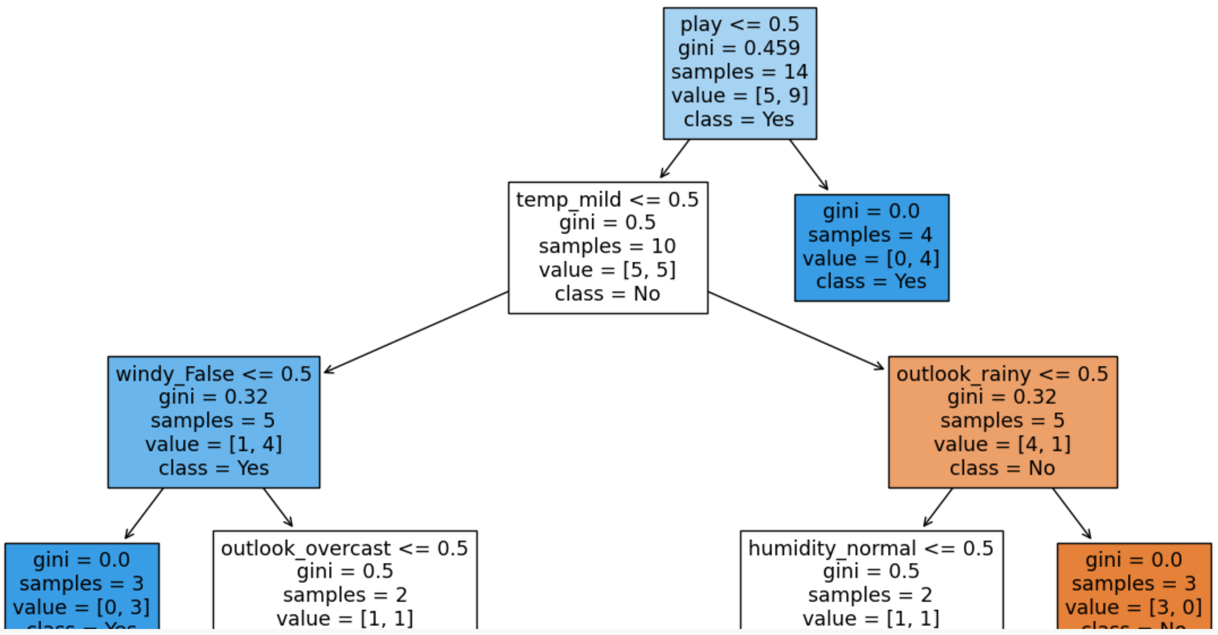
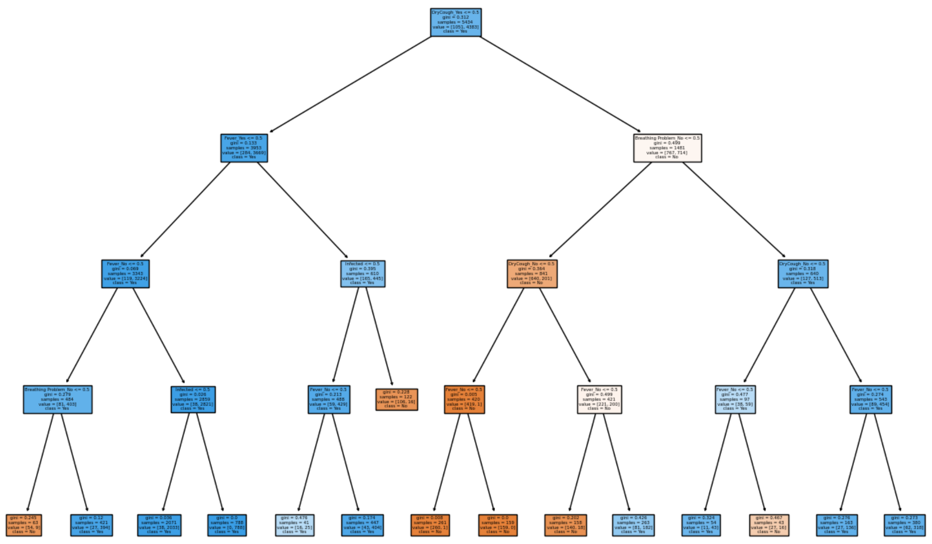
```
import pandas as pd
import numpy as np
import math
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt


data = pd.read_csv('..//content/PlayTennis.csv')

def find_entropy(data):
    """
    Returns the entropy of the class or features
    formula: - ∑ P(X)logP(X)
    """
    entropy = 0
    for i in range(data.nunique()):
        x = data.value_counts()[i]/data.shape[0]
        entropy += (- x * math.log(x,2))
    return round(entropy,3)

def information_gain(data, data_):
    """
    Returns the information gain of the features
    """
    info = 0
    for i in range(data_.nunique()):
        df = data[data_ == data_.unique()[i]]
        w_avg = df.shape[0]/data.shape[0]
        entropy = find_entropy(df.AllergicReaction)
        x = w_avg * entropy
        info += x
    ig = find_entropy(data.AllergicReaction) - info
    return round(ig, 3)

def entropy_and_infogain(datax, feature):
    """
    Grouping features with the same class and computing their
    entropy and information gain for splitting
    """
    for i in range(data[feature].nunique()):
        df = datax[datax[feature]==data[feature].unique()[i]]
        if df.shape[0] < 1:
            continue

        print(f'Entropy of {feature} - {data[feature].unique()[i]} = {find_entropy(df.play)}')
    print(f'Information Gain for {feature} = {information_gain(datax, datax[feature])}')

print(f'Entropy of the entire dataset: {find_entropy(data.play)}')

# entropy_and_infogain(data, 'Restaurant')

# entropy_and_infogain(data, 'Meal')

# entropy_and_infogain(data, 'Day')

# entropy_and_infogain(data, 'Cost')

# entropy_and_infogain(data, 'Vegan')

# sunny = data[data['outlook'] == 'sunny']

# print(f'Entropy of the Sunny dataset: {find_entropy(sunny.play)}')

# entropy_and_infogain(sunny, 'temp')

# entropy_and_infogain(sunny, 'humidity')

# entropy_and_infogain(sunny, 'windy')

# rainy = data[data['outlook'] == 'rainy']

# print(f'Entropy of the Rainy dataset: {find_entropy(rainy.play)}')
# entropy_and_infogain(rainy, 'temp')

# entropy_and_infogain(rainy, 'humidity')

# entropy_and_infogain(rainy, 'windy')

# One-hot encode categorical columns
categorical_cols = ['outlook', 'temp', 'humidity', 'windy']
encoder = OneHotEncoder()
transformed_data = encoder.fit_transform(data[categorical_cols])
encoded_cols = encoder.get_feature_names_out(categorical_cols)
transformed_df = pd.DataFrame(transformed_data.toarray(), columns=encoded_cols)

# Combine encoded data with original data
data_encoded = pd.concat([data.drop(columns=categorical_cols), transformed_df], axis=1)

# Fit decision tree classifier
X = data_encoded.drop('play', axis=1)
y = data_encoded['play']
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Plot the decision tree
plt.figure(figsize=(15,10))
plot_tree(clf, filled=True, feature_names=data_encoded.columns[:-1], class_names=['No', 'Yes'])
plt.show()
```

```
Entropy of the entire dataset: 0.94
```



```python
import pandas as pd
import numpy as np
import math
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt


data = pd.read_csv('..//content/Infection.csv')

def find_entropy(data):
    """
    Returns the entropy of the class or features
    formula: - ∑ P(X)logP(X)
    """
    entropy = 0
    for i in range(data.nunique()):
        x = data.value_counts()[i]/data.shape[0]
        entropy += (- x * math.log(x,2))
    return round(entropy,3)

def information_gain(data, data_):
    """
    Returns the information gain of the features
    """
    info = 0
    for i in range(data_.nunique()):
        df = data[data_ == data_.unique()[i]]
        w_avg = df.shape[0]/data.shape[0]
        entropy = find_entropy(df.Infected)
        x = w_avg * entropy
        info += x
    ig = find_entropy(data.Infected) - info
    return round(ig, 3)

def entropy_and_infogain(datax, feature):
    """
    Grouping features with the same class and computing their
    entropy and information gain for splitting
    """
    for i in range(data[feature].nunique()):
        df = datax[datax[feature]==data[feature].unique()[i]]
        if df.shape[0] < 1:
            continue

        print(f'Entropy of {feature} - {data[feature].unique()[i]} = {find_entropy(df.Infected)}')
    print(f'Information Gain for {feature} = {information_gain(datax, datax[feature])}')

print(f'Entropy of the entire dataset: {find_entropy(data.Infected)}')

# entropy_and_infogain(data, 'Restaurant')

# entropy_and_infogain(data, 'Meal')

# entropy_and_infogain(data, 'Day')

# entropy_and_infogain(data, 'Cost')

# entropy_and_infogain(data, 'Vegan')

# sunny = data[data['outlook'] == 'sunny']

# print(f'Entropy of the Sunny dataset: {find_entropy(sunny.play)}')

# entropy_and_infogain(sunny, 'temp')

# entropy_and_infogain(sunny, 'humidity')

# entropy_and_infogain(sunny, 'windy')

# rainy = data[data['outlook'] == 'rainy']

# print(f'Entropy of the Rainy dataset: {find_entropy(rainy.play)}')
# entropy_and_infogain(rainy, 'temp')

# entropy_and_infogain(rainy, 'humidity')

# entropy_and_infogain(rainy, 'windy')

# One-hot encode categorical columns
categorical_cols = ['Breathing Problem', 'Fever', 'DryCough', 'SoreThroat']
encoder = OneHotEncoder()
transformed_data = encoder.fit_transform(data[categorical_cols])
encoded_cols = encoder.get_feature_names_out(categorical_cols)
transformed_df = pd.DataFrame(transformed_data.toarray(), columns=encoded_cols)

# Combine encoded data with original data
data_encoded = pd.concat([data.drop(columns=categorical_cols), transformed_df], axis=1)

# Fit decision tree classifier
X = data_encoded.drop('Infected', axis=1)
y = data_encoded['Infected']
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Plot the decision tree
plt.figure(figsize=(15,10))
plot_tree(clf, filled=True, feature_names=data_encoded.columns[:-1], class_names=['No', 'Yes'])
plt.show()
```

```
Entropy of the entire dataset: 0.709
```


DryCough_Yes <= 0.5 gini = 0.312 samples = 5434 value = [1051, 4383] class = Yes
Fever_Yes <= 0.5 gini = 0.133 samples = 3953 value = [284, 3669] class = Yes
Breathing Problem_No <= 0.5 gini = 0.499 samples = 1481 value = [767, 714] class = No
Fever_No <= 0.5 gini = 0.069 samples = 3343 value = [119, 3224] class = Yes
Infected <= 0.5 gini = 0.395 samples = 610 value = [165, 445] class = Yes
DryCough_No <= 0.5 gini = 0.364 samples = 841 value = [640, 201] class = No
DryCough_No <= 0.5 gini = 0.318 samples = 640 value = [127, 513] class = Yes
Breathing Problem_No <= 0.5 gini = 0.279 samples = 484 value = [81, 403] class = Yes
Infected <= 0.5 gini = 0.026 samples = 2859 value = [38, 2821] class = Yes
Fever_No <= 0.5 gini = 0.213 samples = 488 value = [59, 429] class = Yes
gini = 0.228 samples = 122 value = [106, 16] class = No
Fever_No <= 0.5 gini = 0.005 samples = 420 value = [419, 1] class = No
Fever_No <= 0.5 gini = 0.499 samples = 421 value = [221, 200] class = No
Fever_No <= 0.5 gini = 0.477 samples = 97 value = [38, 59] class = Yes
Fever_No <= 0.5 gini = 0.274 samples = 543 value = [89, 454] class = Yes
gini = 0.245 samples = 63 value = [54, 9] class = No
gini = 0.12 samples = 421 value = [27, 394] class = Yes
gini = 0.036 samples = 2071 value = [38, 2033] class = Yes
gini = 0.0 samples = 788 value = [0, 788] class = Yes
gini = 0.476 samples = 41 value = [16, 25] class = Yes
gini = 0.174 samples = 447 value = [43, 404] class = Yes
gini = 0.008 samples = 261 value = [260, 1] class = No
gini = 0.0 samples = 159 value = [159, 0] class = No
gini = 0.202 samples = 158 value = [140, 18] class = No
gini = 0.426 samples = 263 value = [81, 182] class = Yes
gini = 0.324 samples = 54 value = [11, 43] class = Yes
gini = 0.467 samples = 43 value = [27, 16] class = No
gini = 0.276 samples = 163 value = [27, 136] class = Yes
gini = 0.273 samples = 380 value = [62, 318] class = Yes