# Finding the quality of alcohol using Decision-Tree-Classifier model

In [3]:
```
import pandas as pd
import numpy as np
```

## Preprocessing

In [2]:
```
df=pd.read_csv("https://raw.githubusercontent.com/shrikant-temburwar/Wine-Qu
df
```

Out[2]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcoh |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9 |
| **1** | 7.8 | 0.880 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.99680 | 3.20 | 0.68 | 9 |
| **2** | 7.8 | 0.760 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.99700 | 3.26 | 0.65 | 9 |
| **3** | 11.2 | 0.280 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.99800 | 3.16 | 0.58 | 9 |
| **4** | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 | 0.56 | 9 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1594** | 6.2 | 0.600 | 0.08 | 2.0 | 0.090 | 32.0 | 44.0 | 0.99490 | 3.45 | 0.58 | 10 |
| **1595** | 5.9 | 0.550 | 0.10 | 2.2 | 0.062 | 39.0 | 51.0 | 0.99512 | 3.52 | 0.76 | 11 |
| **1596** | 6.3 | 0.510 | 0.13 | 2.3 | 0.076 | 29.0 | 40.0 | 0.99574 | 3.42 | 0.75 | 11 |
| **1597** | 5.9 | 0.645 | 0.12 | 2.0 | 0.075 | 32.0 | 44.0 | 0.99547 | 3.57 | 0.71 | 10 |
| **1598** | 6.0 | 0.310 | 0.47 | 3.6 | 0.067 | 18.0 | 42.0 | 0.99549 | 3.39 | 0.66 | 11 |

1599 rows × 12 columns

In [4]:
```
df['quality'].unique()
```

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

In [6]:
```
len(df['quality'].unique())
```

Out[6]:
```
6
```

In [5]:
```
1 df['quality'].value_counts()
```

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

In [8]:
```
1 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]:
```
1 df.duplicated().sum()
```

Out[9]:
```
240
```

In [11]:
```
1 df=df.drop_duplicates()
```

In [12]:
```
1 df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
1 X=df.drop("quality",axis=1)
2 y=df['quality']
```

In [14]:
```
1 from sklearn.model_selection import train_test_split,GridSearchCV
2
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, ra
```

## Training

In [16]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [17]:
```
model=DecisionTreeClassifier()
```

In [18]:
```
model.fit(X_train,y_train)
```

Out[18]:
```
DecisionTreeClassifier()
```

In [19]:
```
model.score(X_train,y_train)
```

Out[19]:
```
1.0
```

In [21]:
```
y_predict=model.predict(X_test)
```

In [22]:
```
from sklearn.metrics import accuracy_score

```

## Testing

In [23]:
```
accuracy_score(y_test,y_predict)
```

Out[23]:
```
0.47438752783964366
```

## Tuning and finding best grid_parameters

In [24]:
```
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 [25]:
```
grid_search=GridSearchCV(estimator=model,param_grid=grid_param,cv=5)
```

In [26]:
```
1 grid_search.fit(X_train,y_train)
```

Out[26]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']})
```

In [27]:
```
1 grid_search.best_params_
```

Out[27]:
```
{'criterion': 'gini',
 'max_depth': 4,
 'min_samples_leaf': 2,
 'min_samples_split': 4,
 'splitter': 'random'}
```

In [34]:
```
1 model_with_best_params=DecisionTreeClassifier(criterion= 'gini',max_depth= 4
```

## Training and Testing after Tuning

In [35]:
```
1 model_with_best_params.fit(X_train,y_train)
```

Out[35]:
```
DecisionTreeClassifier(max_depth=4, min_samples_leaf=2, min_samples_split=4,
                       splitter='random')
```

In [36]:
```
1 y_prediction2=model_with_best_params.predict(X_test)
```

In [38]:
```
1 accuracy_score(y_test,y_prediction2)
```

Out[38]:
```
0.5634743875278396
```

In [ ]:
```
1
```

In [ ]:
```
1
```