In [237]:

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [283]:

```
df = pd.read_csv('winequalityN.csv')
#df.drop(columns = 'quality')
df = df.dropna()
df.columns
```

Out[283]:

```
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 [239]:

```
df_wine = pd.DataFrame(df,
                columns=['type','fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol'])
df_wine = pd.DataFrame(df)
```

In [245]:

```
df_wine.isnull().sum()
```

Out[245]:

```
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 [248]:

```
df_wine = df_wine.dropna()
df_wine.isnull().sum()
```

Out[248]:

```
type                    0
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 [240]:

```
def myfunction(t):
    if t == "red":
        return 0
    elif t == "white":
        return 1
    else:
        return 2

df_wine["type"] = df_wine["type"].apply(myfunction)
df_wine.head(10)
```

Out[240]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40
4	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40
5	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44
6	1	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47
7	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45
8	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49
9	1	8.1	0.22	0.43	1.5	0.044	28.0	129.0	0.9938	3.22	0.45

In [ ]:

```

```

In [225]:

```
df_wine.head()
```

Out[225]:

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

In [226]:

```
df_wine.tail()
```

Out[226]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphat
6492	0	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.
6493	0	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	N
6494	0	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.
6495	0	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.
6496	0	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.

In [227]:

```
df_wine.type.value_counts()
```

Out[227]:

```
1    4898
0    1599
Name: type, dtype: int64
```

In [249]:

```
X = df_wine.iloc[:,1:12]

y = df_wine['type']
X.head()
```

Out[249]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcoh
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9

In [288]:

```
df["type"] = df["type"].apply(myfunction)
df.head(10)
```

Out[288]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45
1	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49
2	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44
3	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40
4	1	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40
5	1	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44
6	1	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47
7	1	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45
8	1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49
9	1	8.1	0.22	0.43	1.5	0.044	28.0	129.0	0.9938	3.22	0.45

In [289]:

```
X2 = df.iloc[:, 0:12]
Y2 = df['quality']
Y2
```

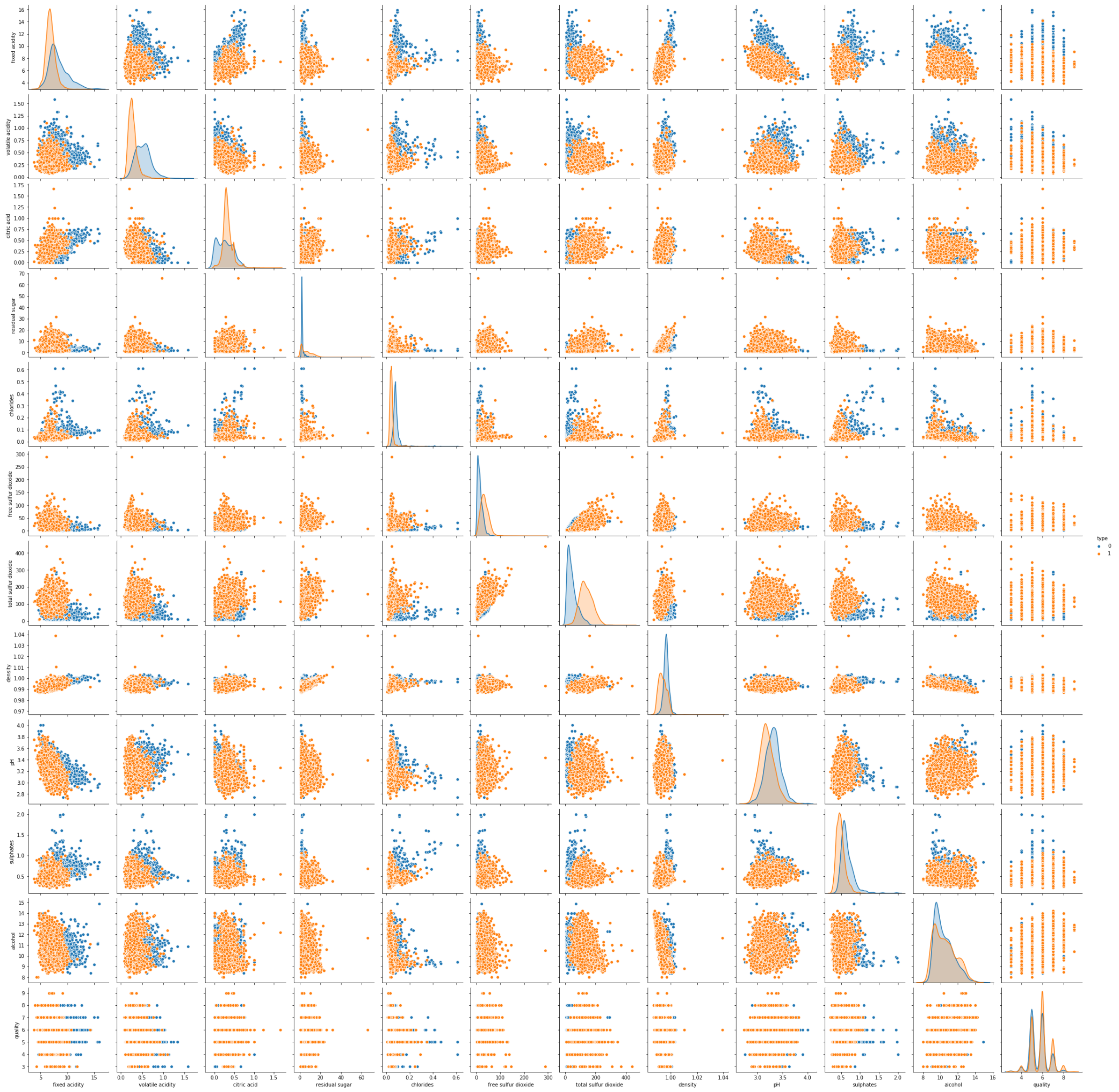
Out[289]:

```
0       6
1       6
2       6
3       6
4       6
       ..
6491    6
6492    5
6494    6
6495    5
6496    6
Name: quality, Length: 6463, dtype: int64
```

Visualization

In [229]:

```
sb.pairplot (df_wine, hue='type')
plt.show()
```



Model Training

In [250]:

```
#this is for predicting type

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 100)
```

In [251]:

```
X_train.head()
```

Out[251]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	al
2677	6.2	0.250	0.31	3.2	0.030	32.0	150.0	0.99014	3.18	0.31	
5368	13.0	0.320	0.65	2.6	0.093	15.0	47.0	0.99960	3.05	0.61	
2577	6.7	0.250	0.36	8.6	0.037	63.0	206.0	0.99553	3.18	0.50	
4997	8.1	0.545	0.18	1.9	0.080	13.0	35.0	0.99720	3.30	0.59	
1770	7.8	0.390	0.26	9.9	0.059	33.0	181.0	0.99550	3.04	0.42	

In [274]:

```
knn_model = KNeighborsClassifier(n_neighbors = 50)

knn_model.fit(X_train, y_train)
```

Out[274]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=50, p=2,
                     weights='uniform')
```

Prediction

In [275]:

```
y_pred = knn_model.predict(X_test)
```

In [276]:

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

```
0.9344059405940595
```

In [ ]:

```

```

In [ ]:

```
print(y_pred)
```

In [290]:

```
#this is for predicting quality

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y2, random_state = 100)
```

In [291]:

```
knn_model = KNeighborsClassifier(n_neighbors = 50)

knn_model.fit(X2_train, Y2_train)
```

Out[291]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=50, p=2,
                     weights='uniform')
```

In [292]:

```
Y2_pred = knn_model.predict(X2_test)
```

In [293]:

```
print(accuracy_score(Y2_test, Y2_pred))
```

```
0.4542079207920792
```

In [ ]:

```

```