Assignment 4

Group 4:

Group names:

1. Amira Abu Issa
2. Aya Metwally
3. Heba Mostafa

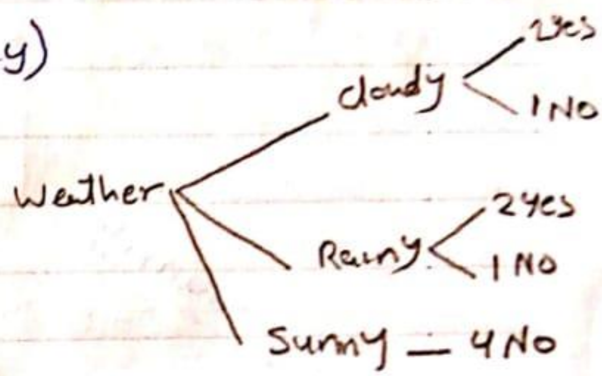
Part 1: Numerical Questions

(a) Please build a decision tree by using Gini Children (i.e., Gini = Pk i=1 ni n GINI(i), where NC is the number of classes).

a

our DISition is "Hiking"

First we calculate Gini of "weather"



- Gini (Hiking, weather = cloudy)

$= 1 - [(P_{yes})^2 + (P_{No})^2]$

$= 1 - [(\frac{2}{3})^2 + (\frac{1}{3})^2]$

$= \frac{4}{9}$

- Gini (Hiking, weather = Sunny)

$= 1 - (\frac{4}{4})^2 = 0$

- Gini (Hiking, weather = Rainy)

$= 1 - [(\frac{2}{3})^2 + (\frac{1}{3})^2]$

$= \frac{4}{9}$

Gini children (weather)

$= Gini(cloudy) * P_{cloudy} + Gini(Sunny) * P_{sunny} + Gini(Rainy) * P_{Rainy}$

$= (\frac{4}{9} * \frac{3}{10}) + (0 * \frac{4}{10}) + (\frac{4}{9} * \frac{3}{10})$

$= 0.267$

- we calculte Gini of "Temperature"

- Temperature
  - Hot $\frac{4}{10}$
    - 3 yes
    - 1 No
  - Mild $\frac{4}{10}$
    - 2 yes
    - 2 No
  - Cool $\frac{1}{10}$ — No
  - Cold $\frac{1}{10}$ — yes

$$Gini(Hot) = 1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right] = 0.375$$

$$Gini(Mild) = 1 - \left[\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right] = 0.5$$

$$Gini(cool) = 1 - \left(\frac{1}{1}\right)^2 = 0$$

$$Gini(Cold) = 1 - \left(\frac{1}{1}\right)^2 = 0$$

Gini children (Temperature)

$$= \left(0.375 * \frac{4}{10}\right) + \left(0.5 * \frac{4}{10}\right) + \left(0 * \frac{1}{10}\right) + \left(0 * \frac{1}{10}\right)$$

$$= \cancel{0.35}\ 0.35$$

- we calculate Gini of Humidty

$$Gini(High) = 1 - \left[\left(\frac{5}{6}\right)^2 + \left(\frac{1}{6}\right)^2\right] = \frac{5}{18}$$

$$Gini(Normal) = 1 - \left[\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right] = \frac{3}{8}$$

Gini children (Humidty)

$$= \left(\frac{5}{18} * \frac{6}{10}\right) + \left(\frac{3}{8} * \frac{4}{10}\right) = 0.316$$

- Humidty
  - $\frac{6}{10}$ High
    - 5 No
    - 1 yes
  - $\frac{4}{10}$ Normal
    - 3 yes
    - 1 No

- we Calculate "wind"

wind < 7 strong < 5 No, 2 yes
wind < 3 weak < 2 yes, 1 No

$$Gini(strong) = 1 - \left[\left(\frac{5}{7}\right)^2 + \left(\frac{2}{7}\right)^2\right] = \frac{20}{49}$$

$$Gini(weak) = 1 - \left[\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right] = \frac{4}{9}$$

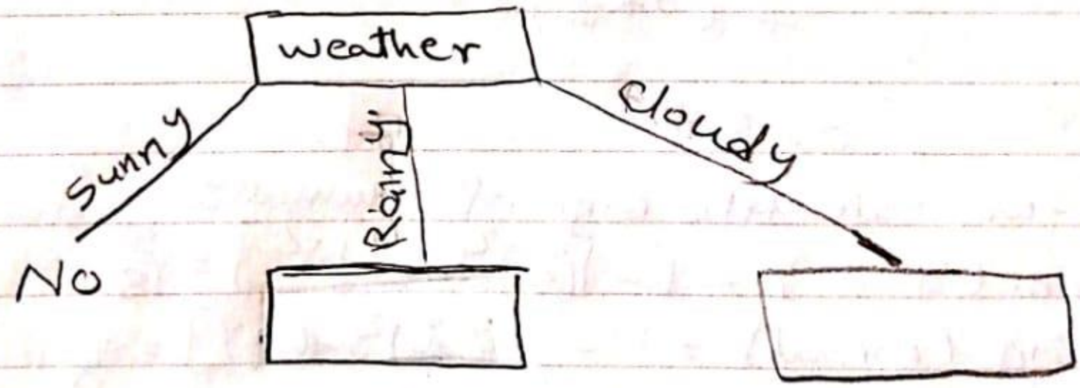
Gini Children(wind)

$$= \left(\frac{20}{49} * \frac{7}{10}\right) + \left(\frac{4}{9} * \frac{3}{10}\right)$$

$$= 0.419$$

The minumum Gini Children of "weather"

So the Root is "weather"



We must repeat this again to get new node

When weather = Cloudy

weather	Temperature	Humidty	wind	Hiking
Cloudy	Hot	High	Strong	No
Cloudy	mild	Normal	strong	yes
cloudy	mild	High	weak	yes

Gini of "Temperature

$$Gini(Hot) = 1 - [(\tfrac{1}{1})^2] = 0$$
$$Gini(mild) = 1 - [\tfrac{2}{2}]^2 = 0$$
$$Gini\ \text{~~Children~~ of Temperature} = (0 * \tfrac{1}{3}) + (0 * \tfrac{2}{3})$$
$$= 0$$

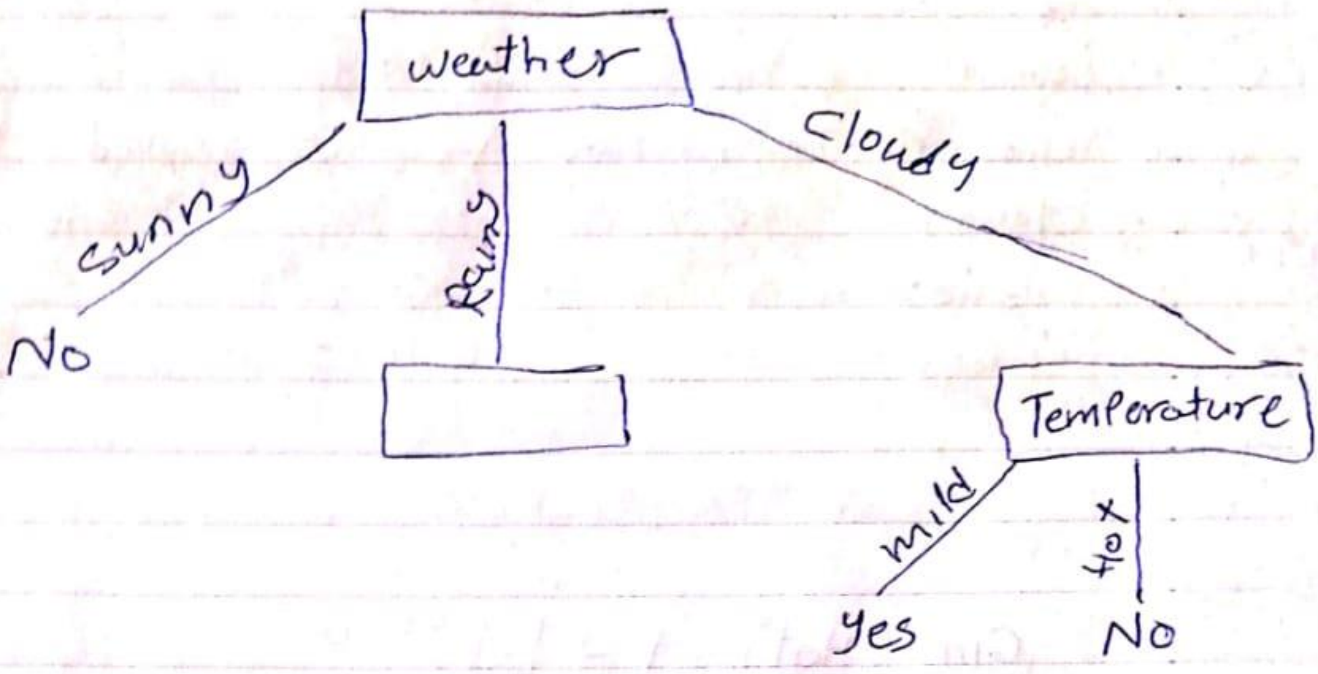
Gini of "Humidty"

$$Gini(High) = 1 - [(\tfrac{1}{2})^2 + (\tfrac{1}{2})^2] = 0.5$$
$$Gini(Normal) = 1 - [(1)^2] = 0$$
$$Gini\ Children(Humidty) = (0.5 * \tfrac{2}{3}) + (0 * \tfrac{1}{3})$$
$$= 0.33$$

Gini of "wind"

$$Gini(strong) = 1 - [(\tfrac{1}{2})^2 + (\tfrac{1}{2})^2] = 0.5$$
$$Gini(Weak) = 1 - [(1)^2] = 0$$
$$Gini\ Children(wind) = (0.5 * \tfrac{2}{3}) + (0 * \tfrac{1}{3})$$
$$= 0.33$$

The minumum Gini Children of "Temperature" so the new node is "Temperature" when (weather = Cloudy)



When weather = Rainy we will get Gini ("Temperature"), ("Humidty") and ("wind")
First we write our new Data when weather = Rainy

Weather	Temperature	Humidty	Wind	Hiking
Rainy	Cold	Normal	Strong	Yes
Rainy	Cool	Normal	Strong	No
Rainy	Hot	Normal	Weak	Yes

Gini of "Temperature"

$$Gini(Cold) = 1 - (1)^2 = 0$$
$$Gini(Cool) = 1 - (1)^2 = 0$$
$$Gini(Hot) = 1 - (1)^2 = 0$$

$$Gini\ children\ (Temperature) = (0 * \tfrac{1}{3}) + (0 * \tfrac{1}{3}) + (0 * \tfrac{1}{3})$$
$$= 0$$

Gini of "Humidty"

Gini (Normal) = $1 - ((\frac{2}{3})^2 + (\frac{1}{3})^2)$

$= \frac{4}{9}$

Gini children (Humidty) = $(\frac{4}{9} * \frac{3}{3}) = \frac{4}{9}$

Gini of "wind"

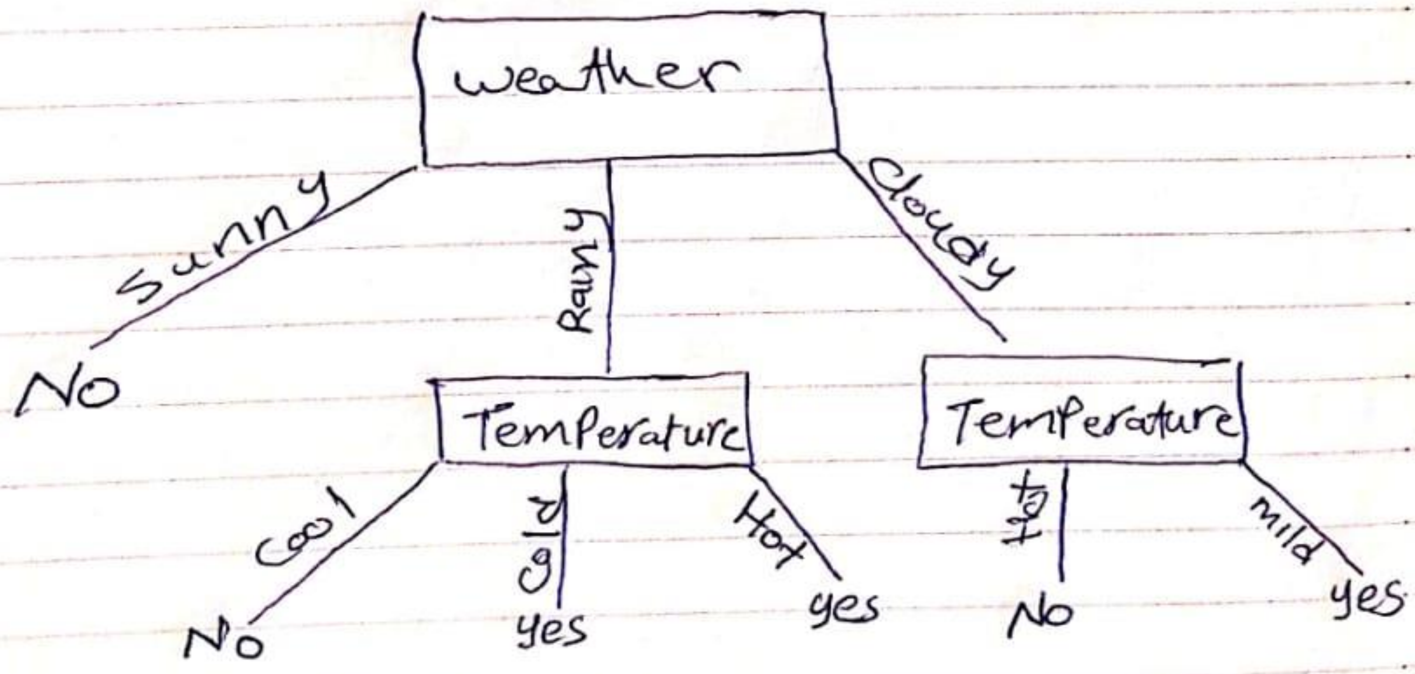
Gini(Strong) = $1 - ((\frac{1}{2})^2 + (\frac{1}{2})^2) = 0.5$

Gini(weak = $1 - (1)^2 = 0$

Gini children (wind) = $(0.5 * \frac{2}{3}) + (0 * \frac{1}{3})$

$= 0.33$

The minumum Gini children is Gini of "Temperature" when (weather = "Rainy")



This the disition tree when we use Gini method

(b) Please build a decision tree by using Information Gain (i.e., IG(T, a) = Entropy(T)− Entropy(T|a), More information about IG).

---

b)

$$\text{Entropy}(S) = -P_{yes}\log_2 P_{yes} - P_{No}\log_2 P_{No}$$

$$= -\frac{4}{10}\log_2\frac{4}{10} - \frac{6}{10}\log_2\frac{6}{10}$$

$$= 0.971$$

Temperature

$$\text{GAIN}(S, \cancel{\text{Temperature}}) = \text{Entropy}(S) - P_{Hot}[\text{Entropy}(Hot)]$$
$$- P_{mild}[-\text{Entropy}(mild)]$$
$$- P_{cool}[\text{Entropy}(cool)]$$
$$- P_{cold}[\text{Entropy}(cold)]$$
$$= 0.971 - \frac{4}{10}\left[-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right]$$
$$-\frac{4}{10}\left[\frac{-3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right]$$
$$-\frac{1}{10}\left[-\log_2 1\right] - \frac{1}{10}\left[-\log_2 1\right]$$

$$= 0.247$$

Weather

$$\text{GAIN}(S, \cancel{\text{Temperature}}) = \text{Entropy}(S) - P_{cloudy}[-\text{Entropy}_{cloudy}]$$

$$- P_{Rainy}[-\text{Entropy}_{Rainy}]$$
$$- P_{sunny}[-\text{Entropy}_{sunny}]$$

$$= 0.971 - \frac{3}{10}\left[-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right]$$
$$-\frac{4}{10}\left[-\frac{4}{4}\log_2\frac{4}{4}\right]$$
$$-\frac{3}{10}\left[\frac{-2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right]$$

$GAin(s, weather) = 0.42$

$GAin(S, Humdity) = \{-P_{High}[-Entropy_{High}] - P_{normal}[-Entropy_{normal}]$
$+ Entropy(S)$

$$= 0.971 - \frac{6}{10}\left[\frac{-5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}\right] - \frac{4}{10}\left[\frac{-3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right]$$

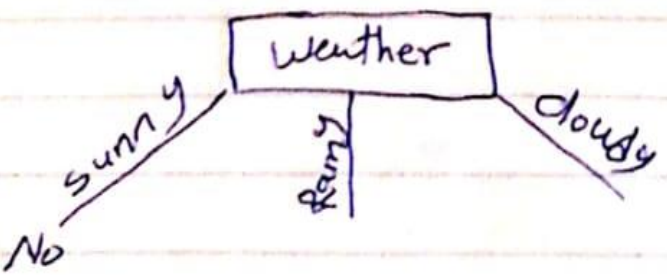
$$= 0.257$$

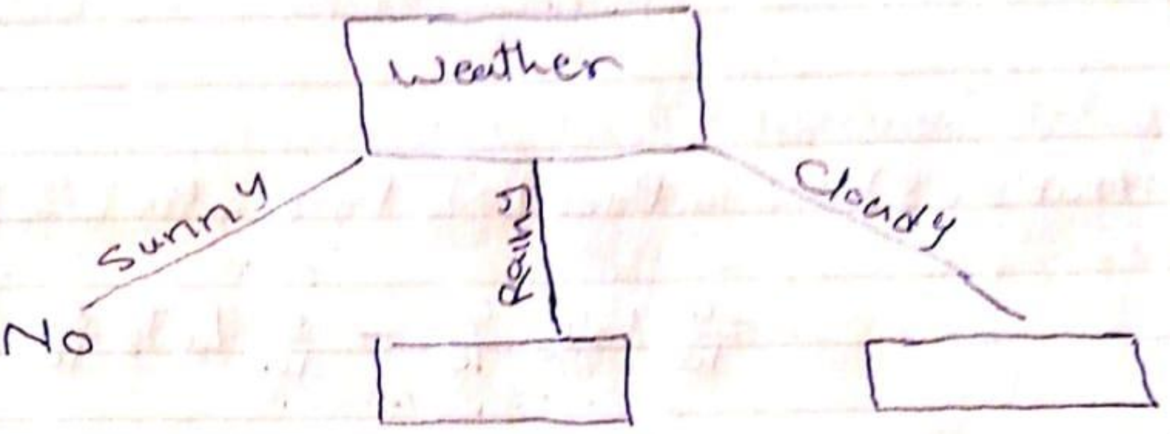
$GAin(S, wind) = Entropy(S) - P_{strong}[-Entropy_{strong}] - P_{weak}[-Entropy_{weak}]$

$$= 0.971 - \frac{7}{10}\left[\frac{-2}{7}\log_2\frac{2}{7} - \frac{5}{7}\log_2\frac{5}{7}\right] - \frac{3}{10}\left[\frac{-1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right]$$

$$= 0.0913$$

The maximum information GAin is the Root so the the Root is "weather"





our new Data when (weather = cloudy)

weather	Temperature	Humidity	Wind	Hiking
cloudy	Hot	High	Strong	No
cloudy	mild	Normal	strong	yes
Cloudy	mild	High	weak	yes

Entropy (S, weather = cloudy)

$= -P_{yes} \log_2 P_{yes} - P_{No} \log_2 P_{No}$

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

$= 0.918$

GAin (S, Temperature) = Entropy (S) $- \sum \frac{n_i}{n}$ Entropy(i)

$= 0.918 - \frac{2}{3}\left[\frac{-2}{2} \log_2 \frac{2}{2}\right] - \frac{1}{3}\left[1 \log_2 1\right]$

$= 0.918$

GAin (S, Humidity) = Entropy (S) $- \sum$

$= 0.918 - \frac{2}{3}\left[\frac{-1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2}\right] - \frac{1}{3}\left[-\log_2 1\right]$

$GAin(S, Humidty) = 0.251$

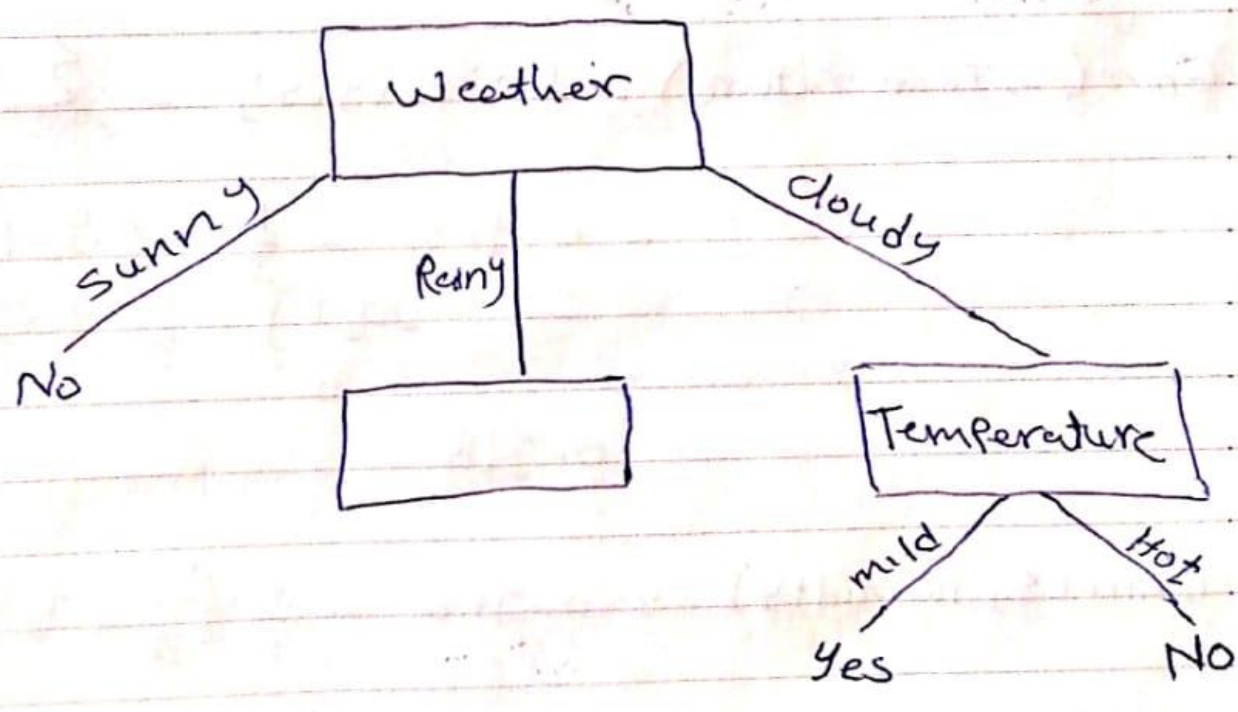
$$GAin(S, wind) = Entropy(S) - \sum \frac{n_i}{n} Entropy(i)$$

$$= 0.918 - \frac{2}{3}\left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right]$$

$$- \frac{1}{3}\left[-\log_2 1\right]$$

$$= 0.251$$

The new node For (weather = Cloudy) IS "Temperature"



our new Data (weather = Rainy)

weather	Temperature	Humidity	wind	Hiking
Rainy	Cold	Normal	strong	yes
Rainy	CooL	Normal	strong	No
Rainy	Hot	Normal	weak	yes

(weather = Rainy)

$$\text{Entropy}(S) = -P_{yes}\log_2 P_{yes} - P_{No}\log_2 P_{No}$$

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

$$= 0.918$$

$$\text{GAin}(S, \text{Temperature}) = \text{Entropy}(S) - \sum \frac{n_i}{n}\text{Entropy}(i)$$

$$= 0.918 - \frac{1}{3}[-\log_2 1] - \frac{1}{3}[-\log_2 1] - \frac{1}{3}[-\log_2 1]$$

$$= 0.918$$

$$\text{GAin}(S, \text{Humidty}) = 0.918 - \frac{3}{3}\left[-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right]$$

$$= -0.0003$$

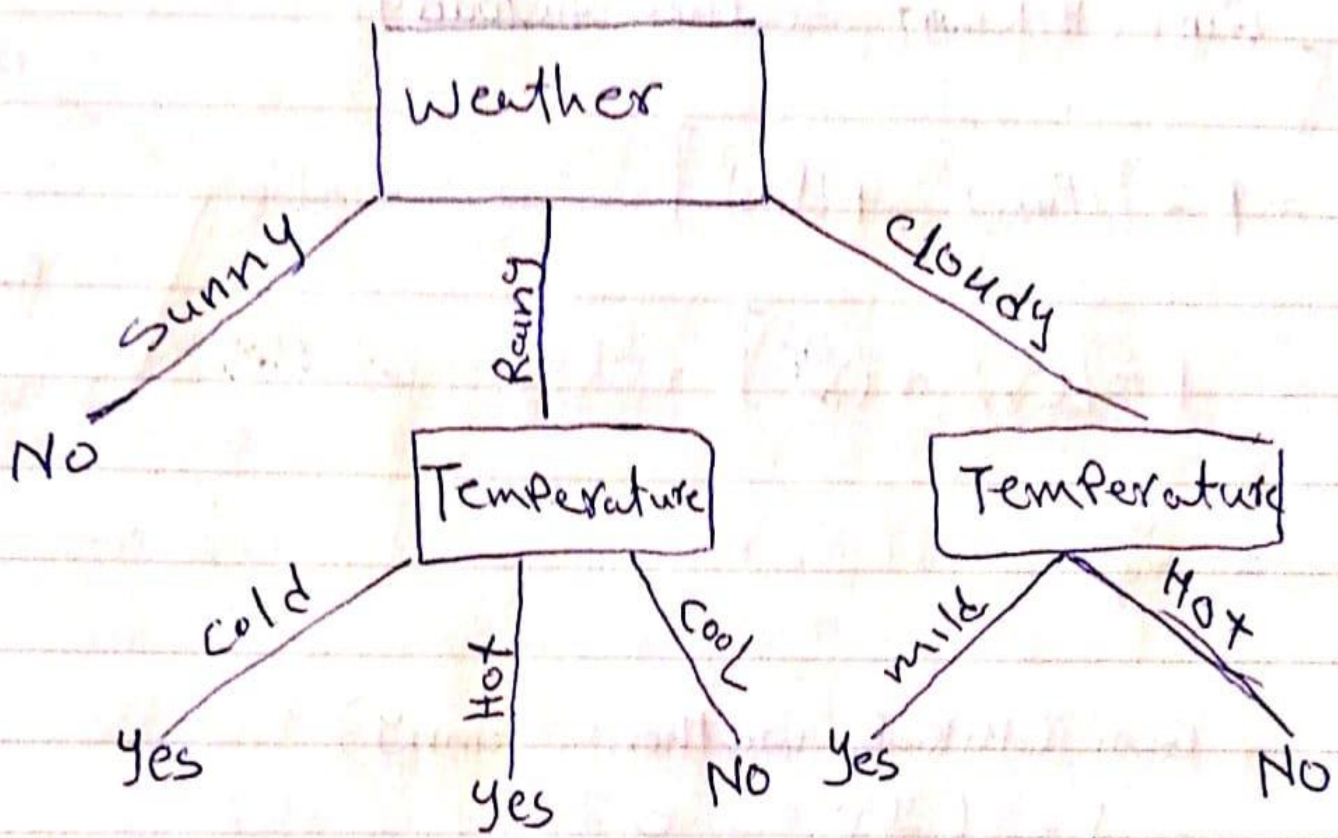
$$\text{GAin}(S, \text{wind}) = \text{Entropy}(S) - \sum \frac{n_i}{n}\text{Entropy}(i)$$

$$= 0.918 - \frac{2}{3}\left[-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right] - \frac{1}{3}[-\log_2 1]$$

GAin(s, wind) = 0.251

The new node for (weather = Rainy)

is "Temperature"



This is the decition tree when we Calculte by information Gain

(c) <u>Comparison of Gini Index and Information Gain:</u>

Both Gini index and information gain are commonly used metrics for building decision trees.

Advantages of Gini Index:

- Gini index is computationally faster than information gain, especially when dealing with large datasets.

- Gini index is less prone to overfitting than information gain, since it does not rely on the number of instances in each class.

Disadvantages of Gini Index:

- Gini index is not as informative as information gain, since it only considers the impurity of the current node and not the potential future nodes.

- Gini index is biased towards features with many categories, since they tend to have lower Gini index values.

Advantages of Information Gain:

- Information gain provides more information about the relationship between the features and the label, since it considers the entropy of both the current node and the potential future nodes.

- Information gain is less biased towards features with many categories, since it takes into account the number of instances in each category.

Disadvantages of Information Gain:

- Information gain is computationally more expensive than Gini index, especially when dealing with large datasets.

- Information gain is more prone to overfitting than Gini index, especially when dealing with noisy or irrelevant features.

Part 2: Programming Questions

(a)

View the dataset:

```
# Load dataset
dataset = pd.read_csv('/content/KDD.csv')
# print shape of dataset
print("Shape of dataset",dataset.shape)
# show the first five rows
dataset
```

Shape of dataset (494021, 39)

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	0	181	5450	0	0	0	0	0	1	0	...	9	1.0	0.0
1	0	239	486	0	0	0	0	0	1	0	...	19	1.0	0.0
2	0	235	1337	0	0	0	0	0	1	0	...	29	1.0	0.0
3	0	219	1337	0	0	0	0	0	1	0	...	39	1.0	0.0
4	0	217	2032	0	0	0	0	0	1	0	...	49	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
494016	0	310	1881	0	0	0	0	0	1	0	...	255	1.0	0.0
494017	0	282	2286	0	0	0	0	0	1	0	...	255	1.0	0.0
494018	0	203	1200	0	0	0	0	0	1	0	...	255	1.0	0.0
494019	0	291	1200	0	0	0	0	0	1	0	...	255	1.0	0.0
494020	0	219	1234	0	0	0	0	0	1	0	...	255	1.0	0.0

494021 rows × 39 columns

Function to preprocess the dataset:

```python
# function to process dataset
def preprocess_data(df):
    # Drop rows with missing values as ValueError: Input X contains NaN.
    #df = df.dropna()
    # Separate the input features (X) and target variable (Y)
    # Select all columns except the last one
    X = df.iloc[:, :-1]
    # Select the last column
    Y = df.iloc[:, -1]
    #  view 38 input feature variables and 1 target
    print("The feature variables:",X.shape[1])
    print("The shape of target:",Y.shape)
    # Normalize the input features using MinMaxScaler
    scaler = MinMaxScaler()
    X_normalized = scaler.fit_transform(X)
    # Perform filter-based feature selection
    # Select top 9 features
    selector = SelectKBest(mutual_info_classif, k=9)
    X_selected = selector.fit_transform(X_normalized, Y)
    # Create a new DataFrame with selected features
    selected_features = selector.get_support(indices=True)
    selected_columns = df.columns[selected_features].tolist()
    my_data = pd.DataFrame(X_selected, columns=selected_columns)
    # Add the target column
    my_data['target'] = Y
    return my_data, X_selected, Y, selected_columns
# Preprocess data
my_data, X_selected, Y,selected_columns = preprocess_data(dataset)
```

Display shape of train dataset (features) and tests dataset (target):

```
The feature variables: 38
The shape of target: (494021,)
```

Display first 5 rows after preprocessing, normalization & feature selection:

	src_bytes	dst_bytes	logged_in	count	srv_count	dst_host_count	dst_host_srv_count	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	target
0	2.610418e-07	0.001057	1.0	0.015656	0.015656	0.035294	0.035294	0.11	0.0	0
1	3.446905e-07	0.000094	1.0	0.015656	0.015656	0.074510	0.074510	0.05	0.0	0
2	3.389216e-07	0.000259	1.0	0.015656	0.015656	0.113725	0.113725	0.03	0.0	0
3	3.158461e-07	0.000259	1.0	0.011742	0.011742	0.152941	0.152941	0.03	0.0	0
4	3.129617e-07	0.000394	1.0	0.011742	0.011742	0.192157	0.192157	0.02	0.0	0

(b)

Function to train DT and compute performance for 3 different subsets:

```python
# function to compute performance of DT in terms of Classification report using 3 subsets
def compute_DT_performance():
  # Define the train and test ratios
  train_ratios = [0.7, 0.6, 0.5]
  test_ratios = [0.3, 0.4, 0.5]
  i=1
  for train_ratio, test_ratio in zip(train_ratios, test_ratios):
      # Split the data into training and testing subsets
      X_train, X_test, Y_train, Y_test = split_data(X_selected,Y,train_ratio, test_ratio)
      # Perform further operations with the subsets, such as training and evaluating the decision tree
      clf = DecisionTreeClassifier()
      clf.fit(X_train, Y_train)
      # Make predictions on the test set and evaluate performance
      Y_pred = clf.predict(X_test)
      report, accuracy, confusion = evaluate_classification(clf, X_test, Y_test)
      print(f"my_data_{i}")
      # Print the results
      print(f"Train Ratio: {train_ratio}, Test Ratio: {test_ratio}")
      print("Classification Report:")
      print(report)
      print()
      i+=1
# call the compute_DT_performance
compute_DT_performance()
```

Display classification report for 3 different subsets:

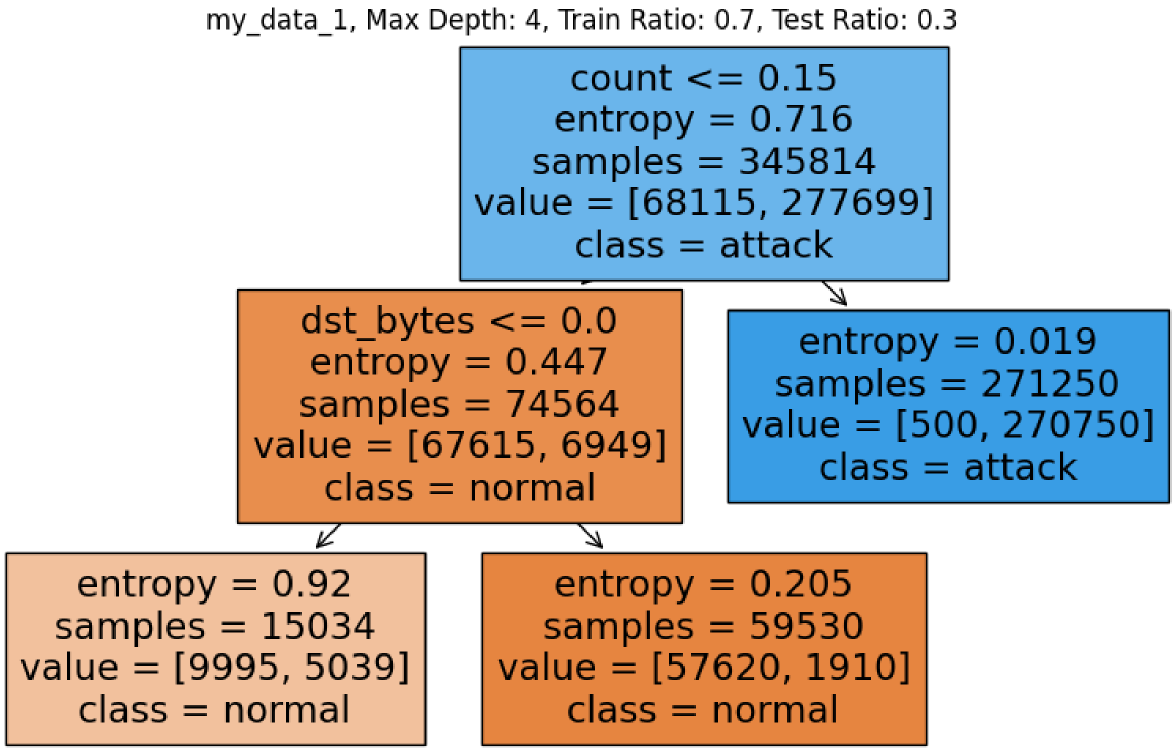
```
my_data_1
Train Ratio: 0.7, Test Ratio: 0.3
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29163
           1       1.00      1.00      1.00    119044

    accuracy                           1.00    148207
   macro avg       1.00      1.00      1.00    148207
weighted avg       1.00      1.00      1.00    148207


my_data_2
Train Ratio: 0.6, Test Ratio: 0.4
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     38962
           1       1.00      1.00      1.00    158647

    accuracy                           1.00    197609
   macro avg       1.00      1.00      1.00    197609
weighted avg       1.00      1.00      1.00    197609


my_data_3
Train Ratio: 0.5, Test Ratio: 0.5
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     48796
           1       1.00      1.00      1.00    198215

    accuracy                           1.00    247011
   macro avg       1.00      1.00      1.00    247011
weighted avg       1.00      1.00      1.00    247011
```

(c)

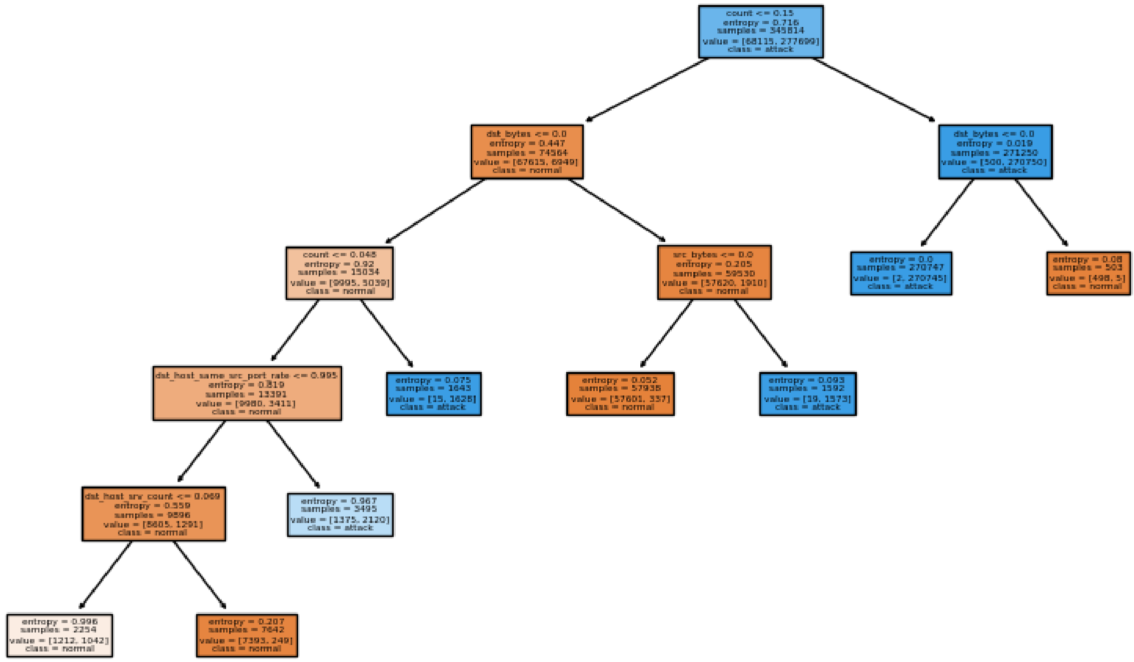
Function to visualize the best split of the Decision tree:

```
# function to train the decision tree model and visualize the best split of the Decision tree
def train_decision_tree():
    # Define the train and test ratios
    train_ratios = [0.7, 0.6, 0.5]
    test_ratios = [0.3, 0.4, 0.5]
    # Define max depths as assumed
    max_depths = [4, 6, 8]
    i = 0
    for max_depth in max_depths:
        for j in range(3):
            train_ratio = train_ratios[j]
            test_ratio = test_ratios[j]
            # Split the data into training and testing subsets
            X_train, X_test, Y_train, Y_test = split_data(X_selected,Y,train_ratio, test_ratio)
            # Initialize and train the Decision Tree classifier
            clf = DecisionTreeClassifier(max_depth=max_depth, criterion='entropy',max_leaf_nodes=ceil((2**max_depth)*0.1+1))
            clf.fit(X_train, Y_train)
            # Plot the decision tree
            plt.figure(figsize=(10, 6))
            plot_tree(clf, filled=True, feature_names=my_data.columns[:-1], class_names=['normal', 'attack'])
            n = j + 1
            plt.title(f"my_data_{n}, Max Depth: {max_depth}, Train Ratio: {train_ratio}, Test Ratio: {test_ratio}")
            plt.show()
            i += 1
```

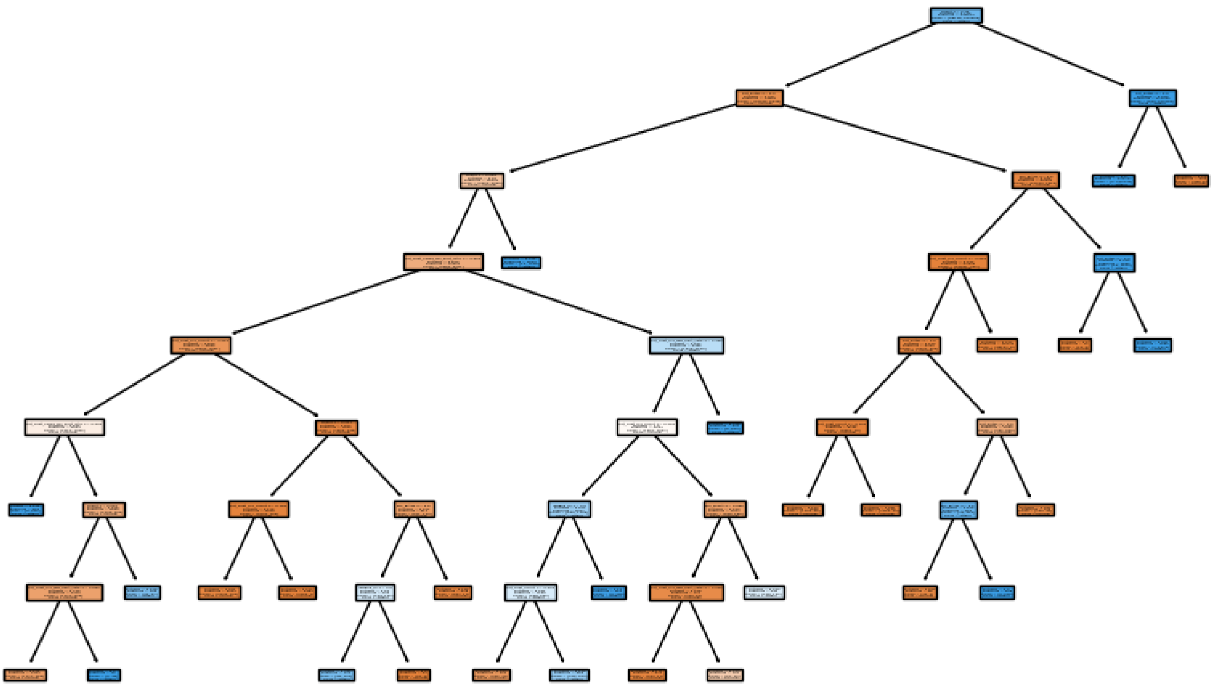
```
train_decision_tree()
```



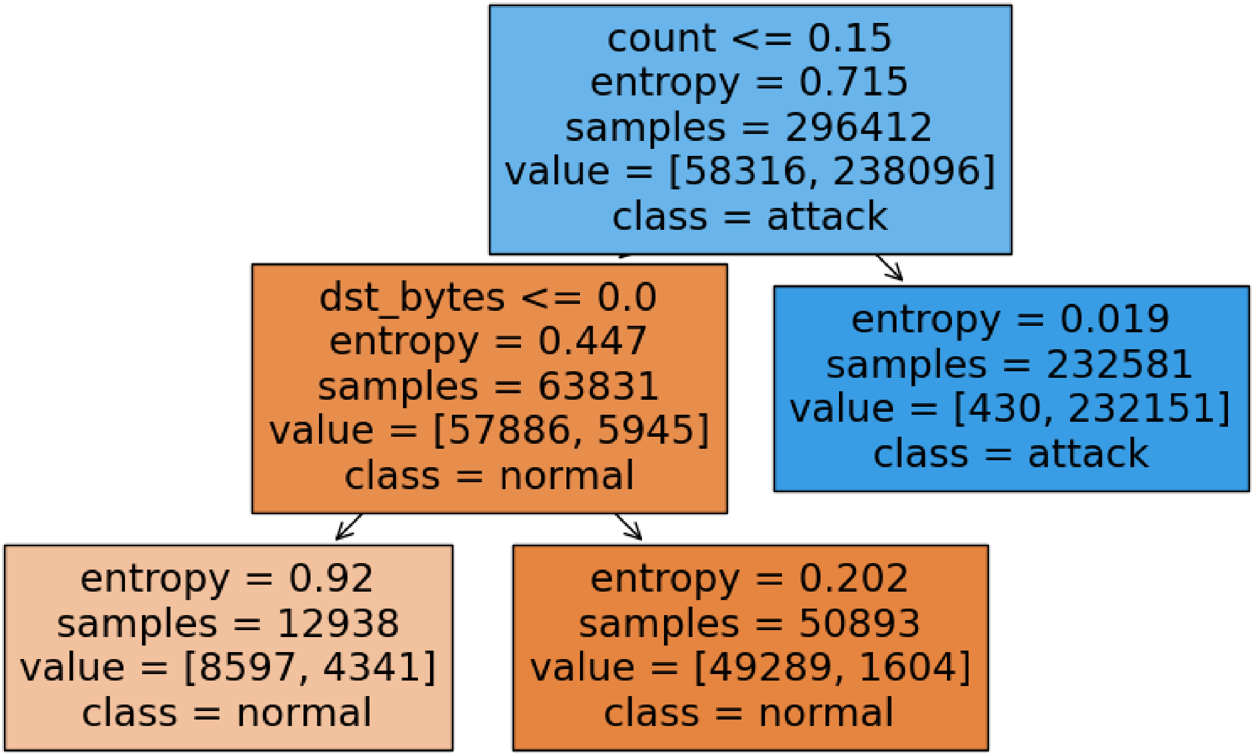
my_data_1, Max Depth: 6, Train Ratio: 0.7, Test Ratio: 0.3



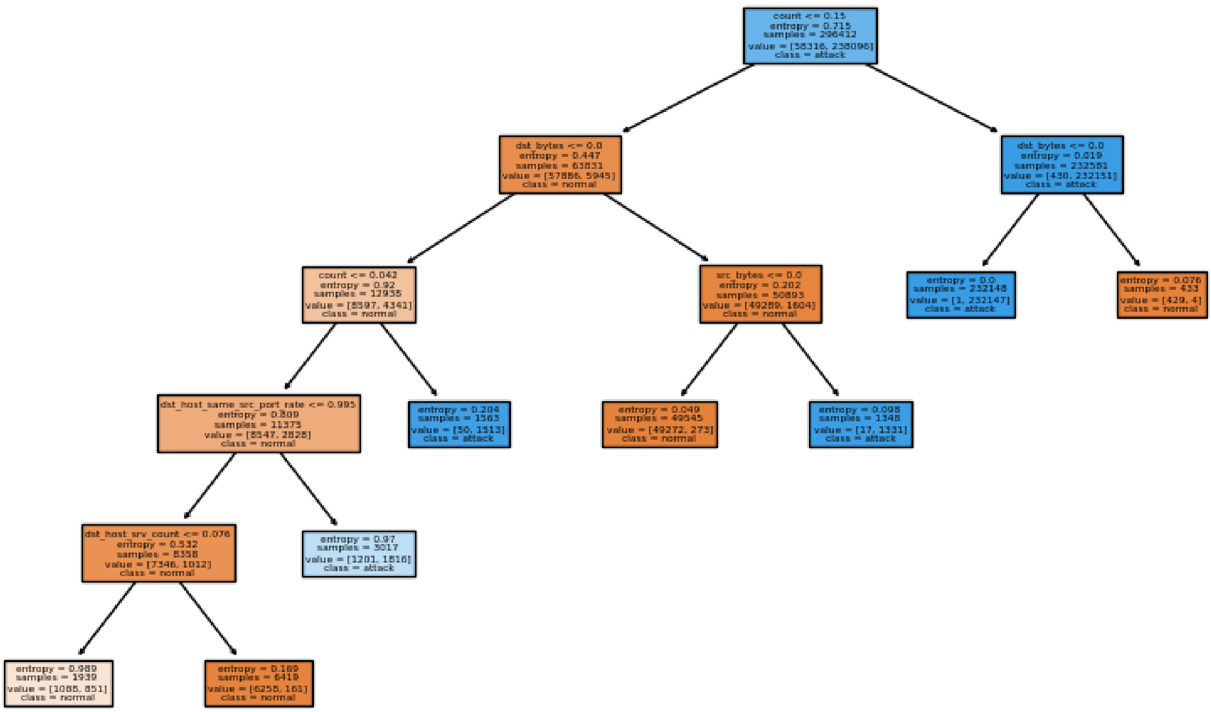
my_data_1, Max Depth: 8, Train Ratio: 0.7, Test Ratio: 0.3



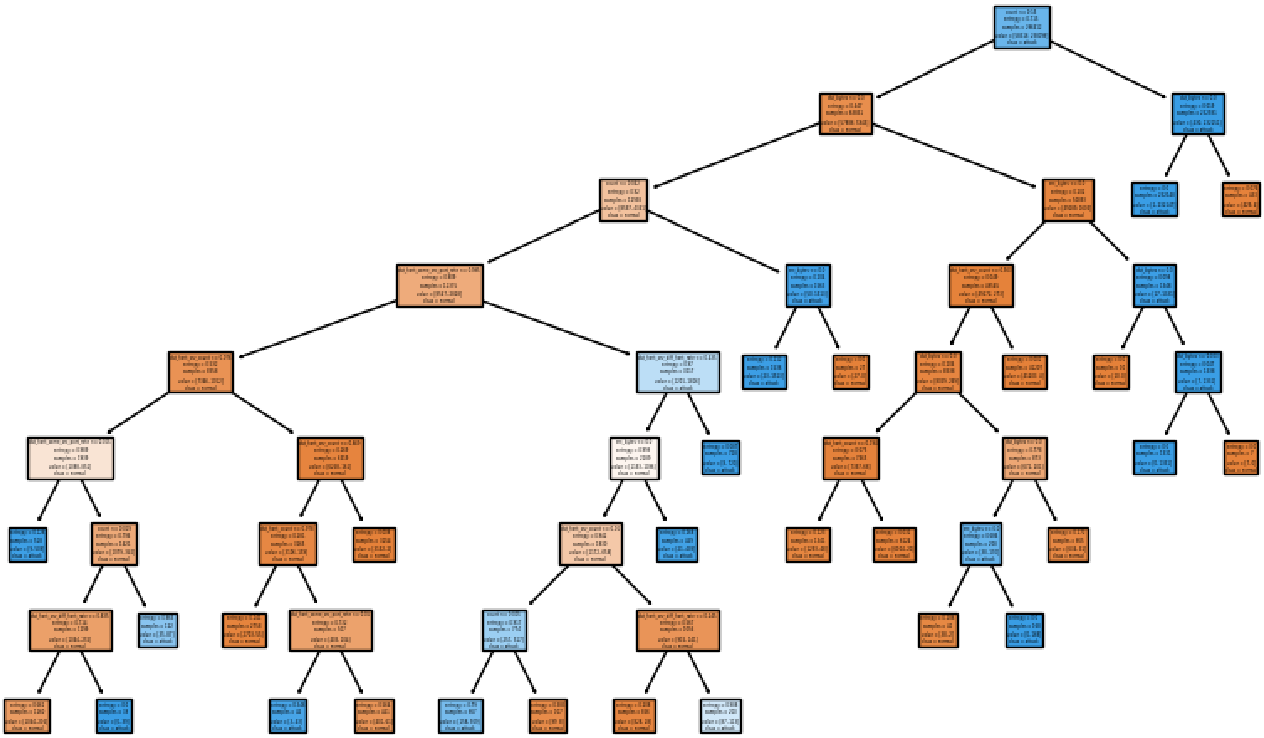
my_data_2, Max Depth: 4, Train Ratio: 0.6, Test Ratio: 0.4



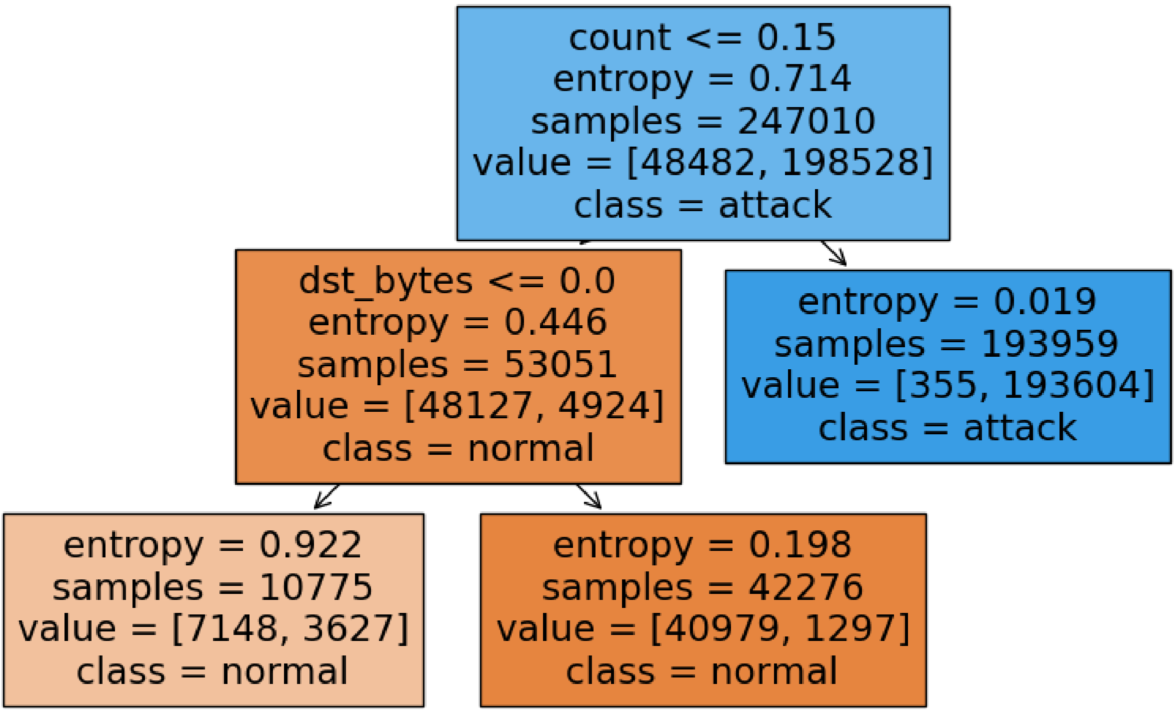
my_data_2, Max Depth: 6, Train Ratio: 0.6, Test Ratio: 0.4



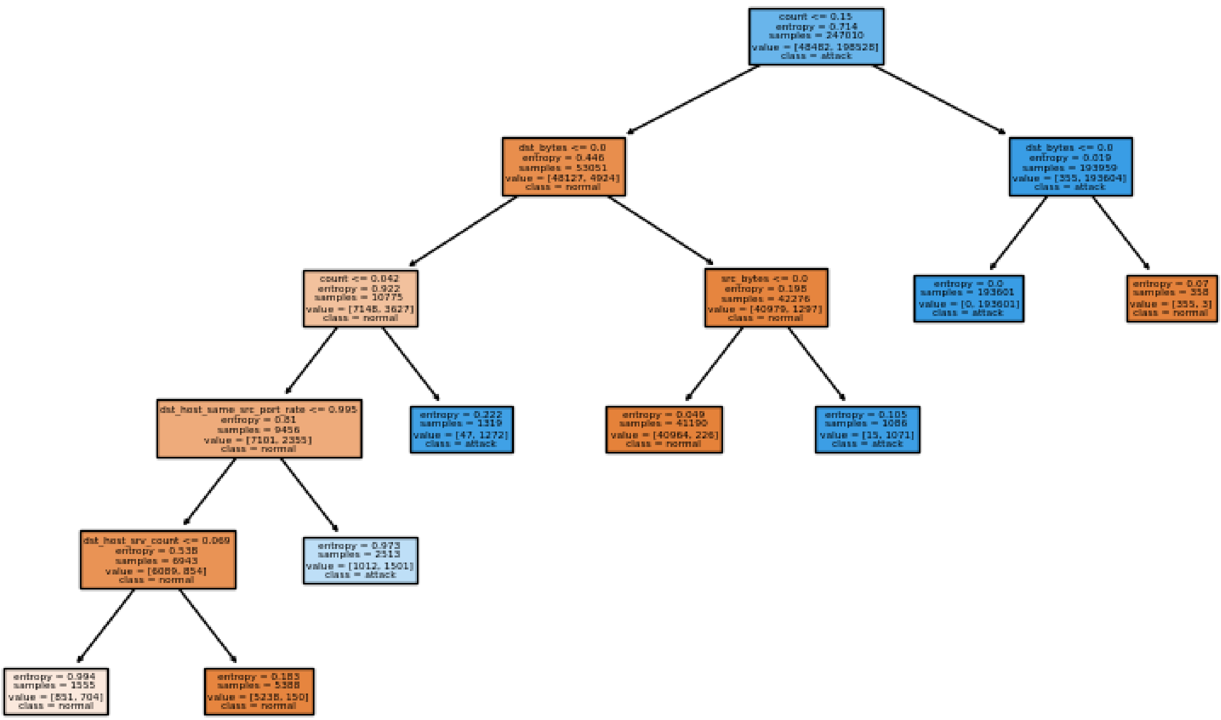
my_data_2, Max Depth: 8, Train Ratio: 0.6, Test Ratio: 0.4



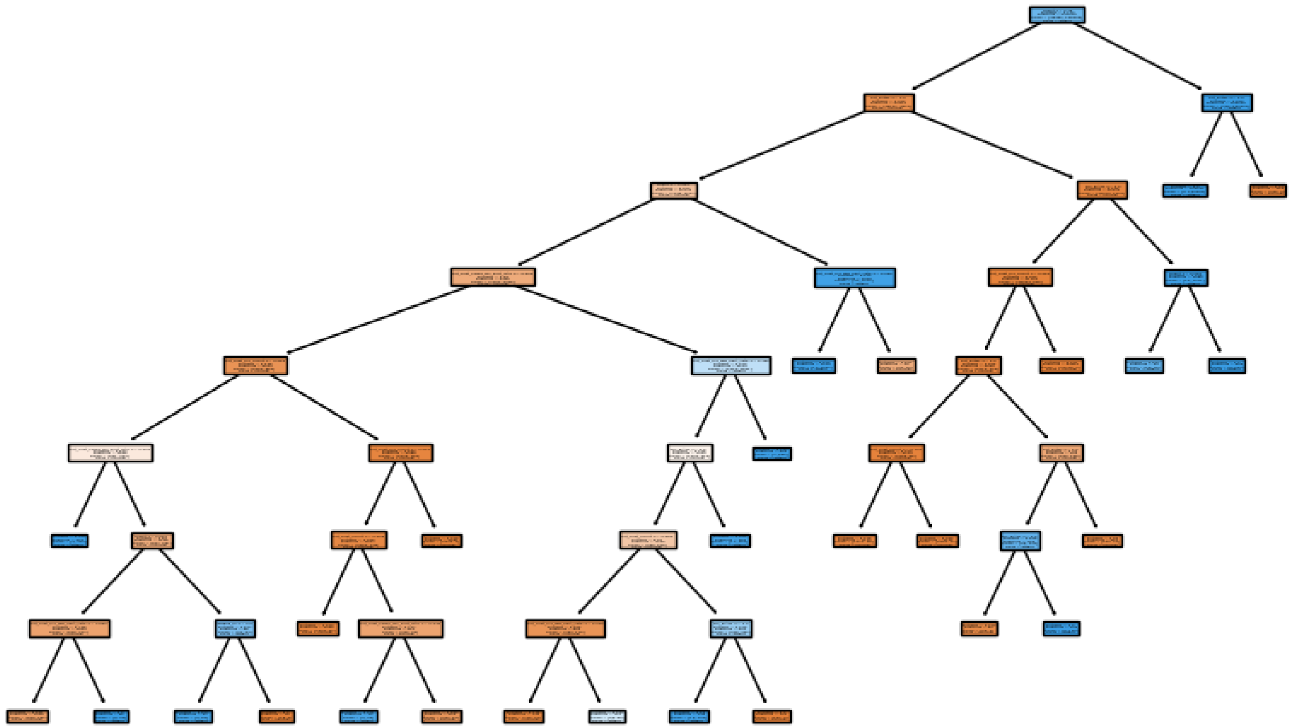
my_data_3, Max Depth: 4, Train Ratio: 0.5, Test Ratio: 0.5



my_data_3, Max Depth: 6, Train Ratio: 0.5, Test Ratio: 0.5



my_data_3, Max Depth: 8, Train Ratio: 0.5, Test Ratio: 0.5



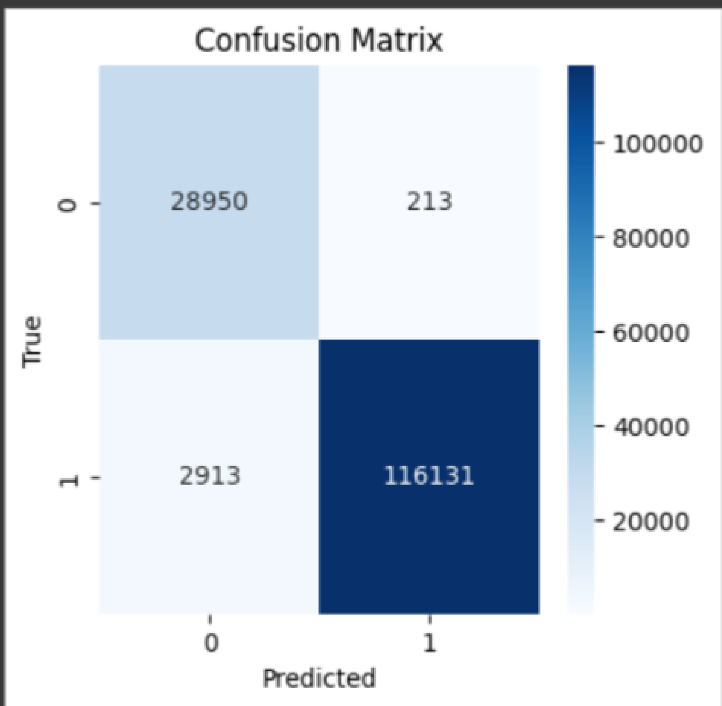
(d)

Function to compute performance of tuned DT:

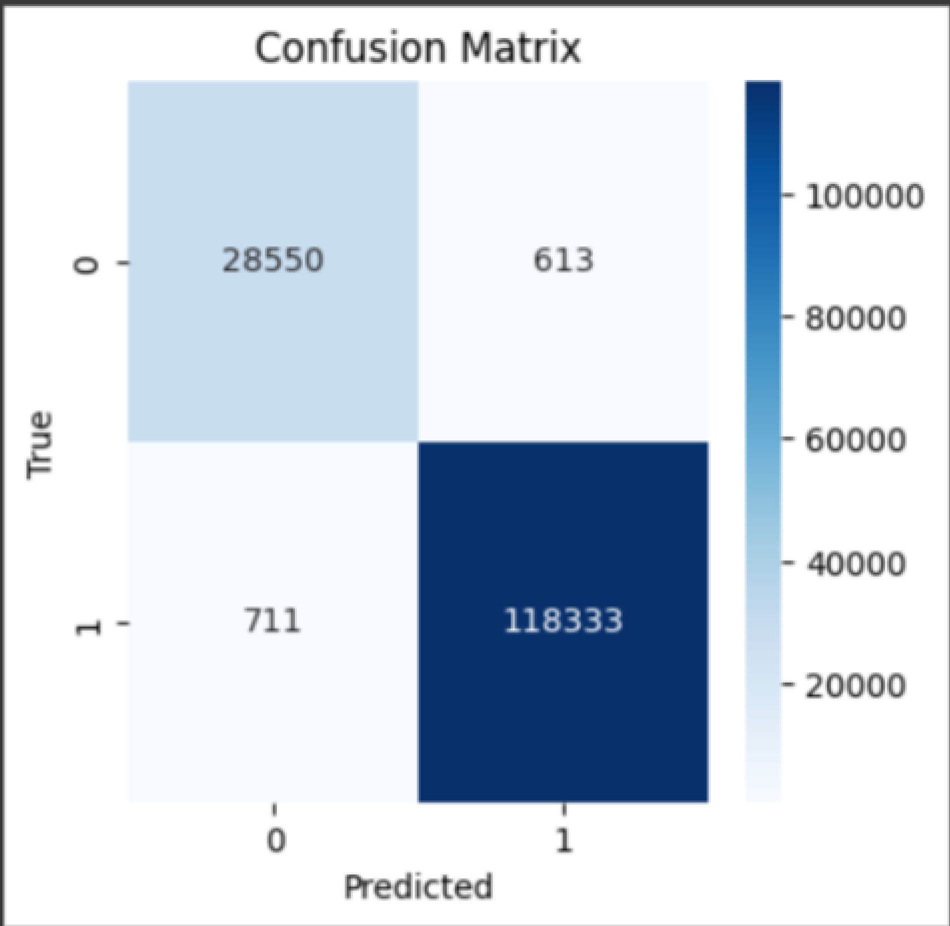
```python
# function to compute and compare tuned DT
def compare_tuned_decision_tree_performance(train_ratio, test_ratio):
    X_train, X_test, Y_train, Y_test = split_data(X_selected,Y, train_ratio, test_ratio)
    max_depths = [4, 6, 8]
    data_labels = {0.3: "my_data_1", 0.4: "my_data_2", 0.5: "my_data_3"}
    data_label = data_labels[test_ratio]
    for max_depth in max_depths:
        # Initialize and train the Decision Tree classifier
        clf = DecisionTreeClassifier(max_depth=max_depth, criterion='entropy',max_leaf_nodes=ceil((2**max_depth)*0.1+1))
        clf.fit(X_train, Y_train)
        # Make predictions on the test set
        Y_pred = clf.predict(X_test)
        report, accuracy, confusion = evaluate_classification(clf, X_test, Y_test)
        # Generate classification report and confusion matrix
        print(f"{data_label}, Max Depth: {max_depth}, Test Ratio: {test_ratio}")
        print("Accuracy:", accuracy)
        print("Classification Report:")
        print(report)
        print(f"Confusion Matrix with: Max Depth: {max_depth}, {test_ratio} test data")
        print()
        plot_confusion_matrix(confusion)
        print()
```

Display the accuracy scores, classification report, and confusion matrix of tuned classifier for each subset:

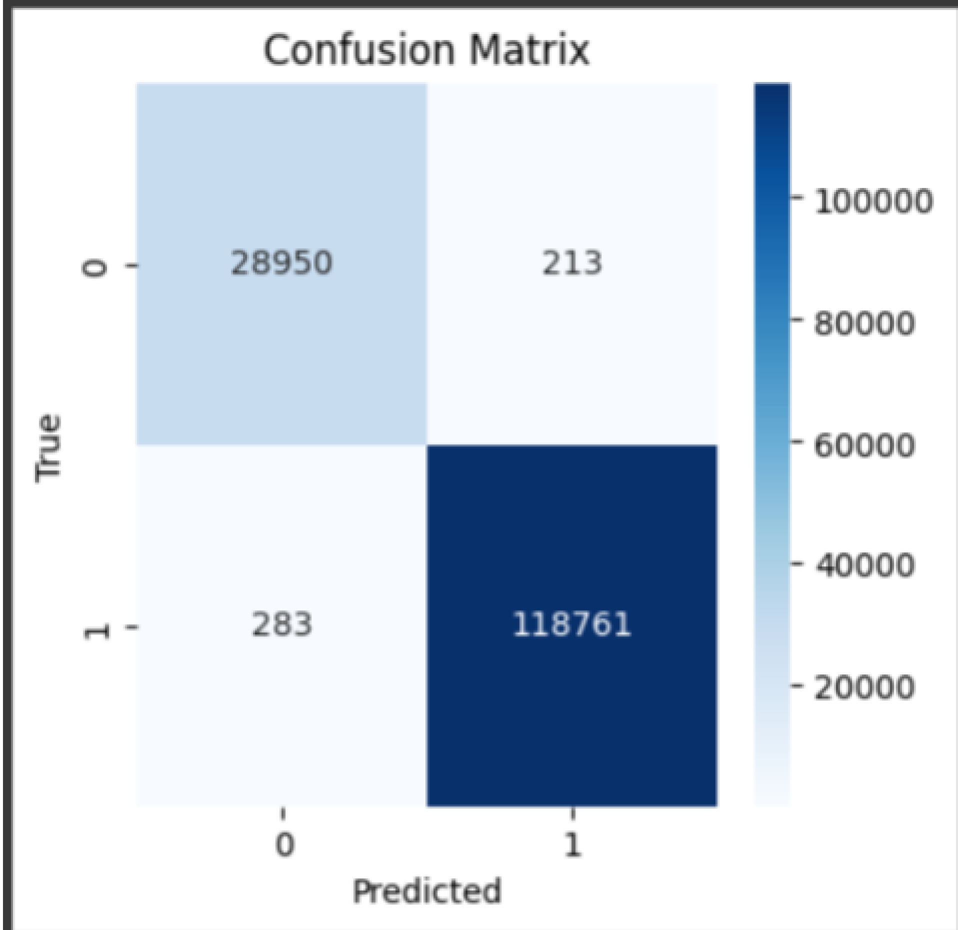
```
my_data_1, Max Depth: 4, Test Ratio: 0.3
Accuracy: 0.9789078788451288
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     29163
           1       1.00      0.98      0.99    119044

    accuracy                           0.98    148207
   macro avg       0.95      0.98      0.97    148207
weighted avg       0.98      0.98      0.98    148207

Confusion Matrix with: Max Depth: 4, 0.3 test data
```



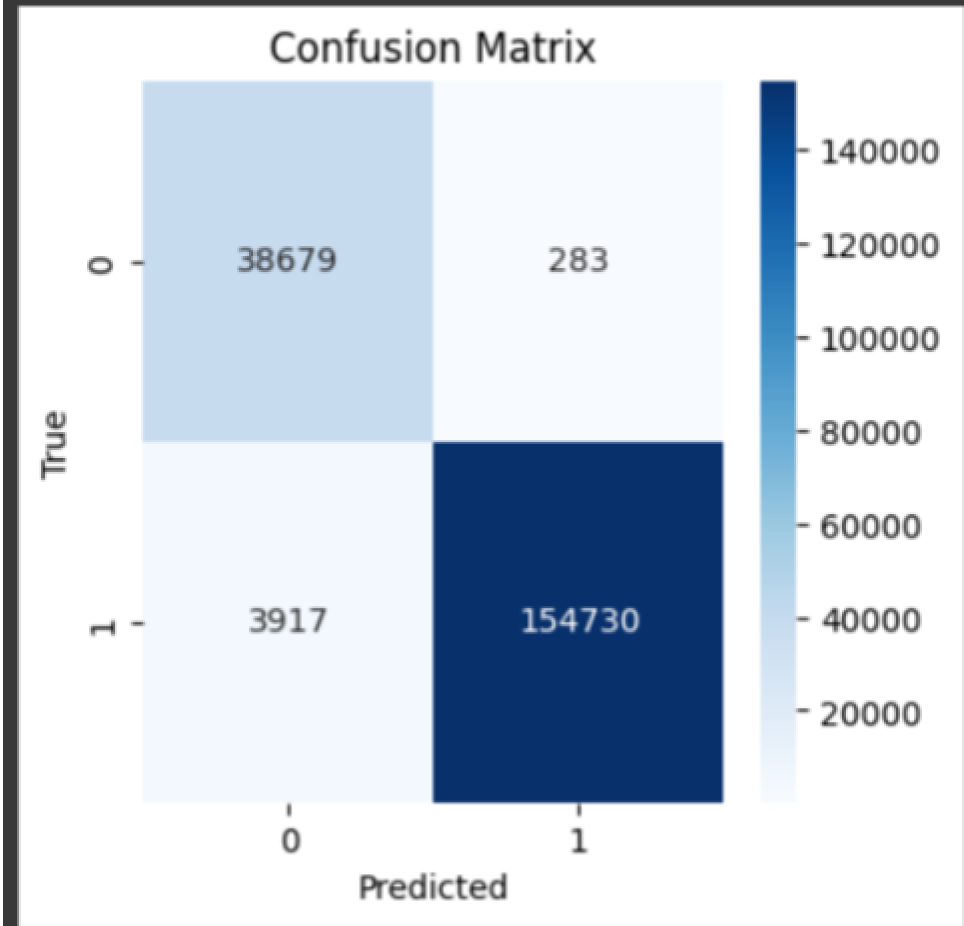
```
my_data_1, Max Depth: 6, Test Ratio: 0.3
Accuracy: 0.9910665488134839
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     29163
           1       0.99      0.99      0.99    119044

    accuracy                           0.99    148207
   macro avg       0.99      0.99      0.99    148207
weighted avg       0.99      0.99      0.99    148207

Confusion Matrix with: Max Depth: 6, 0.3 test data
```



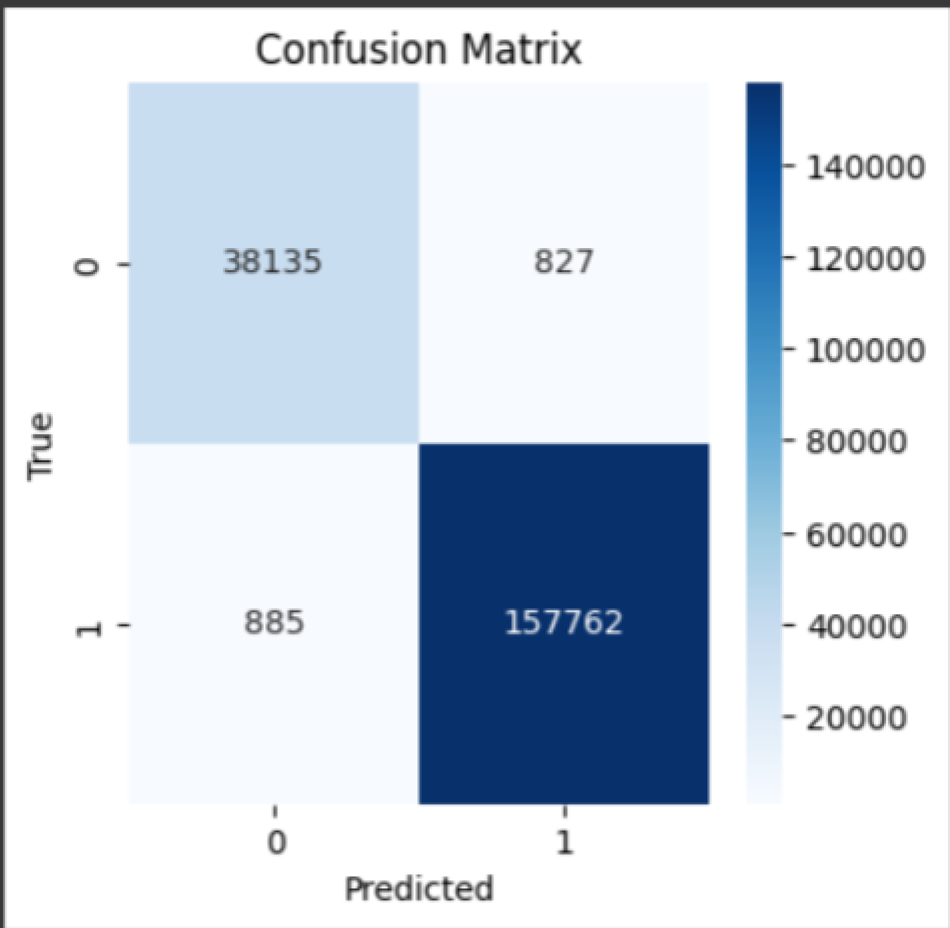
```
my_data_1, Max Depth: 8, Test Ratio: 0.3
Accuracy: 0.9966533294648701
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     29163
           1       1.00      1.00      1.00    119044

    accuracy                           1.00    148207
   macro avg       0.99      1.00      0.99    148207
weighted avg       1.00      1.00      1.00    148207

Confusion Matrix with: Max Depth: 8, 0.3 test data
```



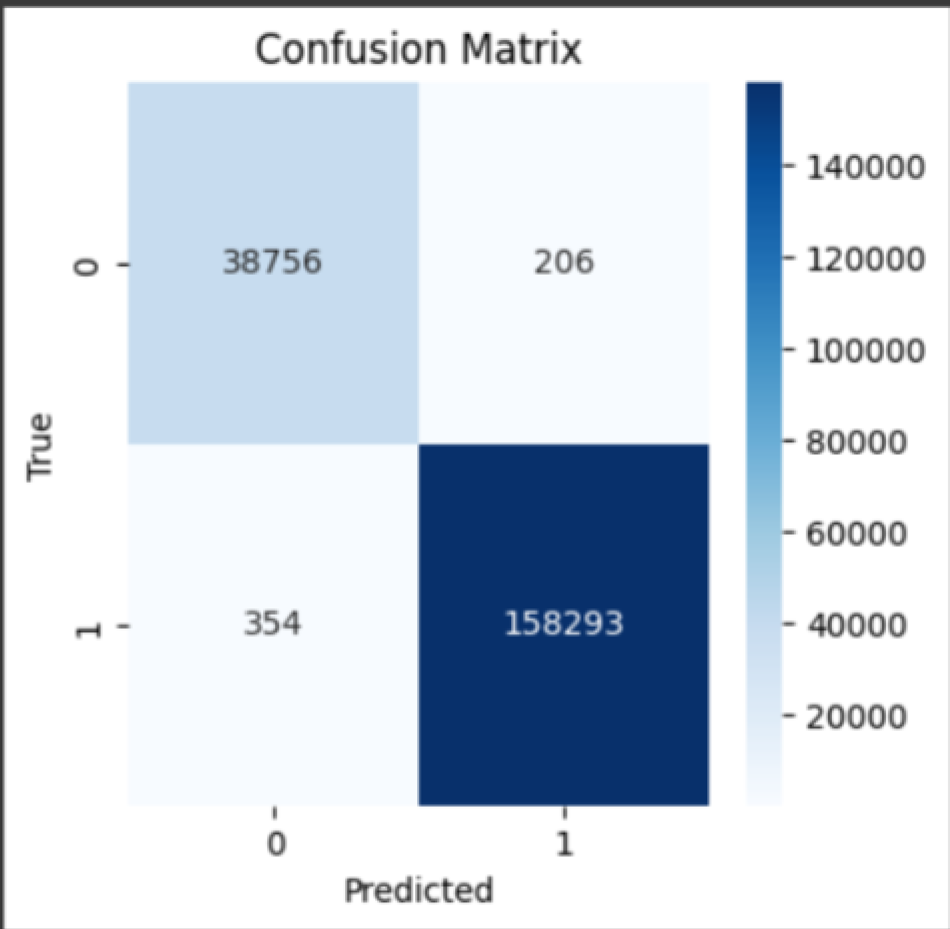
```
my_data_2, Max Depth: 4, Test Ratio: 0.4
Accuracy: 0.9787459073220349
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     38962
           1       1.00      0.98      0.99    158647

    accuracy                           0.98    197609
   macro avg       0.95      0.98      0.97    197609
weighted avg       0.98      0.98      0.98    197609

Confusion Matrix with: Max Depth: 4, 0.4 test data
```



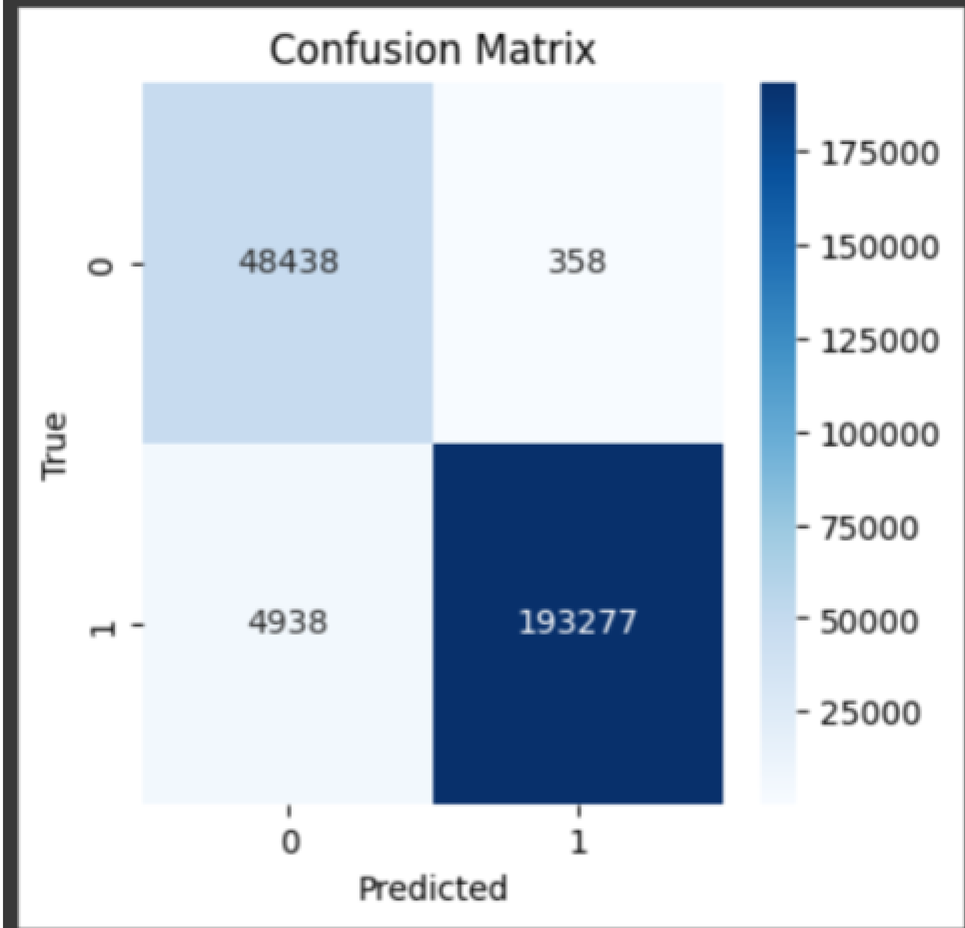
```
my_data_2, Max Depth: 6, Test Ratio: 0.4
Accuracy: 0.9913364269846009
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     38962
           1       0.99      0.99      0.99    158647

    accuracy                           0.99    197609
   macro avg       0.99      0.99      0.99    197609
weighted avg       0.99      0.99      0.99    197609

Confusion Matrix with: Max Depth: 6, 0.4 test data
```



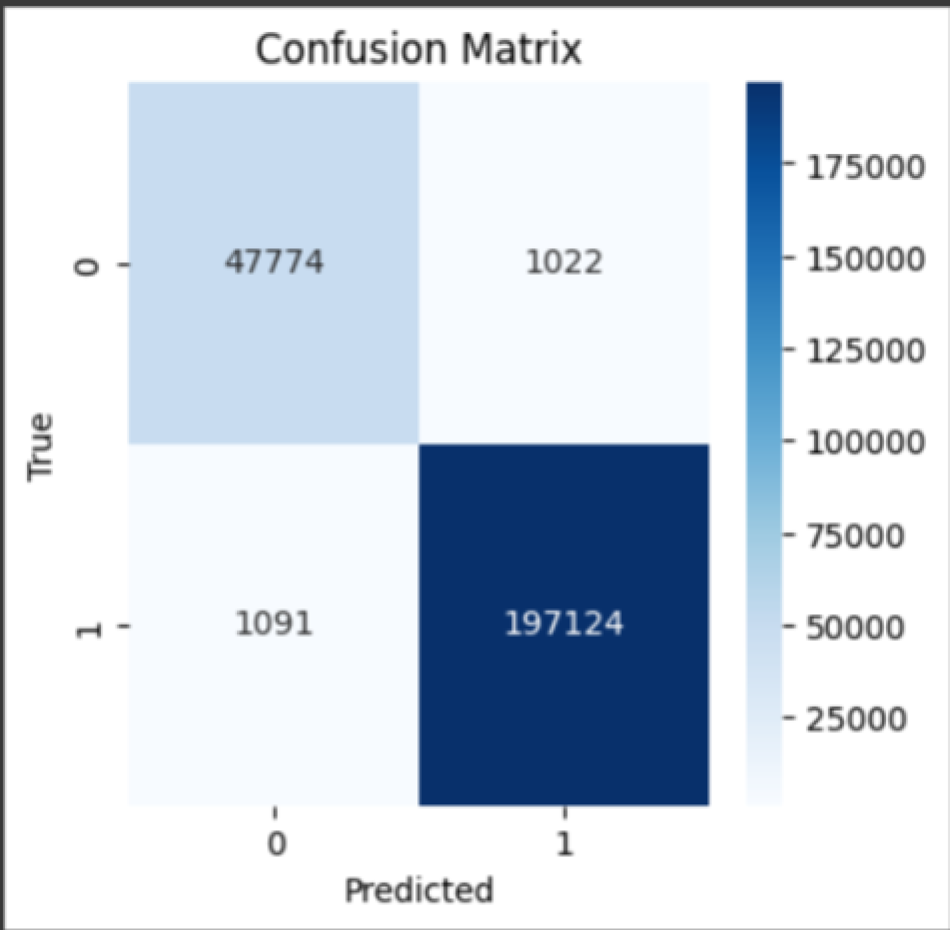
```
my_data_2, Max Depth: 8, Test Ratio: 0.4
Accuracy: 0.9971661209762713
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     38962
           1       1.00      1.00      1.00    158647

    accuracy                           1.00    197609
   macro avg       0.99      1.00      1.00    197609
weighted avg       1.00      1.00      1.00    197609

Confusion Matrix with: Max Depth: 8, 0.4 test data
```



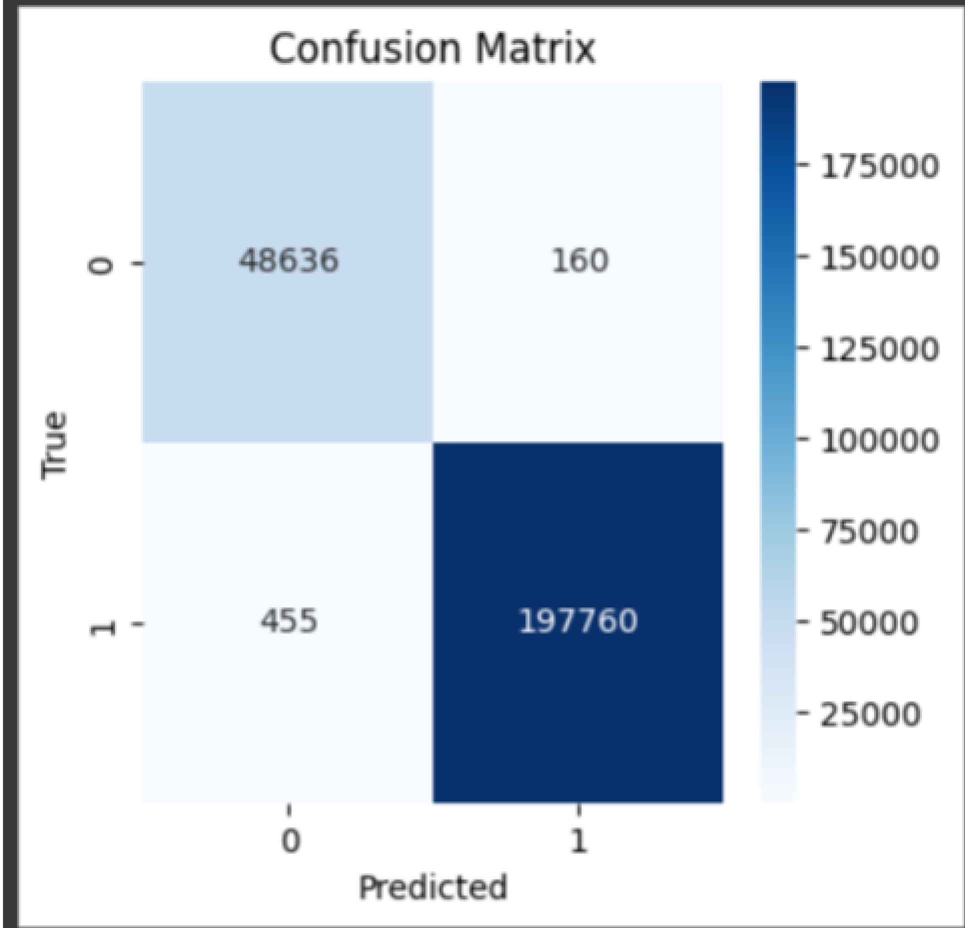
```
my_data_3, Max Depth: 4, Test Ratio: 0.5
Accuracy: 0.9785596592864285
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     48796
           1       1.00      0.98      0.99    198215

    accuracy                           0.98    247011
   macro avg       0.95      0.98      0.97    247011
weighted avg       0.98      0.98      0.98    247011

Confusion Matrix with: Max Depth: 4, 0.5 test data
```



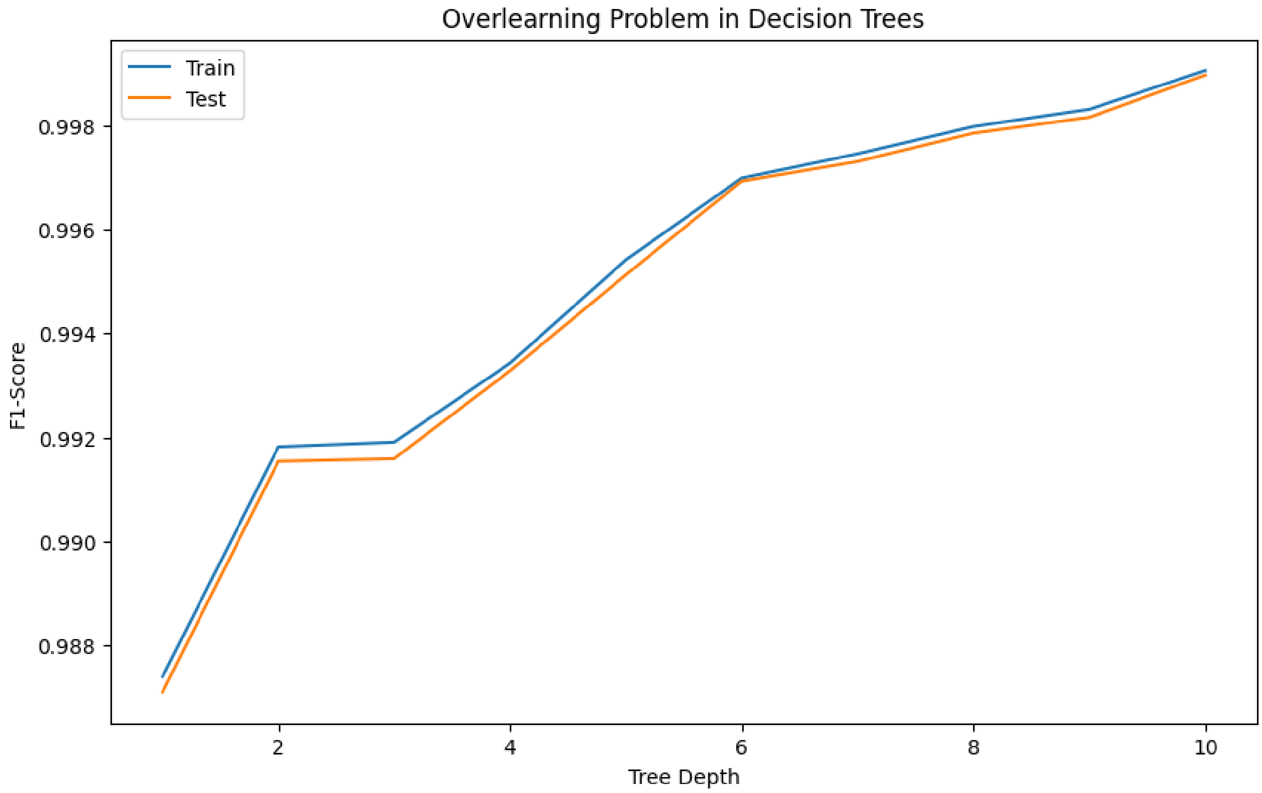
```
my_data_3, Max Depth: 6, Test Ratio: 0.5
Accuracy: 0.9914457250891661
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98     48796
           1       0.99      0.99      0.99    198215

    accuracy                           0.99    247011
   macro avg       0.99      0.99      0.99    247011
weighted avg       0.99      0.99      0.99    247011

Confusion Matrix with: Max Depth: 6, 0.5 test data
```

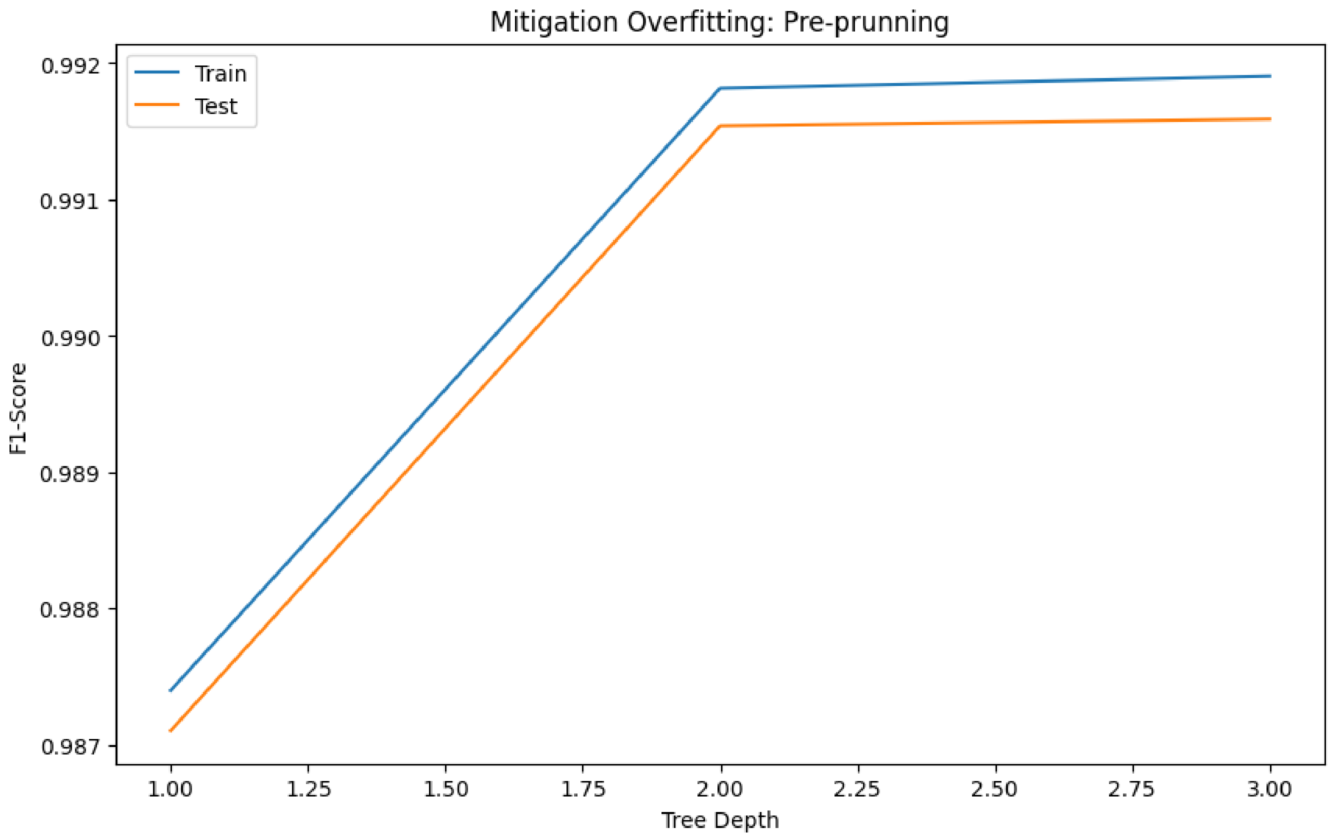


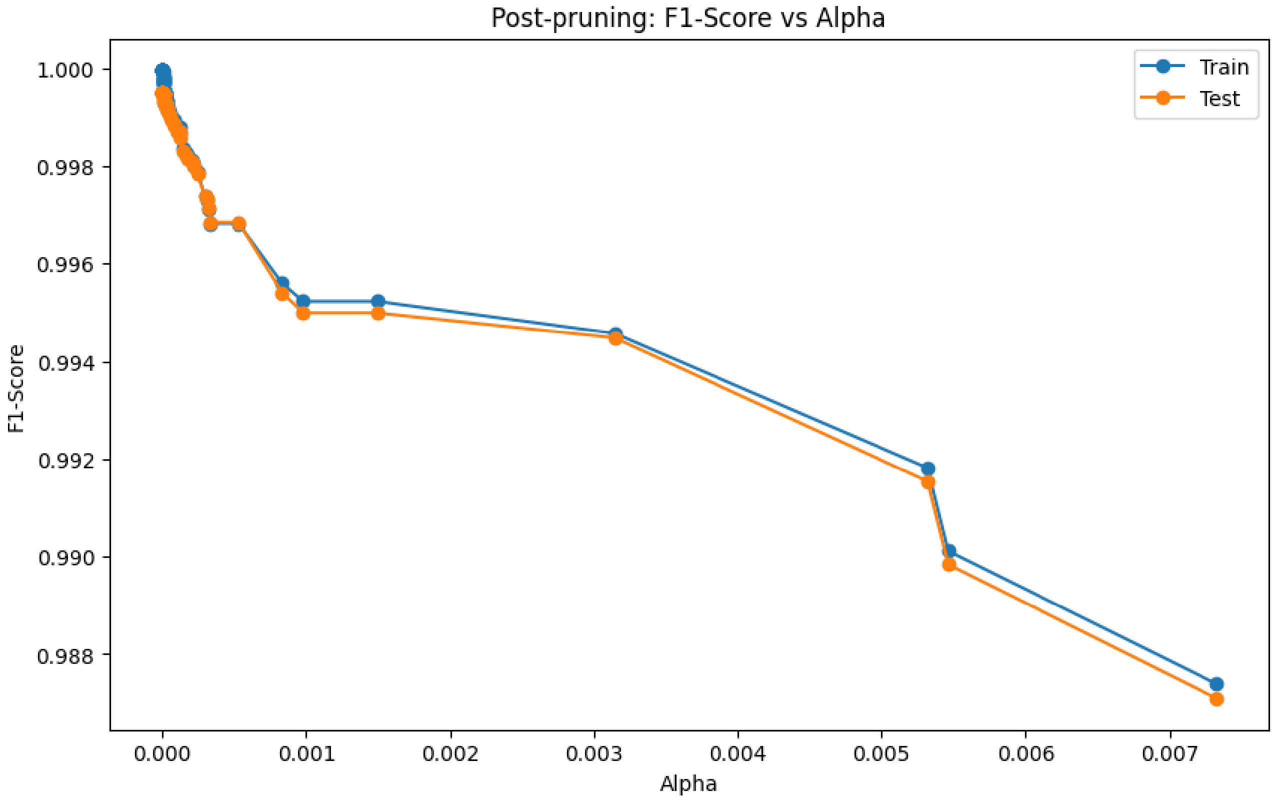
```
my_data_3, Max Depth: 8, Test Ratio: 0.5
Accuracy: 0.9975102323378311
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     48796
           1       1.00      1.00      1.00    198215

    accuracy                           1.00    247011
   macro avg       0.99      1.00      1.00    247011
weighted avg       1.00      1.00      1.00    247011

Confusion Matrix with: Max Depth: 8, 0.5 test data
```



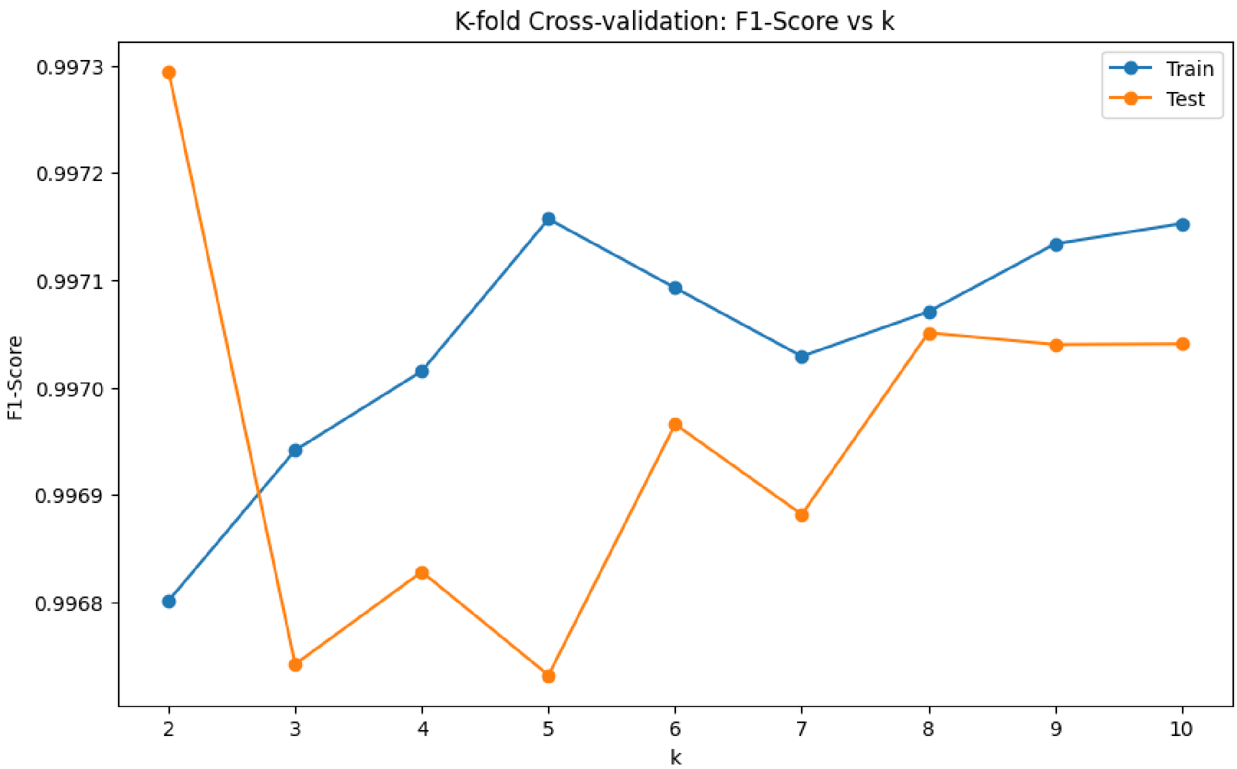
(d) Before mitigation, I didn’t find overfitting problem although I tried several parameters for Decision Tree algorithm. But I had do the task as explained and perform the 3 mitigation strategies as asked.







Post-pruning: F1-Score vs Alpha
Train
Test
F1-Score
Alpha
1.000
0.998
0.996
0.994
0.992
0.990
0.988
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007



K-fold Cross-validation: F1-Score vs k
Train
Test
F1-Score
k
0.9973
0.9972
0.9971
0.9970
0.9969
0.9968
2
3
4
5
6
7
8
9
10