Assignment 8

Title: Decision Tree

Name: Pranjal Rane

NUID: 002756852

Split the dataset into 60% training set and 40% test set.

In [1]:
```
import pandas as pd
columns = ['variance','skewness','curtosis','entropy']
```

In [2]:
```
df = pd.read_csv('data_banknote_authentication.txt',sep=',',names=columns)
```

In [3]:
```
class_name = 'entropy'
feature_names = [name for name in columns if name!=class_name]
X = df[feature_names]
y = df[class_name]
```

In [4]:
```
X
```

Out[4]:

	variance	skewness	curtosis
3.62160	8.66610	-2.8073	-0.44699
4.54590	8.16740	-2.4586	-1.46210
3.86600	-2.63830	1.9242	0.10645
3.45660	9.52280	-4.0112	-3.59440
0.32924	-4.45520	4.5718	-0.98880
...	...	...	...
0.40614	1.34920	-1.4501	-0.55949
-1.38870	-4.87730	6.4774	0.34179
-3.75030	-13.45860	17.5932	-2.77710
-3.56370	-8.38270	12.3930	-1.28230
-2.54190	-0.65804	2.6842	1.19520

1372 rows × 3 columns

In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.40, randor
```

In [6]:
```
X_train
```

Out[6]:

	variance	skewness	curtosis
5.02970	-4.97040	3.50250	-0.237510
-2.37970	-1.44020	1.12730	0.160760
-1.66620	-0.30005	1.42380	0.024986
0.11592	3.22190	-3.43020	-2.845700
6.09190	2.96730	-1.32670	1.455100
...	...	...	...
1.16400	3.91300	-4.55440	-3.867200
-2.29180	-7.25700	7.95970	0.921100
-7.03640	9.29310	0.16594	-4.539600
-3.46050	2.69010	0.16165	-1.022400
-3.35820	-7.24040	11.44190	-0.571130

823 rows × 3 columns

Using scikit-learn's DecisionTreeClassifier, train a supervised learning model that can be used to generate predictions for your data.

In [7]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train,y_train)
```

Out[7]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

Report the tree depth, number of leaves, feature importance, train score, and test score of the tree.

In [8]:

```
print(f"Number of nodes in tree : = {clf.tree_.node_count}\n")
print(f"Max depth of the tree = {clf.tree_.max_depth}\n")
feature_importance = {column:feature_importance_score for column,feature_impor
print(f"Feature importance of columns in tree :\n{feature_importance}\n")
print(f"Train score of tree: {clf.score(X_train,y_train)}\n")
print(f"Test score of tree: {clf.score(X_test,y_test)}\n")
```

```
Number of nodes in tree : = 121

Max depth of the tree = 11

Feature importance of columns in tree :
{'variance': 0.6134501333866513, 'skewness': 0.2003914035109112, 'curtosis':
0.18615846310243747}

Train score of tree: 1.0

Test score of tree: 0.9107468123861566
```

Now you will generate decision trees on the same training set using fixed tree depths. The tree depth can be set using max=d, where d is the depth of the tree.

Decrease depth from the decision tree in Step 2, and for every depth (from max depth to depth 1), report tree depth, number of leaves, feature importance, train score, and test score of the tree.

In [9]:

```
import numpy as np
max_depth = clf.tree_.max_depth
depths = np.arange(1, max_depth + 1)

tree_depths = []
num_leaves_list = []
feature_importances = []
train_scores = []
test_scores = []

for depth in depths:
    fixed_tree = DecisionTreeClassifier(max_depth=depth,random_state=42)
    fixed_tree.fit(X_train, y_train)

    tree_depths.append(fixed_tree.get_depth())

    num_leaves_list.append(fixed_tree.get_n_leaves())

    feature_importances.append(fixed_tree.feature_importances_)

    train_scores.append(fixed_tree.score(X_train, y_train))
    test_scores.append(fixed_tree.score(X_test, y_test))

    print("Tree Depth (Fixed):", depth)
    print(f"Number of nodes in tree {fixed_tree.tree_.node_count}")
    print(f"Max depth of the tree {fixed_tree.tree_.max_depth}")
    feature_importance = {column:feature_importance_score for column,feature_in
    print(f"Feature importance of columns in tree :\n{feature_importance}")
    print(f"Train score of tree: {fixed_tree.score(X_train,y_train)}")
    print(f"Test score of tree: {fixed_tree.score(X_test,y_test)}")
    print()
```

```
Tree Depth (Fixed): 1
Number of nodes in tree 3
Max depth of the tree 1
Feature importance of columns in tree :
{'variance': 1.0, 'skewness': 0.0, 'curtosis': 0.0}
Train score of tree: 0.6974483596597812
Test score of tree: 0.7158469945355191

Tree Depth (Fixed): 2
Number of nodes in tree 7
Max depth of the tree 2
Feature importance of columns in tree :
{'variance': 0.5939685953474626, 'skewness': 0.31357184081621264, 'curtosis':
0.09245956383632468}
Train score of tree: 0.732685297691373
Test score of tree: 0.73224043715847

Tree Depth (Fixed): 3
Number of nodes in tree 15
Max depth of the tree 3
Feature importance of columns in tree :
{'variance': 0.6907456742365798, 'skewness': 0.20078679192070073, 'curtosis':
0.10846753384271939}
Train score of tree: 0.8651275820170109
Test score of tree: 0.8506375227686703

Tree Depth (Fixed): 4
Number of nodes in tree 29
Max depth of the tree 4
Feature importance of columns in tree :
{'variance': 0.6770750497956859, 'skewness': 0.2273373825380974, 'curtosis':
0.09558756766621658}
Train score of tree: 0.8784933171324423
Test score of tree: 0.8615664845173042

Tree Depth (Fixed): 5
Number of nodes in tree 43
Max depth of the tree 5
Feature importance of columns in tree :
{'variance': 0.6733490242445285, 'skewness': 0.2163992069035024, 'curtosis':
0.11025176885196918}
Train score of tree: 0.9125151883353585
Test score of tree: 0.8761384335154827

Tree Depth (Fixed): 6
Number of nodes in tree 59
Max depth of the tree 6
Feature importance of columns in tree :
{'variance': 0.6420710703463705, 'skewness': 0.1883056959212199, 'curtosis':
0.16962323373240967}
Train score of tree: 0.959902794653706
Test score of tree: 0.8979963570127505

Tree Depth (Fixed): 7
Number of nodes in tree 79
Max depth of the tree 7
Feature importance of columns in tree :
{'variance': 0.6380615853170323, 'skewness': 0.18846717828617351, 'curtosis':
0.17347123639679435}
Train score of tree: 0.9732685297691372
```

```
Test score of tree: 0.9016393442622951

Tree Depth (Fixed): 8
Number of nodes in tree 95
Max depth of the tree 8
Feature importance of columns in tree :
{'variance': 0.6232645504956679, 'skewness': 0.20889415074880974, 'curtosis':
0.16784129875552234}
Train score of tree: 0.9842041312272175
Test score of tree: 0.9089253187613844

Tree Depth (Fixed): 9
Number of nodes in tree 109
Max depth of the tree 9
Feature importance of columns in tree :
{'variance': 0.6113564960778676, 'skewness': 0.20407737572807844, 'curtosis':
0.184566128194054}
Train score of tree: 0.9927095990279465
Test score of tree: 0.9089253187613844

Tree Depth (Fixed): 10
Number of nodes in tree 117
Max depth of the tree 10
Feature importance of columns in tree :
{'variance': 0.619315954659742, 'skewness': 0.2016329335956422, 'curtosis': 0.
1790511117446159}
Train score of tree: 0.9975698663426489
Test score of tree: 0.912568306010929

Tree Depth (Fixed): 11
Number of nodes in tree 121
Max depth of the tree 11
Feature importance of columns in tree :
{'variance': 0.6134501333866513, 'skewness': 0.2003914035109112, 'curtosis':
0.18615846310243747}
Train score of tree: 1.0
Test score of tree: 0.9107468123861566
```

In [10]:

```python
import numpy as np
max_depth = clf.tree_.max_depth
depths = np.arange(1, max_depth + 1)

tree_depths = []
num_leaves_list = []
feature_importances = []
train_scores = []
test_scores = []

for depth in depths:
    fixed_tree = DecisionTreeClassifier(max_depth=depth,random_state=42)
    fixed_tree.fit(X_train, y_train)

    tree_depths.append(fixed_tree.get_depth())

    num_leaves_list.append(fixed_tree.get_n_leaves())

    feature_importances.append(fixed_tree.feature_importances_)

    train_scores.append(fixed_tree.score(X_train, y_train))
```

```
    test_scores.append(fixed_tree.score(X_test, y_test))

    print("Tree Depth (Fixed):", depth)
    print(f"Number of nodes in tree {fixed_tree.tree_.node_count}")
    print(f"Max depth of the tree {fixed_tree.tree_.max_depth}")
    feature_importance = {column:feature_importance_score for column,feature_in
    print(f"Feature importance of columns in tree :\n{feature_importance}")
    print(f"Train score of tree: {fixed_tree.score(X_train,y_train)}")
    print(f"Test score of tree: {fixed_tree.score(X_test,y_test)}")
    print()
```

```
Tree Depth (Fixed): 1
Number of nodes in tree 3
Max depth of the tree 1
Feature importance of columns in tree :
{'variance': 1.0, 'skewness': 0.0, 'curtosis': 0.0}
Train score of tree: 0.6974483596597812
Test score of tree: 0.7158469945355191

Tree Depth (Fixed): 2
Number of nodes in tree 7
Max depth of the tree 2
Feature importance of columns in tree :
{'variance': 0.5939685953474626, 'skewness': 0.31357184081621264, 'curtosis':
0.09245956383632468}
Train score of tree: 0.732685297691373
Test score of tree: 0.73224043715847

Tree Depth (Fixed): 3
Number of nodes in tree 15
Max depth of the tree 3
Feature importance of columns in tree :
{'variance': 0.6907456742365798, 'skewness': 0.20078679192070073, 'curtosis':
0.10846753384271939}
Train score of tree: 0.8651275820170109
Test score of tree: 0.8506375227686703

Tree Depth (Fixed): 4
Number of nodes in tree 29
Max depth of the tree 4
Feature importance of columns in tree :
{'variance': 0.6770750497956859, 'skewness': 0.2273373825380974, 'curtosis':
0.09558756766621658}
Train score of tree: 0.8784933171324423
Test score of tree: 0.8615664845173042

Tree Depth (Fixed): 5
Number of nodes in tree 43
Max depth of the tree 5
Feature importance of columns in tree :
{'variance': 0.6733490242445285, 'skewness': 0.2163992069035024, 'curtosis':
0.11025176885196918}
Train score of tree: 0.9125151883353585
Test score of tree: 0.8761384335154827

Tree Depth (Fixed): 6
Number of nodes in tree 59
Max depth of the tree 6
Feature importance of columns in tree :
{'variance': 0.6420710703463705, 'skewness': 0.1883056959212199, 'curtosis':
0.16962323373240967}
Train score of tree: 0.959902794653706
Test score of tree: 0.8979963570127505

Tree Depth (Fixed): 7
Number of nodes in tree 79
Max depth of the tree 7
Feature importance of columns in tree :
{'variance': 0.6380615853170323, 'skewness': 0.18846717828617351, 'curtosis':
0.17347123639679435}
Train score of tree: 0.9732685297691372
```

```
Test score of tree: 0.9016393442622951

Tree Depth (Fixed): 8
Number of nodes in tree 95
Max depth of the tree 8
Feature importance of columns in tree :
{'variance': 0.6232645504956679, 'skewness': 0.20889415074880974, 'curtosis':
0.16784129875552234}
Train score of tree: 0.9842041312272175
Test score of tree: 0.9089253187613844

Tree Depth (Fixed): 9
Number of nodes in tree 109
Max depth of the tree 9
Feature importance of columns in tree :
{'variance': 0.6113564960778676, 'skewness': 0.20407737572807844, 'curtosis':
0.184566128194054}
Train score of tree: 0.9927095990279465
Test score of tree: 0.9089253187613844

Tree Depth (Fixed): 10
Number of nodes in tree 117
Max depth of the tree 10
Feature importance of columns in tree :
{'variance': 0.619315954659742, 'skewness': 0.2016329335956422, 'curtosis': 0.
1790511117446159}
Train score of tree: 0.9975698663426489
Test score of tree: 0.912568306010929

Tree Depth (Fixed): 11
Number of nodes in tree 121
Max depth of the tree 11
Feature importance of columns in tree :
{'variance': 0.6134501333866513, 'skewness': 0.2003914035109112, 'curtosis':
0.18615846310243747}
Train score of tree: 1.0
Test score of tree: 0.9107468123861566
```

Show the visual output of the decision tree from Step-2.

In [11]:

```python
from sklearn import tree
import graphviz

dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=feature_names,
                                filled=True, rounded=True)

graph = graphviz.Source(dot_data)
graph.render(filename='decision_tree',format='png')
```

Out[11]:

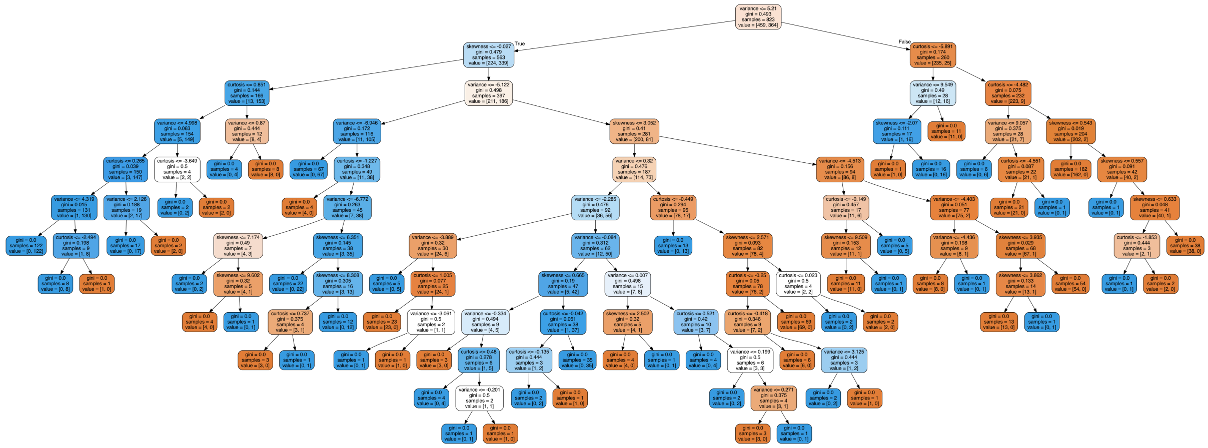
```
'decision_tree.png'
```

In [12]:

```python
from IPython.display import Image

Image(url="decision_tree.png")
```

Out[12]:



Show the visual output of the decision tree with highest test score from Step-5.

In [13]:

```
best_tree_index = np.argmax(test_scores)
best_tree = DecisionTreeClassifier(max_depth=best_tree_index+1,random_state=42
best_tree.fit(X_train,y_train)
# Export the decision tree to a dot file
dot_data_best = tree.export_graphviz(
    best_tree,
    out_file=None,
    feature_names=X.columns,
    class_names=["Authentic", "Counterfeit"],
    filled=True,
    rounded=True,
)

graph_best = graphviz.Source(dot_data_best)
graph_best.render("best_decision_tree_visualization", format="png")
```

Out[13]:

```
'best_decision_tree_visualization.png'
```

In [14]:

```
Image(url="best_decision_tree_visualization.png")
```

Out[14]: