DECISION TREE CLASSIFIER

STEP-1 IMPORT DATA

```
import pandas as pd
df = pd.read_csv("mldata1.csv")
df.head()
```

	age	height	weight	gender	likeness
0	27	170.688	76.0	Male	Biryani
1	41	165	70.0	Male	Biryani
2	29	171	80.0	Male	Biryani
3	27	173	102.0	Male	Biryani

Step-2 Making input and Output Variable

```
df["gender"] = df["gender"].replace("Male",1)
df["gender"] = df["gender"].replace("Female",0)
```

```
X = df[["weight","gender"]]
y = df["likeness"]
```

Step-3 Making Machine Learning Model

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier().fit(X,y)
model.predict([[50,1]])
```

Saving...

```
-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
array(['Samosa'], dtype=object)
```

Step-4 Checking machine learning model performance

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
model = DecisionTreeClassifier().fit(X_train,y_train)
predicted_values = model.predict(X_test)
predicted_values
```

```
array(['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Samosa', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Samosa',
       'Biryani', 'Biryani', 'Biryani', 'Pakora', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Pakora', 'Biryani', 'Biryani', 'Biryani',
       'Biryani'], dtype=object)
```

```
score = accuracy_score(y_test, predicted_values)
score
```

```
0.6326530612244898
```

Step-5 Making Visualization

```
from sklearn import tree
model = DecisionTreeClassifier().fit(X,y)
tree.export_graphviz(model,out_file= "foodie.dot",
feature_names=["age","gender"],
class_names=sorted(y.unique()),
label="all",rounded=True,filled=True)
```

```
from sklearn import tree
from sklearn.datasets import load_iris
import graphviz

# Create a sample dataset (replace this with your own data)
X = [[30, 0], [25, 1], [35, 1], [40, 0]]
y = ['No', 'Yes', 'Yes', 'No']
```

Saving...

```
model = tree.DecisionTreeClassifier()
model.fit(X, y)

# Export the decision tree as a DOT file
dot_data = tree.export_graphviz(
    model,
    out_file=None,
    feature_names=["age", "gender"],
    class_names=sorted(set(y)),
    label="all",
    rounded=True,
    filled=True
)

# Save the DOT file
with open("foodie.dot", "w") as f:
    f.write(dot_data)

# Convert the DOT file to a visual representation (e.g., PDF, PNG, or SVG)
graph = graphviz.Source(dot_data)
graph.render(filename="foodie", format="pdf")
```

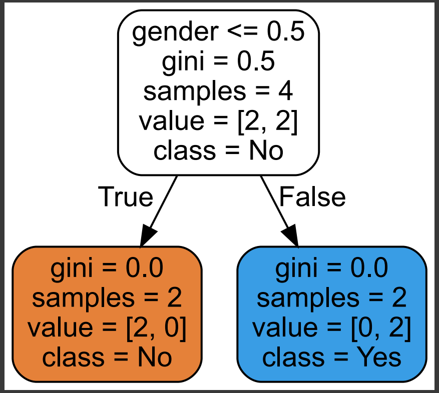
```
'foodie.pdf'
```

```
import graphviz
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

# Create a sample dataset
X = [[30, 0], [25, 1], [35, 1], [40, 0]]
y = ['No', 'Yes', 'Yes', 'No']

# Train the decision tree model
model = DecisionTreeClassifier()
model.fit(X, y)

# Export the decision tree as a DOT file
dot_data = tree.export_graphviz(
    model,
    out_file=None,
    feature_names=["age", "gender"],
    class_names=sorted(set(y)),
    label="all",
    rounded=True,
    filled=True
)

# C       DOT fil
```

Saving...

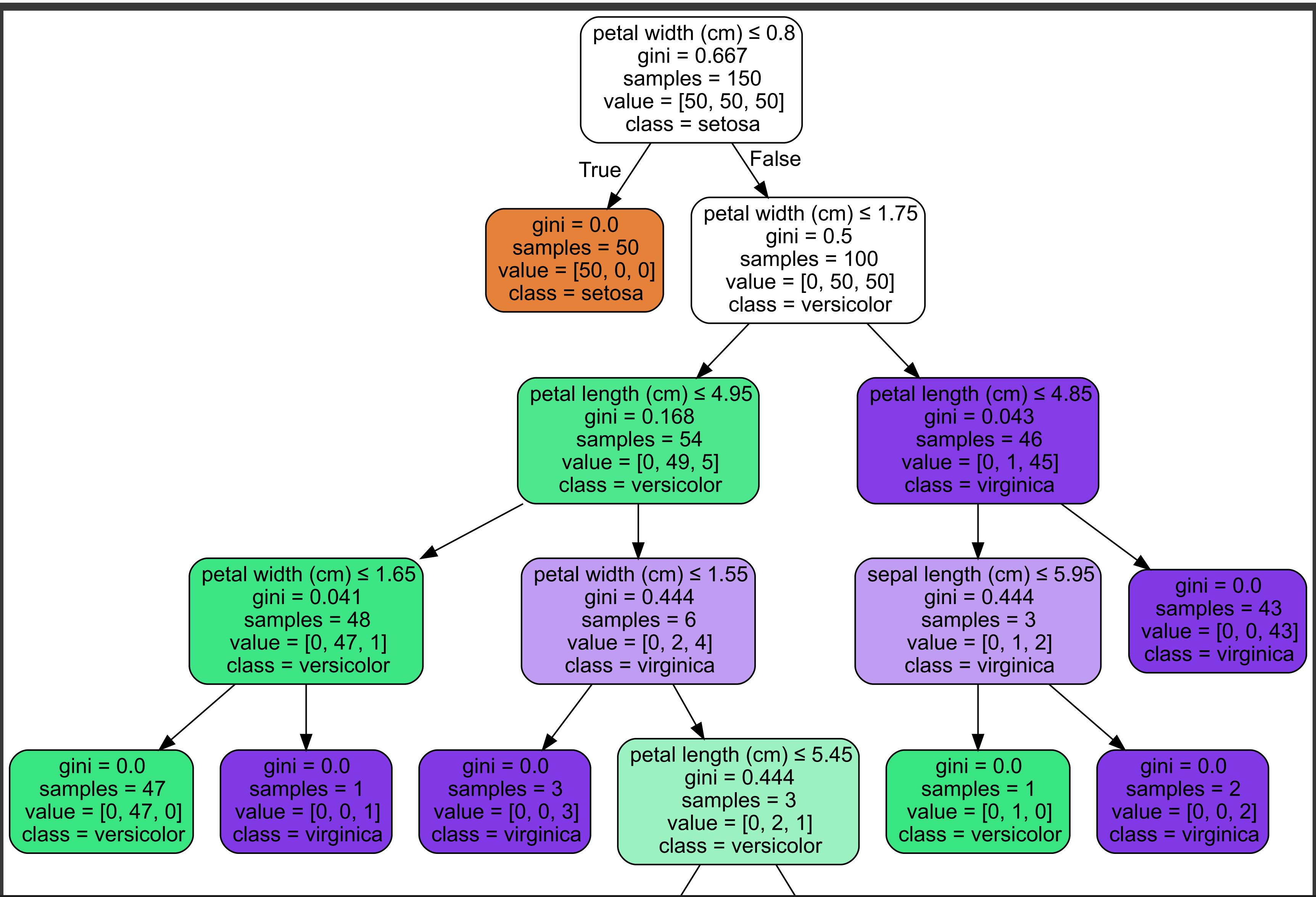
```
# Convert the DOT file to a visual representation
graph = graphviz.Source(dot_data)
graph.render("foodie", format="png")
```

```
# Display the decision tree
graph
```



```
from sklearn import tree
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import graphviz

# Load the dataset
iris = load_iris()
X = iris.data
y = iris.target

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=iris.feature_names,
                               class_names=iris.target_names,
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```

Saving... ✕



```
import pandas as pd

                  eeClassifier
import graphviz

# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29],
```

Saving...

```
    'height': [170.688, 165, 171, 173, 164],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=X.columns,
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-df4453425eb0> in <cell line: 22>()
     20 
     21 # Train the decision tree model
---> 22 model = DecisionTreeClassifier().fit(X, y)
     23 
     24 # Export the decision tree visualization to a .dot file

                          5 frames

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __array__(self, dtype)
   2068 
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071 
   2072     def __array_wrap__(

                                        string to float: 'Male'
```

Saving...

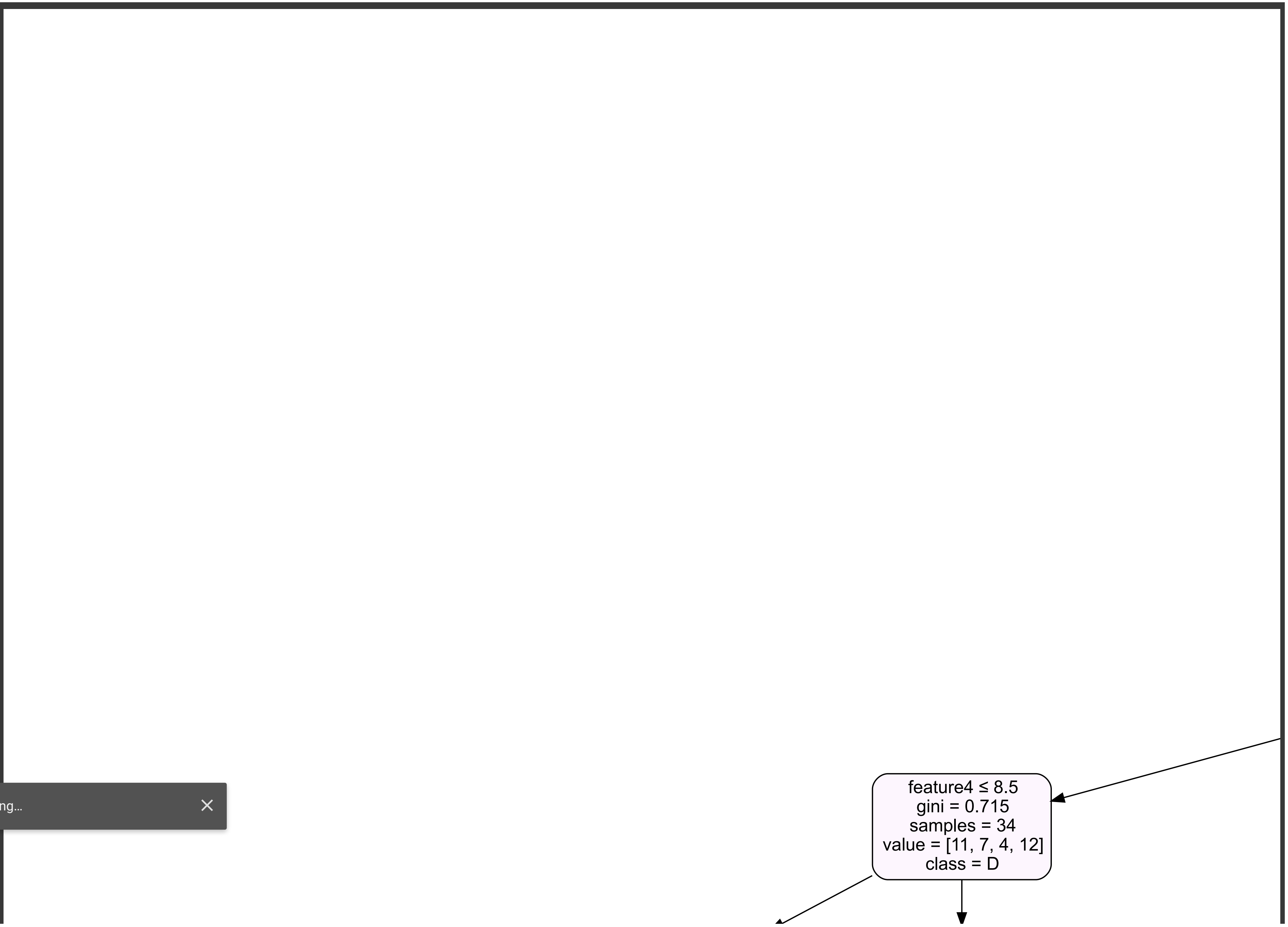
```
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import graphviz

# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29],
    'height': [170.688, 165, 171, 173, 164],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Perform one-hot encoding for the 'gender' feature
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), ['gender'])],
    remainder='passthrough'
)
X = ct.fit_transform(X)

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=ct.get_feature_names_out(),
                                class_names=y.unique(),
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```

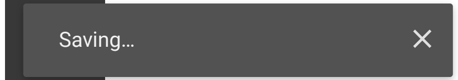
Saving...

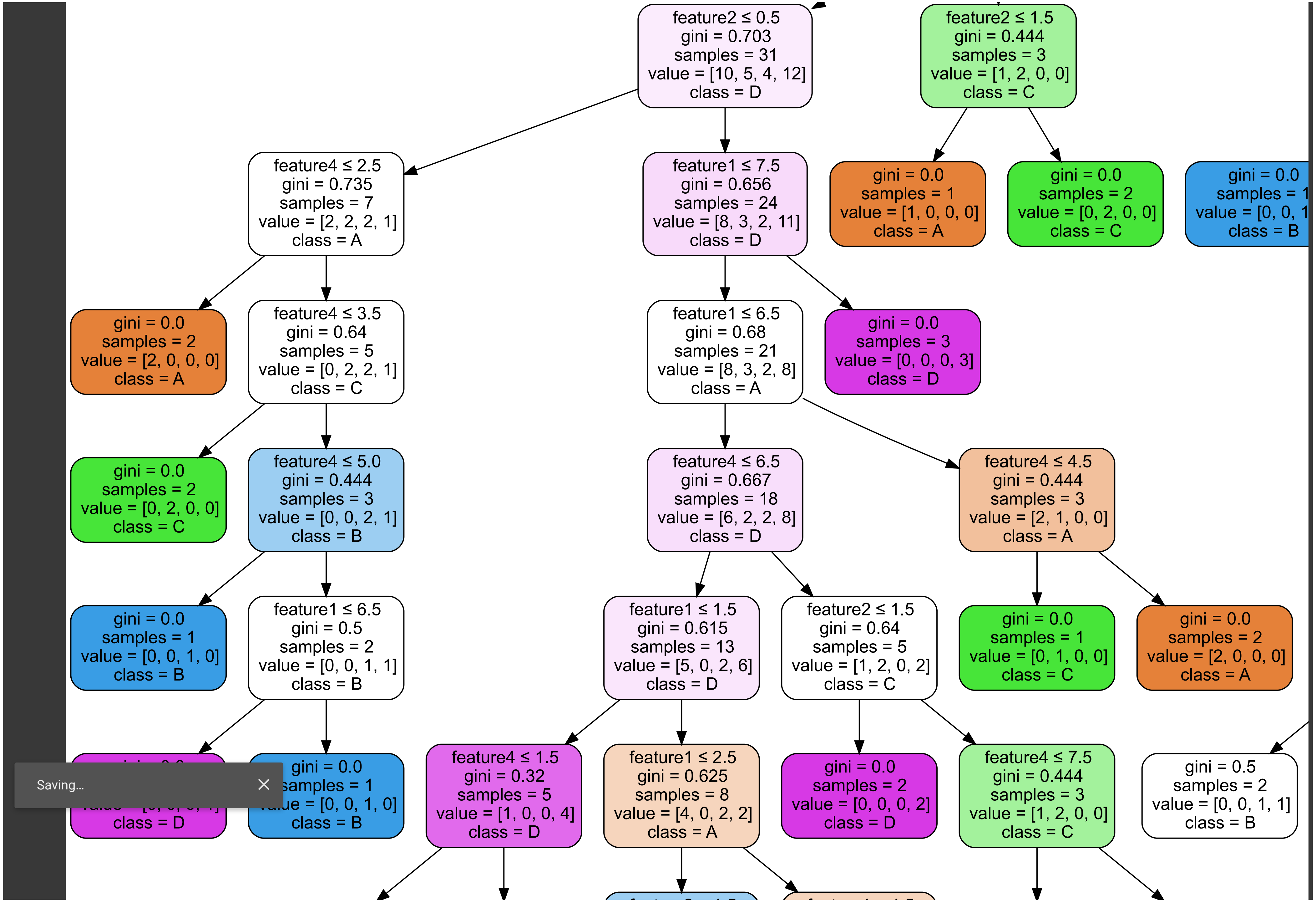
```
import numpy as np
import pandas as pd
```

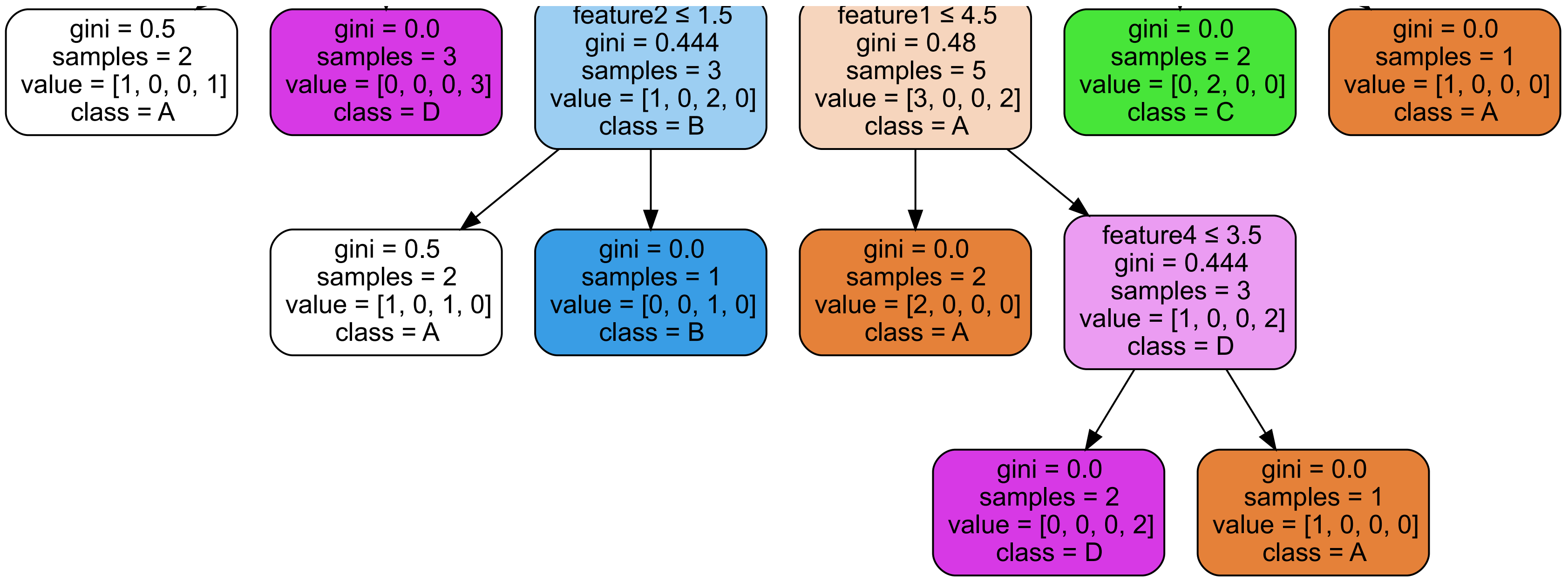
```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import graphviz

# Generate random data
np.random.seed(0)
n_samples = 1000

data = {
    'feature1': np.random.randint(0, 10, n_samples),
    'feature2': np.random.randint(0, 10, n_samples),
    'feature3': np.random.randint(0, 10, n_samples),
    'feature4': np.random.randint(0, 10, n_samples),
    'target': np.random.choice(['A', 'B', 'C', 'D'], n_samples)
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df.drop('target', axis=1)
y = df['target']

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=X.columns,
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("big_tree")  # Save the visualization as "big_tree.pdf"

# Display the decision tree visualization
graph
```

Saving... ✕


feature4 ≤ 8.5
gini = 0.715
samples = 34
value = [11, 7, 4, 12]
class = D


Saving...

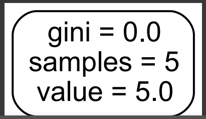

feature2 ≤ 0.5
gini = 0.703
samples = 31
value = [10, 5, 4, 12]
class = D
feature2 ≤ 1.5
gini = 0.444
samples = 3
value = [1, 2, 0, 0]
class = C
feature4 ≤ 2.5
gini = 0.735
samples = 7
value = [2, 2, 2, 1]
class = A
feature1 ≤ 7.5
gini = 0.656
samples = 24
value = [8, 3, 2, 11]
class = D
gini = 0.0
samples = 1
value = [1, 0, 0, 0]
class = A
gini = 0.0
samples = 2
value = [0, 2, 0, 0]
class = C
gini = 0.0
samples = 1
value = [0, 0, 1
class = B
gini = 0.0
samples = 2
value = [2, 0, 0, 0]
class = A
feature4 ≤ 3.5
gini = 0.64
samples = 5
value = [0, 2, 2, 1]
class = C
feature1 ≤ 6.5
gini = 0.68
samples = 21
value = [8, 3, 2, 8]
class = A
gini = 0.0
samples = 3
value = [0, 0, 0, 3]
class = D
gini = 0.0
samples = 2
value = [0, 2, 0, 0]
class = C
feature4 ≤ 5.0
gini = 0.444
samples = 3
value = [0, 0, 2, 1]
class = B
feature4 ≤ 6.5
gini = 0.667
samples = 18
value = [6, 2, 2, 8]
class = D
feature4 ≤ 4.5
gini = 0.444
samples = 3
value = [2, 1, 0, 0]
class = A
gini = 0.0
samples = 1
value = [0, 0, 1, 0]
class = B
feature1 ≤ 6.5
gini = 0.5
samples = 2
value = [0, 0, 1, 1]
class = B
feature1 ≤ 1.5
gini = 0.615
samples = 13
value = [5, 0, 2, 6]
class = D
feature2 ≤ 1.5
gini = 0.64
samples = 5
value = [1, 2, 0, 2]
class = C
gini = 0.0
samples = 1
value = [0, 1, 0, 0]
class = C
gini = 0.0
samples = 2
value = [2, 0, 0, 0]
class = A
class = D
gini = 0.0
samples = 1
value = [0, 0, 1, 0]
class = B
feature4 ≤ 1.5
gini = 0.32
samples = 5
value = [1, 0, 0, 4]
class = D
feature1 ≤ 2.5
gini = 0.625
samples = 8
value = [4, 0, 2, 2]
class = A
gini = 0.0
samples = 2
value = [0, 0, 0, 2]
class = D
feature4 ≤ 7.5
gini = 0.444
samples = 3
value = [1, 2, 0, 0]
class = C
gini = 0.5
samples = 2
value = [0, 0, 1, 1]
class = B
Saving...



Saving...

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import graphviz
```

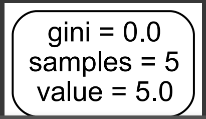
```
# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29],
    'height': [170.688, 165, 171, 173, 164],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Perform one-hot encoding for the 'gender' feature
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), ['gender'])],
    remainder='passthrough'
)
X = ct.fit_transform(X)

# Train the decision tree model with a larger max_depth
max_depth = 5  # Adjust this value to increase/decrease the depth of the tree
model = DecisionTreeClassifier(max_depth=max_depth).fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=ct.get_feature_names_out(),
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```



Saving... ✕

```
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

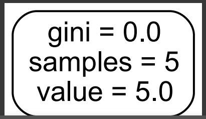
```
import graphviz

# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29],
    'height': [170.688, 165, 171, 173, 164],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Perform one-hot encoding for the 'gender' feature
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), [3])],  # Assuming 'gender' is the 4th column (index 3)
    remainder='passthrough'
)
X = ct.fit_transform(X)

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=ct.get_feature_names_out(),
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```



Saving... ✕

```
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

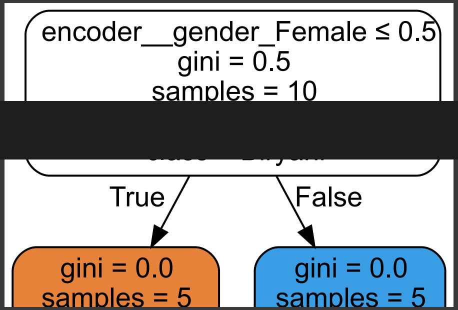
```
import graphviz

# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29],
    'height': [170.688, 165, 171, 173, 164],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Perform one-hot encoding for the 'gender' feature
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), [3])],  # Assuming 'gender' is the 4th column (index 3)
    remainder='passthrough'
)
X = ct.fit_transform(X)

# Train the decision tree model with adjusted parameters
model = DecisionTreeClassifier(max_depth=5, min_samples_split=2, min_samples_leaf=1).fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=ct.get_feature_names_out(),
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```



Saving...

```
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
import graphviz
import graphviz

# Create the dataset
data = {
    'age': [27, 41, 29, 27, 29, 22, 30, 33, 28, 25],
    'height': [170.688, 165, 171, 173, 164, 175, 169, 163, 167, 170],
    'weight': [76.0, 70.0, 80.0, 102.0, 67.0, 68.0, 65.0, 60.0, 75.0, 72.0],
    'gender': ['Male', 'Male', 'Male', 'Male', 'Male', 'Female', 'Female', 'Female', 'Female', 'Female'],
    'likeness': ['Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
                 'Pizza', 'Pizza', 'Pizza', 'Pizza', 'Pizza']
}

df = pd.DataFrame(data)

# Split the dataset into features (X) and target (y)
X = df[['age', 'height', 'weight', 'gender']]
y = df['likeness']

# Perform one-hot encoding for the 'gender' feature
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), [3])],  # Assuming 'gender' is the 4th column (index 3)
    remainder='passthrough'
)
X = ct.fit_transform(X)

# Train the decision tree model
model = DecisionTreeClassifier().fit(X, y)

# Export the decision tree visualization to a .dot file
dot_data = tree.export_graphviz(model, out_file=None,
                               feature_names=ct.get_feature_names_out(),
                               class_names=y.unique(),
                               filled=True, rounded=True,
                               special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("foodie")  # Save the visualization as "foodie.pdf"

# Display the decision tree visualization
graph
```

Saving... ✕



[redacted]

✓ 0s completed at 22:20

Saving...