In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

In [2]:
```
df=pd.read_csv('C:\\Users\\HP\\Documents\\Data Science\\WineQT.csv')
```

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

Out[3]:

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

In [4]:
```
df.shape
```

Out[4]:
```
(1143, 13)
```

In [5]:
```
df.drop(['Id'],axis=1,inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	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 [7]:
```
df.columns
```

Out[7]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [9]:
```
df['quality'].unique()
```

Out[9]:
```
array([5, 6, 7, 4, 8, 3], dtype=int64)
```

In [10]:
```
df.mean()
```

Out[10]:
```
fixed acidity            8.311111
volatile acidity         0.531339
citric acid              0.268364
residual sugar           2.532152
chlorides                0.086933
free sulfur dioxide     15.615486
total sulfur dioxide    45.914698
density                  0.996730
pH                       3.311015
sulphates                0.657708
alcohol                 10.442111
quality                  5.657043
dtype: float64
```

In [11]:
```
#bins should always have a minimum value, average and maximum number
bins=(1, 5.5, 8)
groupname=('good','bad')
```

In [12]:
```
df['quality']=pd.cut(df['quality'],bins=bins, labels=groupname)
```

In [13]:
```
df['quality'].unique()
```

Out[13]:
```
['good', 'bad']
Categories (2, object): ['good' < 'bad']
```

In [14]:
```
df.head()
```

Out[14]:

	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	good
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	good
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	good
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	bad
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	good

In [15]:
```
label=LabelEncoder()
```

In [16]:
```
df['quality']=label.fit_transform(df['quality'])
```

In [17]:
```
df.head()
```

Out[17]:

	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	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	1

In [18]:
```
x=df.drop(['quality'],axis=1)
y=df['quality']
```

In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [20]:
```
x_train.shape
```

Out[20]:
```
(914, 11)
```

In [21]:
```
x_test.shape
```

Out[21]:
```
(229, 11)
```

In [22]:
```
nb=GaussianNB()
```

In [23]:
```
nb.fit(x_train,y_train)
```

Out[23]:
```
▾ GaussianNB
GaussianNB()
```

In [24]:
```
nbpred=nb.predict(x_test)
```

In [25]:
```
nbpred[:10]
```

Out[25]:
```
array([1, 1, 1, 1, 0, 0, 1, 1, 0, 0])
```

In [26]:
```
svm=SVC()
```

In [27]:
```
svm.fit(x_train,y_train)
```

Out[27]:
```
SVC
SVC()
```

In [28]:
```
svmpred=svm.predict(x_test)
```

In [29]:
```
svmpred[:10]
```

Out[29]:
```
array([0, 1, 1, 0, 0, 0, 0, 1, 0, 1])
```

In [30]:
```
accuracy=accuracy_score(y_test,svmpred)*100
```

In [31]:
```
accuracy
```

Out[31]:
```
68.12227074235808
```

In [32]:
```
knn=KNeighborsClassifier(n_neighbors=5)
```

In [33]:
```
knn.fit(x_train,y_train)
```

Out[33]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [34]:
```
knnpred=knn.predict(x_test)
```

In [35]:
```
knnpred[:10]
```

Out[35]:
```
array([1, 1, 1, 0, 0, 0, 1, 1, 0, 1])
```

In [36]:
```
accuracy=accuracy_score(y_test,knnpred)*100
```

In [37]:
```
accuracy
```

Out[37]:
```
67.24890829694323
```

In [ ]:
```

```