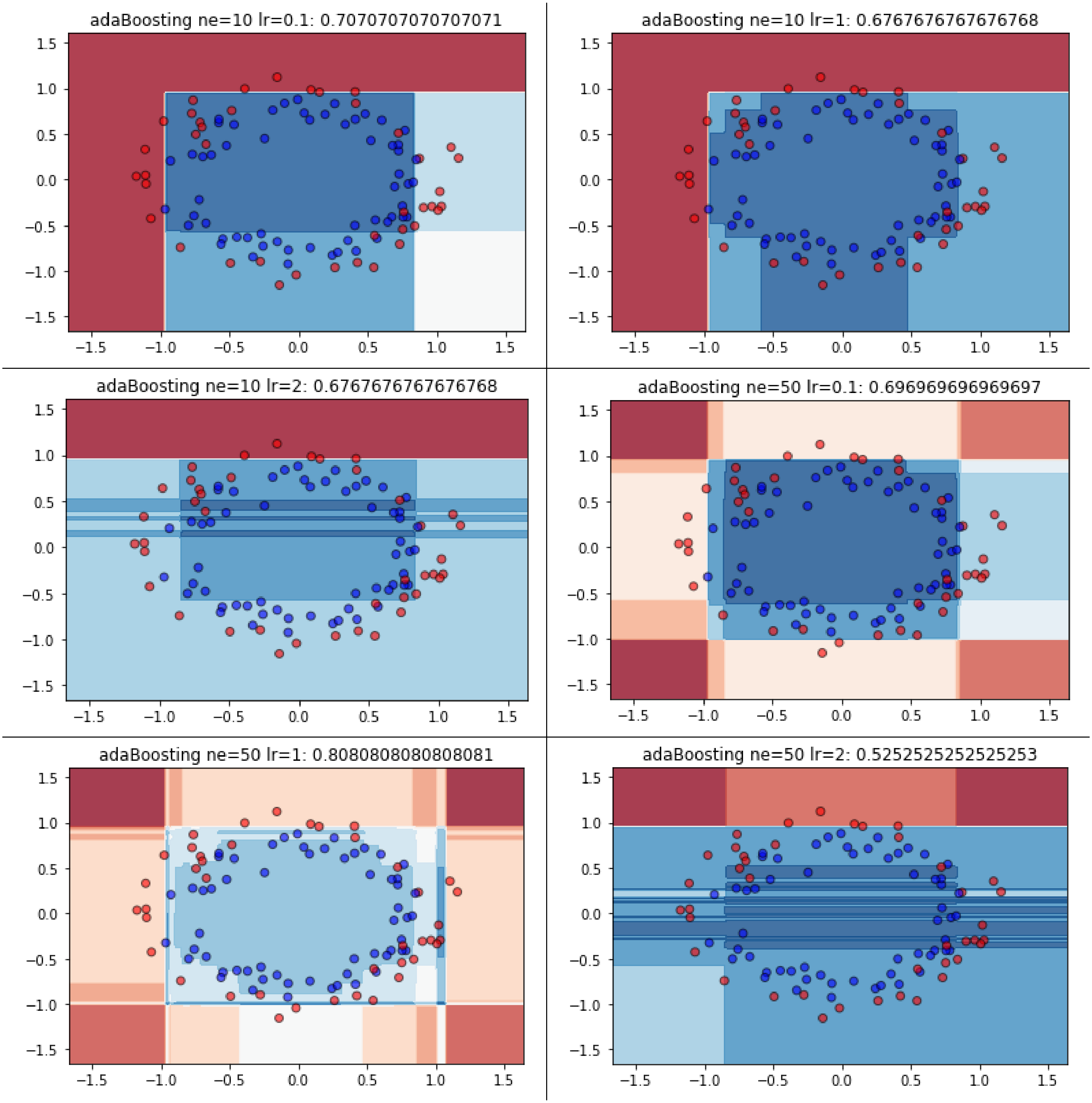
- ***<u>Explain why bagging can reduce the variance and mitigate the overfitting problem?</u>***
  - Variance reduction: if the training sets are completely independent, it will always helps to average an ensemble because this will reduce variance without affecting bias (e.g., bagging) -- reduce sensitivity to individual data pts.
    - Bagging, a Parallel ensemble method (stands for Bootstrap Aggregating), is a way to decrease the variance of the prediction model by generating additional data in the training stage. Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, with replacement. The size of the subsets is the same as the size of the original set. By sampling with replacement, some observations may be repeated in each new training data set. In the case of Bagging, every element has the same probability to appear in a new dataset. By increasing the size of the training set, the model's predictive force cannot be improved. It decreases the variance, helps with overfitting, and narrowly tunes the prediction to an excepted outcome.

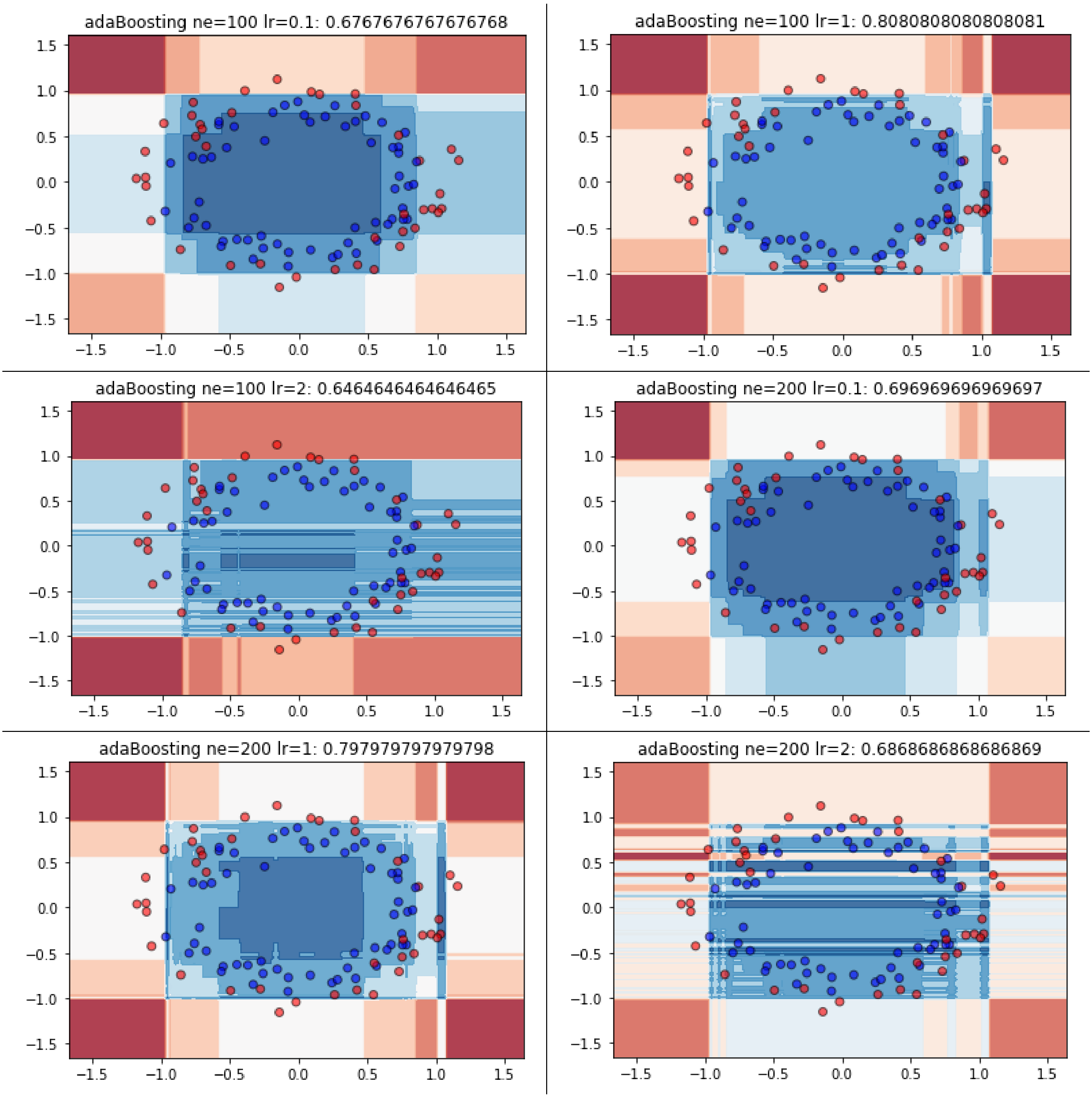
<u>In other words:</u>

- 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.

1.2.3. Boosting

```
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)
        plotEstimator(X_train, y_train, X_test, y_test, classifier,
                      f'adaBoosting ne={estimator} lr={learning_rate}: {score}')
```


adaBoosting ne=10 lr=0.1: 0.7070707070707071
adaBoosting ne=10 lr=1: 0.6767676767676768
adaBoosting ne=10 lr=2: 0.6767676767676768
adaBoosting ne=50 lr=0.1: 0.696969696969697
adaBoosting ne=50 lr=1: 0.8080808080808081
adaBoosting ne=50 lr=2: 0.5252525252525253


adaBoosting ne=100 lr=0.1: 0.6767676767676768
adaBoosting ne=100 lr=1: 0.8080808080808081
adaBoosting ne=100 lr=2: 0.6464646464646465
adaBoosting ne=200 lr=0.1: 0.696969696969697
adaBoosting ne=200 lr=1: 0.797979797979798
adaBoosting ne=200 lr=2: 0.6868686868686869