Machine Learning

Decision Tree Classifier

In [2]:

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

In [3]:

```
df=pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Wine-Quality-Dataset/m
```

In [70]:

```
df.head()
```

Out[70]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9

In [5]:

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

Out[5]:

```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [6]:

```
df['quality'].nunique()
```

Out[6]:

```
6
```

In [7]:

```
df['quality'].value_counts()
```

Out[7]:

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

In [8]:

```
df.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
240
```

In [10]:

```
df = df.drop_duplicates()
```

In [11]:

```
df.duplicated().sum()
```

Out[11]:

```
0
```

Example

In [12]:

```
demo = pd.DataFrame([1,3,4,6,5,4,2,3,4,6,7,4,3])
```

In [13]:

```
demo.duplicated().sum()
```

Out[13]:

```
6
```

Independent and Dependent Seperation

In [14]:

```
X =df.drop("quality",axis=1)
```

In [15]:

```
y=df['quality']
```

train test split

In [16]:

```
from sklearn.model_selection import train_test_split,GridSearchCV
```

In [17]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=10)
```

In [19]:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

In [20]:

```
model.fit(X_train,y_train)
```

Out[20]:

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [21]:

```
model.score(X_train,y_train)
```

Out[21]:

```
1.0
```

In [22]:

```
y_predict = model.predict(X_test)
y_predict
```

Out[22]:

```
array([6, 6, 6, 5, 6, 7, 6, 7, 5, 7, 5, 5, 6, 5, 5, 6, 5, 5, 7, 5, 6, 6,
       6, 5, 5, 5, 6, 6, 5, 5, 6, 6, 6, 5, 6, 5, 5, 4, 7, 4, 5, 6, 6, 6,
       5, 5, 5, 5, 5, 6, 6, 5, 5, 6, 5, 7, 6, 7, 5, 5, 6, 6, 7, 5, 6, 6,
       5, 6, 6, 5, 5, 5, 5, 6, 4, 5, 6, 5, 5, 6, 6, 5, 5, 5, 5, 5, 6, 7,
       5, 5, 5, 5, 5, 6, 6, 5, 5, 6, 5, 6, 6, 5, 6, 5, 5, 5, 6, 7, 5, 5,
       5, 5, 7, 5, 5, 6, 7, 6, 5, 6, 4, 6, 5, 6, 7, 6, 7, 5, 5, 5, 6, 5,
       5, 5, 4, 5, 5, 6, 4, 7, 5, 5, 7, 6, 5, 5, 5, 6, 5, 7, 6, 6, 6, 5,
       5, 6, 6, 5, 6, 6, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 7, 6, 6, 5, 5, 7,
       6, 6, 6, 5, 6, 6, 6, 5, 7, 5, 5, 6, 5, 5, 7, 6, 6, 5, 5, 5, 6, 5,
       7, 7, 6, 5, 5, 6, 6, 5, 6, 4, 6, 6, 6, 5, 7, 7, 6, 5, 5, 5, 7, 8,
       4, 6, 5, 5, 6, 6, 6, 6, 5, 6, 5, 5, 5, 7, 6, 5, 5, 5, 5, 4, 5, 5,
       4, 6, 8, 8, 5, 6, 6, 5, 6, 4, 8, 6, 6, 6, 7, 5, 6, 8, 6, 7, 5, 6,
       6, 6, 5, 6, 5, 6, 6, 6, 7, 5, 6, 6, 7, 7, 7, 6, 7, 5, 5, 6, 5, 6,
       4, 7, 5, 6, 5, 6, 5, 7, 6, 6, 5, 6, 7, 5, 6, 6, 5, 6, 6, 7, 5, 6,
       5, 5, 5, 7, 6, 5, 5, 6, 5, 5, 6, 6, 6, 7, 6, 6, 6, 5, 5, 5, 6, 5,
       6, 5, 7, 6, 6, 6, 5, 5, 5, 5, 7, 5, 6, 5, 5, 5, 6, 7, 7, 6, 6, 7,
       6, 6, 6, 5, 5, 5, 6, 6, 5, 6, 7, 6, 6, 6, 6, 7, 6, 5, 6, 8, 6, 5,
       6, 7, 5, 6, 5, 5, 6, 6, 5, 5, 7, 6, 5, 6, 5, 5, 7, 6, 5, 5, 6, 5,
       6, 5, 6, 6, 5, 5, 5, 4, 7, 5, 5, 6, 6, 6, 5, 6, 5, 5, 5, 6, 6, 5,
       5, 6, 5, 5, 6, 5, 5, 7, 6, 5, 6, 6, 6, 5, 6, 7, 5, 6, 6, 6, 6, 6,
       5, 5, 6, 6, 6, 6, 5, 4, 5], dtype=int64)
```

In [23]:

```
from sklearn.metrics import accuracy_score
```

In [24]:

```
accuracy_score(y_test,y_predict)
```

Out[24]:

```
0.5144766146993318
```

GridSearchCV

In [26]:

```
grid_param = {
    'criterion':['gini','entropy'],
    'max_depth' : range(2,32,1),
    'min_samples_leaf' : range(1,10,1),
    'min_samples_split': range(2,10,1),
    'splitter' : ['best','random']
}
```

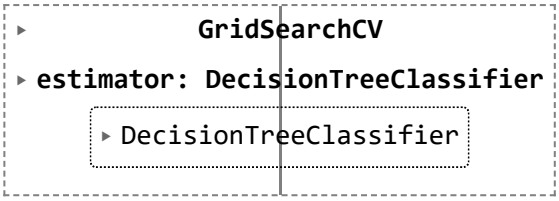
In [28]:

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=model,param_grid=grid_param,cv=5)
```

In [29]:

```
grid_search.fit(X_train,y_train)
```

Out[29]:



In [30]:

```
grid_search.best_params_
```

Out[30]:

```
{'criterion': 'gini',
 'max_depth': 13,
 'min_samples_leaf': 9,
 'min_samples_split': 5,
 'splitter': 'random'}
```

In [ ]:

```

```

In [44]:

```
model_with_best_params = DecisionTreeClassifier(criterion= 'gini',
 max_depth= 13,
 min_samples_leaf= 9,
 min_samples_split= 5,
 splitter= 'random')
```

In [45]:

```
model_with_best_params.fit(X_train,y_train)
```

Out[45]:

```
DecisionTreeClassifier(max_depth=13, min_samples_leaf=9, min_samples_split=
5,
                       splitter='random')
```

In [46]:

```
y_predict2 = model_with_best_params.predict(X_test)
```

In [47]:

```
accuracy_score(y_test,y_predict2)
```

Out[47]:

```
0.5657015590200446
```

In [ ]:

In [ ]: