In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [ ]:

In [ ]:
```
df = pd.read_csv("D:\MIT ADT\Third Year - Sem 2\ML LAB\Assign 7 - KNN\winequalityN.csv")
```

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

Out[ ]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
type                    0
fixed acidity          10
volatile acidity        8
citric acid             3
residual sugar          2
chlorides               2
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      9
sulphates               4
alcohol                 0
quality                 0
dtype: int64
```

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

Out[ ]:
```
1168
```

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

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

Out[ ]:
```
0
```

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

Out[ ]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
6	white	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6

In [ ]:
```
df.columns
```

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

In [ ]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)

col_impute = ["fixed acidity", "volatile acidity", "citric acid", 'residual sugar', 'chlorides', 'pH', 'sulphates']

for i in col_impute:
    df[i] = imputer.fit_transform(df[[i]]).ravel()
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
lbl_enc = LabelEncoder()
df['type'] = lbl_enc.fit_transform(df['type'])
```

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

Out[ ]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
6	1	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6

In [ ]:
```
df.corr()
```

Out[ ]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
type	1.000000	-0.486310	-0.644478	0.183703	0.328597	-0.499735	0.465295	0.694181	-0.428757	-0.310342	-0.489255	0.057334	0.114889
fixed acidity	-0.486310	1.000000	0.215063	0.329085	-0.104652	0.288898	-0.281588	-0.327327	0.477807	-0.270147	0.305710	-0.102778	-0.080482
volatile acidity	-0.644478	0.215063	1.000000	-0.383113	-0.163926	0.367330	-0.349337	-0.401499	0.307121	0.245854	0.225871	-0.064840	-0.264307
citric acid	0.183703	0.329085	-0.383113	1.000000	0.146526	0.054922	0.132147	0.195084	0.094852	-0.343196	0.060445	-0.005557	0.098774
residual sugar	0.328597	-0.104652	-0.163926	0.146526	1.000000	-0.123314	0.398811	0.487338	0.521661	-0.233905	-0.174800	-0.306092	-0.057503
chlorides	-0.499735	0.288898	0.367330	0.054922	-0.123314	1.000000	-0.186824	-0.270034	0.371442	0.026535	0.404614	-0.269132	-0.202115
free sulfur dioxide	0.465295	-0.281588	-0.349337	0.132147	0.398811	-0.186824	1.000000	0.720666	0.006687	-0.141315	-0.198378	-0.170396	0.054456
total sulfur dioxide	0.694181	-0.327327	-0.401499	0.195084	0.487338	-0.270034	0.720666	1.000000	0.007359	-0.222407	-0.274619	-0.249597	-0.050387
density	-0.428757	0.477807	0.307121	0.094852	0.521661	0.371442	0.006687	0.007359	1.000000	0.034152	0.282221	-0.668216	-0.326978
pH	-0.310342	-0.270147	0.245854	-0.343196	-0.233905	0.026535	-0.141315	-0.222407	0.034152	1.000000	0.166237	0.097453	0.039876
sulphates	-0.489255	0.305710	0.225871	0.060445	-0.174800	0.404614	-0.198378	-0.274619	0.282221	0.166237	1.000000	-0.017893	0.041492
alcohol	0.057334	-0.102778	-0.064840	-0.005557	-0.306092	-0.269132	-0.170396	-0.249597	-0.668216	0.097453	-0.017893	1.000000	0.469555
quality	0.114889	-0.080482	-0.264307	0.098774	-0.057503	-0.202115	0.054456	-0.050387	-0.326978	0.039876	0.041492	0.469555	1.000000

In [ ]:

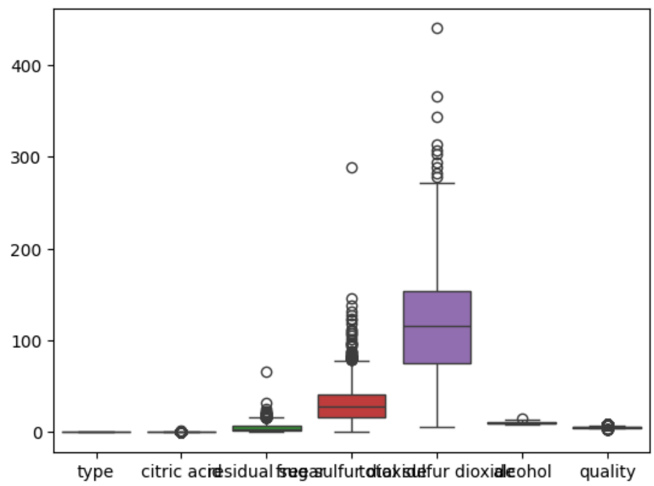
```
df = df.drop(["fixed acidity", "volatile acidity", "chlorides", "density", "pH", "sulphates"], axis=1)
```

In [ ]:

```
sns.boxplot(df)
```

Out[ ]:

```
<Axes: >
```



In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['type', 'citric acid', 'residual sugar', 'free sulfur dioxide',
       'total sulfur dioxide', 'alcohol', 'quality'],
      dtype='object')
```

In [ ]:

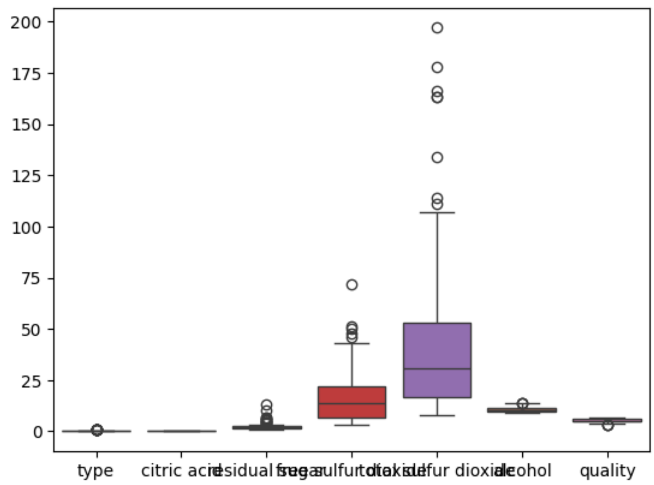
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)

IQR = Q3 - Q1

outliers = ((df<(Q1-1.5*IQR))| df>(Q3+1.5*IQR)).any(axis=1)

df_no_outliers = df[~outliers]
```

In [ ]:

```
sns.boxplot(df_no_outliers)
```

Out[ ]:

```
<Axes: >
```



In [ ]:
```
X = df.drop("type", axis=1)
y = df["type"]
```

In [ ]:
```
from sklearn.model_selection import train_test_split

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y , test_size=0.2,random_state=0)

print(Xtrain.shape, Xtest.shape)
print(ytrain.shape, ytest.shape)
```

```
(4263, 6) (1066, 6)
(4263,) (1066,)
```

In [ ]:
```
#Scale the Data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

Xtrain_std = scaler.fit_transform(Xtrain)
Xtest_std = scaler.transform(Xtest)
```

class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

In [ ]:
```
X.isna().sum()
```

Out[ ]:
```
citric acid             0
residual sugar          0
free sulfur dioxide     0
total sulfur dioxide    0
alcohol                 0
quality                 0
dtype: int64
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
#neigh = KNeighborsClassifier(n_neighbors=3)

for i in range(1,16):
    print("Neighbours: ", i)
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(Xtrain_std, ytrain)
    print("Training accuracy: ", neigh.score(Xtrain_std, ytrain))
    print("Testing Accuracy: ", neigh.score(Xtest_std, ytest))
```

```
Neighbours:  1
Training accuracy:  0.9995308468214872
Testing Accuracy:  0.9390243902439024
Neighbours:  2
Training accuracy:  0.9673938540933614
Testing Accuracy:  0.924015009380863
Neighbours:  3
Training accuracy:  0.9643443584330283
Testing Accuracy:  0.948405253283302
Neighbours:  4
Training accuracy:  0.956368754398311
Testing Accuracy:  0.9409005628517824
Neighbours:  5
Training accuracy:  0.9573070607553367
Testing Accuracy:  0.948405253283302
Neighbours:  6
Training accuracy:  0.9537884119164908
Testing Accuracy:  0.9446529080675422
Neighbours:  7
Training accuracy:  0.9556650246305419
Testing Accuracy:  0.9455909943714822
Neighbours:  8
Training accuracy:  0.952146375791696
Testing Accuracy:  0.9418386491557224
Neighbours:  9
Training accuracy:  0.9528501055594651
Testing Accuracy:  0.948405253283302
Neighbours:  10
Training accuracy:  0.9516772226131832
Testing Accuracy:  0.9409005628517824
Neighbours:  11
Training accuracy:  0.9505043396669013
Testing Accuracy:  0.9455909943714822
Neighbours:  12
Training accuracy:  0.9474548440065681
Testing Accuracy:  0.9437148217636022
Neighbours:  13
Training accuracy:  0.9498006098991321
Testing Accuracy:  0.9446529080675422
Neighbours:  14
Training accuracy:  0.9486277269528501
Testing Accuracy:  0.9409005628517824
Neighbours:  15
Training accuracy:  0.9476894205958245
Testing Accuracy:  0.9399624765478424
```

In [ ]: