Red Wine Quality Classification using Decision Tree

1. Data Pre-processing :

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```python
# Import Dataset

df = pd.read_csv('winequality-red.csv')
df
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [4]:
```
df.head()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

In [5]:
```
df.tail()
```

Out[5]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

In [6]:
```
# Divide the Dataset into Dependent and Independent Variables

X = df.drop('quality', axis=1)
Y = df['quality']
```

In [7]:

```
# Train-test-split

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.20, random_state=42)
```

2. Training the Model :

In [8]:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

In [9]:

```
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
model = classifier.fit(x_train, y_train)
```

3. Predict the Result :

In [10]:

```
y_pred = classifier.predict(x_test)
```

In [11]:

```
pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
```

Out[11]:

	Actual	Predicted
803	6	5
124	5	6
350	6	5
682	5	4
1326	6	6
...	...	...
1259	6	6
1295	5	5
1155	5	5
963	6	6
704	4	5

320 rows × 2 columns

4. Model Evaluation :

In [12]:

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

In [13]:

```
score = accuracy_score(y_test, y_pred)
print(score)
```

```
0.575
```

In [14]:

```
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.66      0.70      0.68       130
           6       0.59      0.54      0.56       132
           7       0.44      0.52      0.48        42
           8       0.00      0.00      0.00         5

    accuracy                           0.57       320
   macro avg       0.28      0.29      0.29       320
weighted avg       0.57      0.57      0.57       320
```

In [15]:

```
mat = confusion_matrix(y_test, y_pred)
print(mat)
```

```
[[ 0  0  1  0  0  0]
 [ 0  0  6  3  1  0]
 [ 1  5 91 28  5  0]
 [ 0  2 36 71 20  3]
 [ 0  1  3 15 22  1]
 [ 0  0  0  3  2  0]]
```

5. Plot the Decision Tree :

In [16]:

```
from sklearn import tree
```

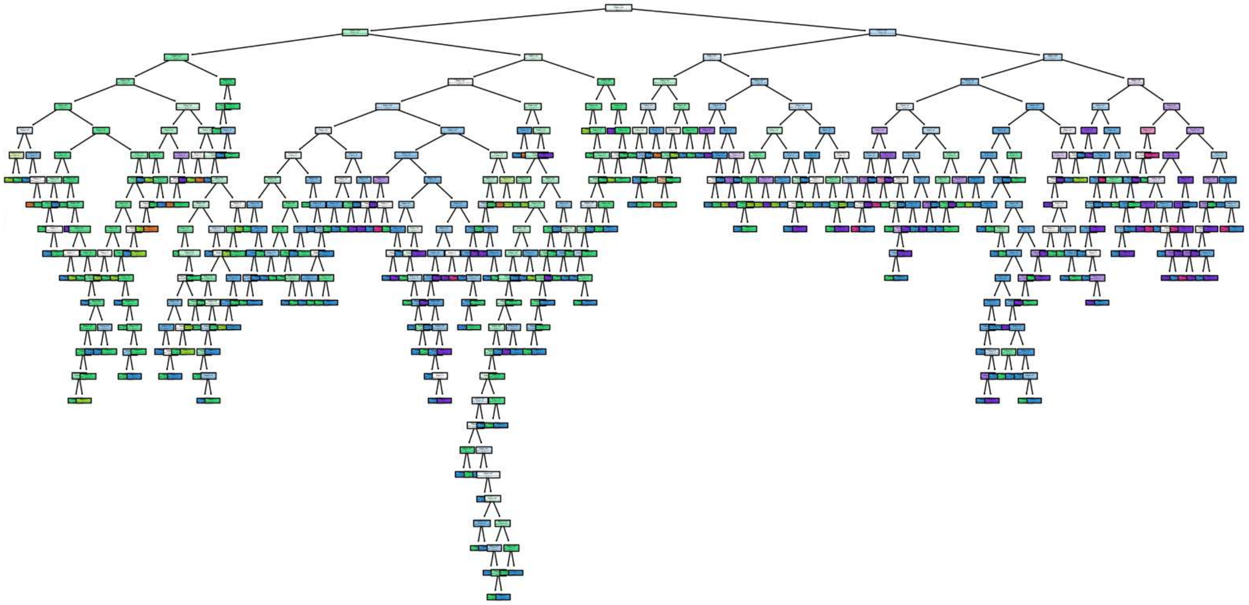
In [17]:

```
plt.figure(figsize=(15, 15))
tree.plot_tree(model, filled=True)
```

```
 Text(0.1489543320529236, 0.46, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0, 0,
0]'),
 Text(0.15578318395219803, 0.46, 'entropy = 0.0\nsamples = 4\nvalue = [0, 0, 4, 0, 0,
0]'),
 Text(0.1506615450277422, 0.54, 'entropy = 0.0\nsamples = 7\nvalue = [0, 0, 7, 0, 0,
0]'),
 Text(0.1694408877507469, 0.62, 'x[7] <= 0.995\nentropy = 1.146\nsamples = 34\nvalue
= [0, 1, 14, 19, 0, 0]'),
 Text(0.1660264618011097, 0.58, 'entropy = 0.0\nsamples = 6\nvalue = [0, 0, 0, 6, 0,
0]'),
 Text(0.17285531370038412, 0.58, 'x[6] <= 54.5\nentropy = 1.186\nsamples = 28\nvalue
= [0, 1, 14, 13, 0, 0]'),
 Text(0.16261203585147246, 0.54, 'x[9] <= 0.545\nentropy = 0.9\nsamples = 19\nvalue =
[0, 0, 13, 6, 0, 0]'),
 Text(0.15919760990183526, 0.5, 'entropy = 0.0\nsamples = 6\nvalue = [0, 0, 6, 0, 0,
0]'),
 Text(0.1660264618011097, 0.5, 'x[2] <= 0.34\nentropy = 0.996\nsamples = 13\nvalue =
[0, 0, 7, 6, 0, 0]'),
 Text(0.16261203585147246, 0.46, 'x[4] <= 0.082\nentropy = 0.971\nsamples = 10\nvalue
= [0, 0, 4, 6, 0, 0]'),
```

In [18]:

```
plt.figure(figsize=(20, 10))
plot_tree(classifier, filled=True, feature_names=X.columns, class_names=['3', '4', '5', '
plt.show()
```



Decision Tree for IRIS Dataset

1. Data Pre-processing :

In [19]:

```
df = pd.read_csv("IRIS.csv")
```

In [20]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [21]:

```
# Get unique values in Obejct data type column

df['species'].unique()
```

Out[21]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [22]:

```
# Encoding the String values from the 'Species' column

df['species'] = df['species'].map({'Iris-setosa':1, 'Iris-versicolor':2, 'Iris-virginica'
```

In [23]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

In [24]:

```
# Divide dataset in dependent and independent variables

X = df.drop('species', axis=1)
Y = df['species']
```

In [25]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state=42
```

2. Training the Model :

In [26]:

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

In [27]:

```
classifier = DecisionTreeClassifier()
model = classifier.fit(x_train, y_train)
```

3. Predict the Output :

In [28]:

```
y_pred = classifier.predict(x_test)
```

4. Model Evaluation Techniques :

In [29]:

```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

In [30]:

```
rp = classification_report(y_test, y_pred)
print(rp)
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00         9
           3       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [31]:

```
score = accuracy_score(y_test, y_pred)
print(score)
```

```
1.0
```

In [32]:

```
mat = confusion_matrix(y_test, y_pred)
print(mat)
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

5. Plot the Decision Tree :

In [33]:

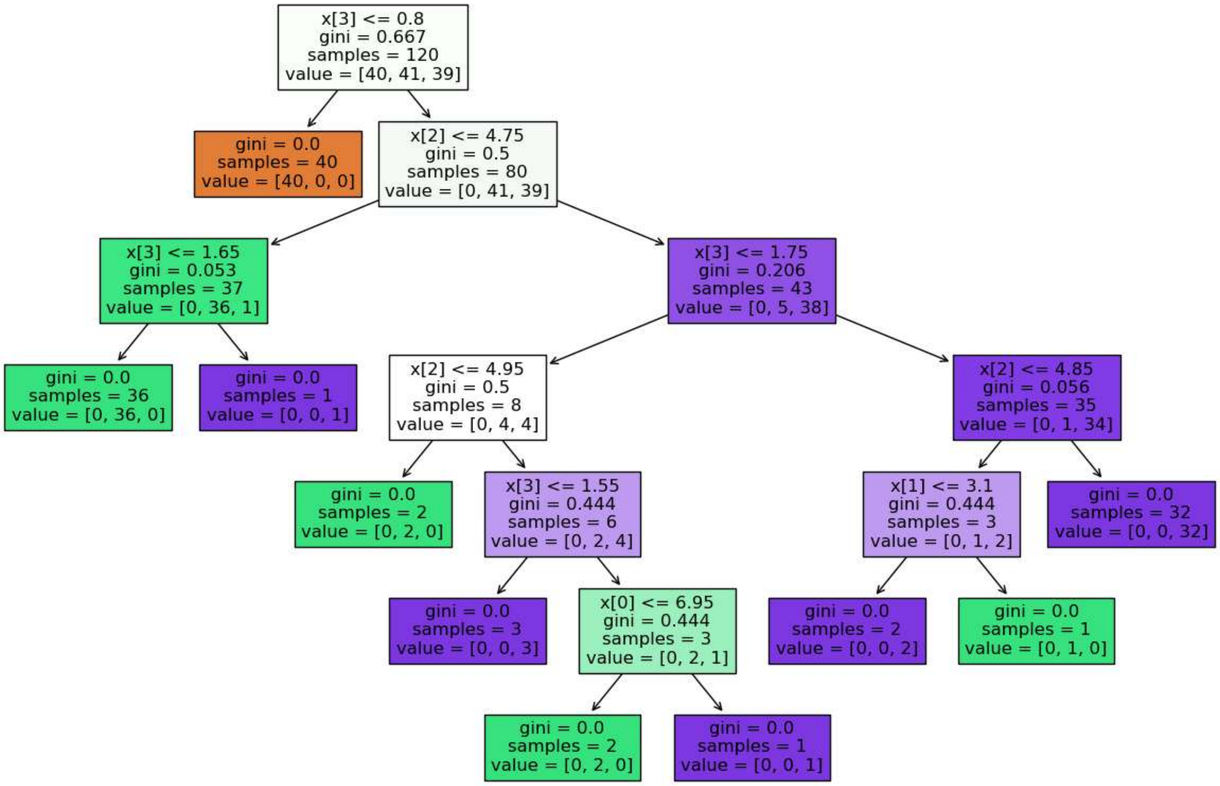
```
from sklearn import tree
```

In [34]:

```
plt.figure(figsize=(15, 10))
tree.plot_tree(model, filled=True)
```

Out[34]:

```
[Text(0.3076923076923077, 0.9285714285714286, 'x[3] <= 0.8\ngini = 0.667\nsamples = 120
\nvalue = [40, 41, 39]'),
 Text(0.23076923076923078, 0.7857142857142857, 'gini = 0.0\nsamples = 40\nvalue = [40,
0, 0]'),
 Text(0.38461538461538464, 0.7857142857142857, 'x[2] <= 4.75\ngini = 0.5\nsamples = 80\n
value = [0, 41, 39]'),
 Text(0.15384615384615385, 0.6428571428571429, 'x[3] <= 1.65\ngini = 0.053\nsamples = 37
\nvalue = [0, 36, 1]'),
 Text(0.07692307692307693, 0.5, 'gini = 0.0\nsamples = 36\nvalue = [0, 36, 0]'),
 Text(0.23076923076923078, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(0.6153846153846154, 0.6428571428571429, 'x[3] <= 1.75\ngini = 0.206\nsamples = 43
\nvalue = [0, 5, 38]'),
 Text(0.38461538461538464, 0.5, 'x[2] <= 4.95\ngini = 0.5\nsamples = 8\nvalue = [0, 4,
4]'),
 Text(0.3076923076923077, 0.35714285714285715, 'gini = 0.0\nsamples = 2\nvalue = [0, 2,
0]'),
 Text(0.46153846153846156, 0.35714285714285715, 'x[3] <= 1.55\ngini = 0.444\nsamples = 6
\nvalue = [0, 2, 4]'),
 Text(0.38461538461538464, 0.21428571428571427, 'gini = 0.0\nsamples = 3\nvalue = [0, 0,
3]'),
 Text(0.5384615384615384, 0.21428571428571427, 'x[0] <= 6.95\ngini = 0.444\nsamples = 3
\nvalue = [0, 2, 1]'),
 Text(0.46153846153846156, 0.07142857142857142, 'gini = 0.0\nsamples = 2\nvalue = [0, 2,
0]'),
 Text(0.6153846153846154, 0.07142857142857142, 'gini = 0.0\nsamples = 1\nvalue = [0, 0,
1]'),
 Text(0.8461538461538461, 0.5, 'x[2] <= 4.85\ngini = 0.056\nsamples = 35\nvalue = [0, 1,
34]'),
 Text(0.7692307692307693, 0.35714285714285715, 'x[1] <= 3.1\ngini = 0.444\nsamples = 3\n
value = [0, 1, 2]'),
 Text(0.6923076923076923, 0.21428571428571427, 'gini = 0.0\nsamples = 2\nvalue = [0, 0,
2]'),
 Text(0.8461538461538461, 0.21428571428571427, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(0.9230769230769231, 0.35714285714285715, 'gini = 0.0\nsamples = 32\nvalue = [0, 0,
32]')]
```


x[3] <= 0.8
gini = 0.667
samples = 120
value = [40, 41, 39]
gini = 0.0
samples = 40
value = [40, 0, 0]
x[2] <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
x[3] <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
x[3] <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
gini = 0.0
samples = 36
value = [0, 36, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]
x[2] <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
x[2] <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
gini = 0.0
samples = 2
value = [0, 2, 0]
x[3] <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
x[1] <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
gini = 0.0
samples = 32
value = [0, 0, 32]
gini = 0.0
samples = 3
value = [0, 0, 3]
x[0] <= 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 2
value = [0, 2, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]

In [ ]: