In [1]:

```python
# Libraries
import pandas as pd
import os
```

In [2]:

```python
# importing dataset

os.chdir('/Users/tomisin/Dropbox/My Mac (Tomisins-MacBook-Pro.local)/Documents/I
```

In [4]:

```python
# importing dataset
df = pd.read_csv('winequality-red.csv')
```

In [5]:

```python
df
```

Out[5]:

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

1599 rows × 12 columns

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [8]:

```
df.isna().sum() #shows columns with their number of missing values
```

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.describe()
```

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total su diox
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.0000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000

In [10]:

```
df.columns
```

Out[10]:

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual s
ugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'de
nsity',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

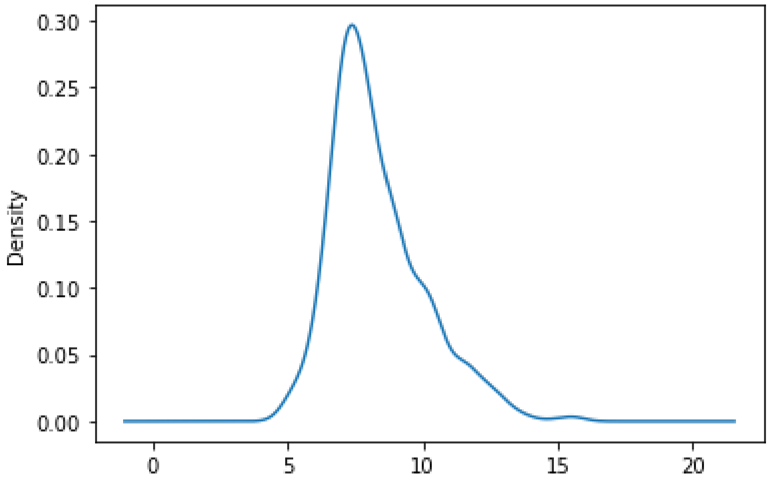
Making plots

In [11]:

```
df['fixed acidity'].plot(kind='density')
```

Out[11]:

```
<AxesSubplot:ylabel='Density'>
```



OUTLIERS

In [12]:

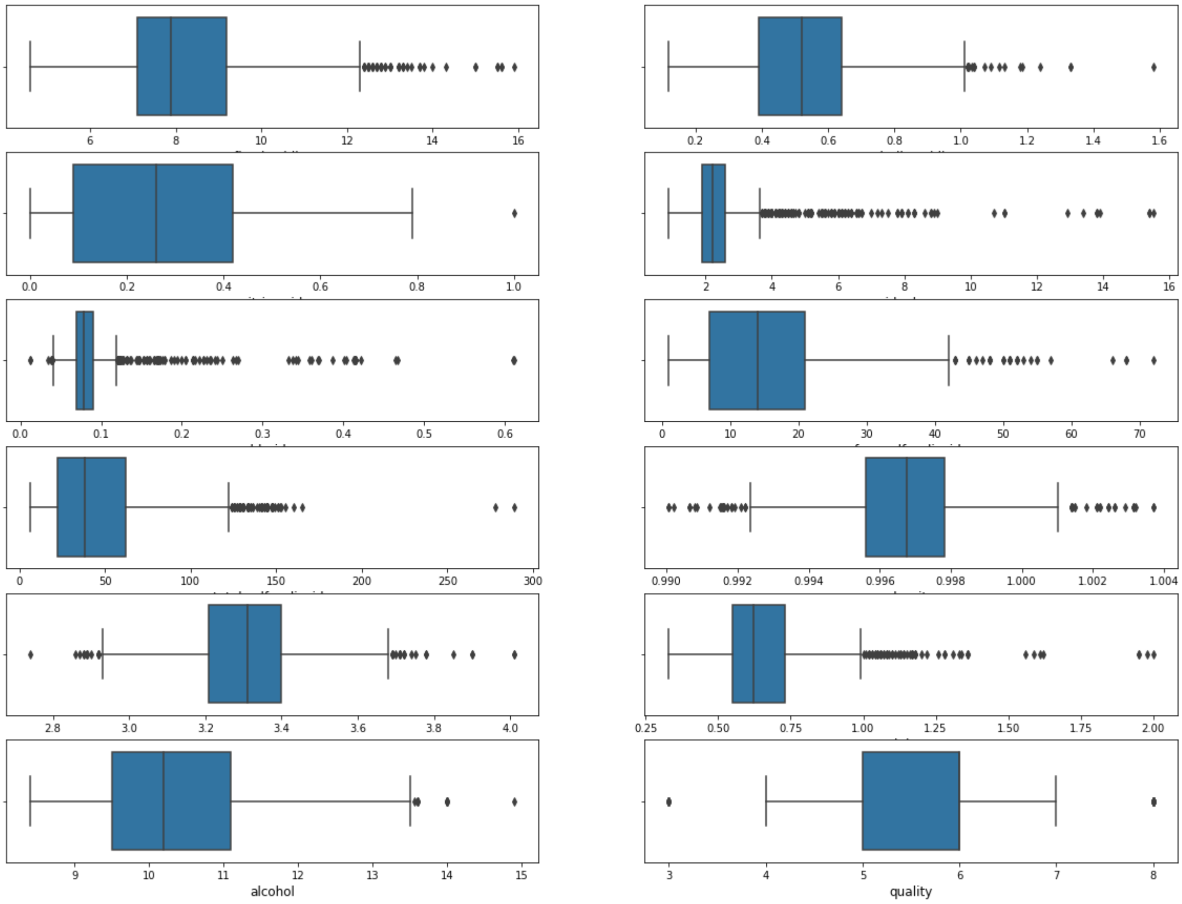
```
import matplotlib.pyplot as plt
import seaborn as sns

```

In [13]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Checking for outliers
plt.figure(figsize = (20, 15))
for i in range (len(df.columns)):
    plt.subplot(6, 2, i+1)
    sns.boxplot(x = df.iloc[:, i])
    plt.xlabel(df.columns[i], size = 12)
```

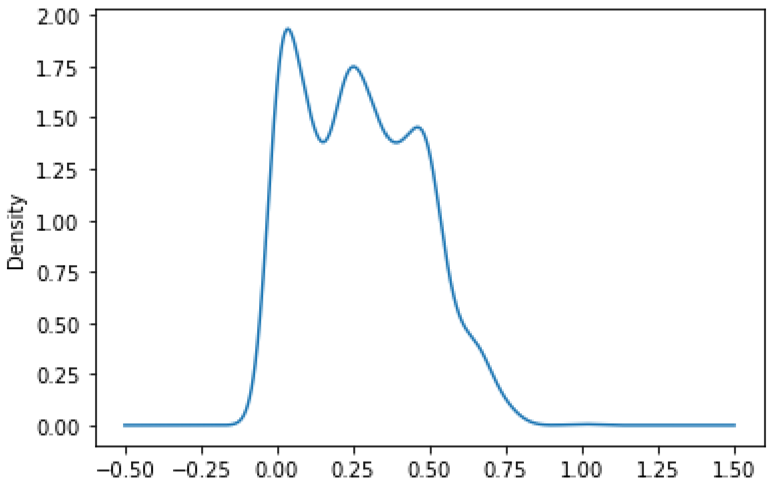


In [14]:

```
df['citric acid'].plot(kind='density')
```

Out[14]:

```
<AxesSubplot:ylabel='Density'>
```

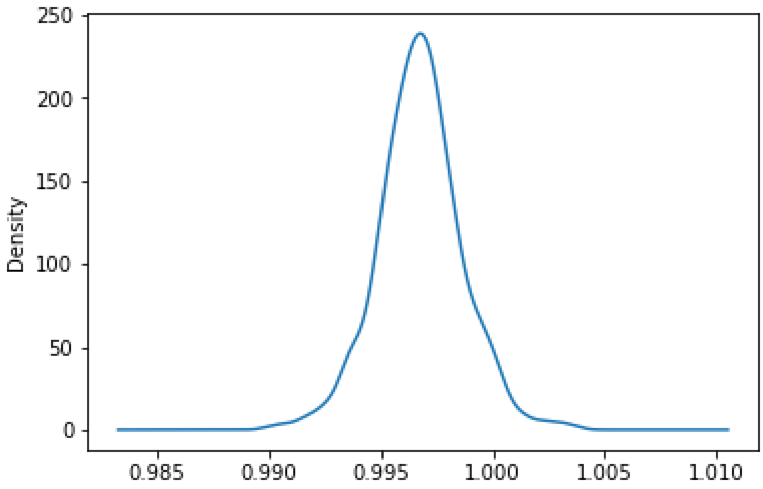


In [15]:

```
df['density'].plot(kind='density')
```

Out[15]:

```
<AxesSubplot:ylabel='Density'>
```



In [16]:

```
df.columns[0]
```

Out[16]:

```
'fixed acidity'
```

Removing outliers

In [17]:

```

def Outliers(data, feature):
    IQ1 = data[feature].quantile(0.25)
    IQ3 = data[feature].quantile(0.75)
    IQR = IQ3 - IQ1

    lower_bound = IQ1 - 1.5 * IQR
    upper_bound = IQ3 + 1.5 * IQR

    index = data.index[ (data[feature] < lower_bound) | (data[feature] > upper_b
    return index
```

Getting index of all the outliers

In [18]:

```

index = []
for i in df.columns:
    index.extend(Outliers(df, i))
index = set(index)
print("Total number of outliers are {}".format(len(index)))

# Dropping all the outliers
df.drop(index, inplace = True, axis = 0)
df.shape
```

```
Total number of outliers are 420
```

Out[18]:

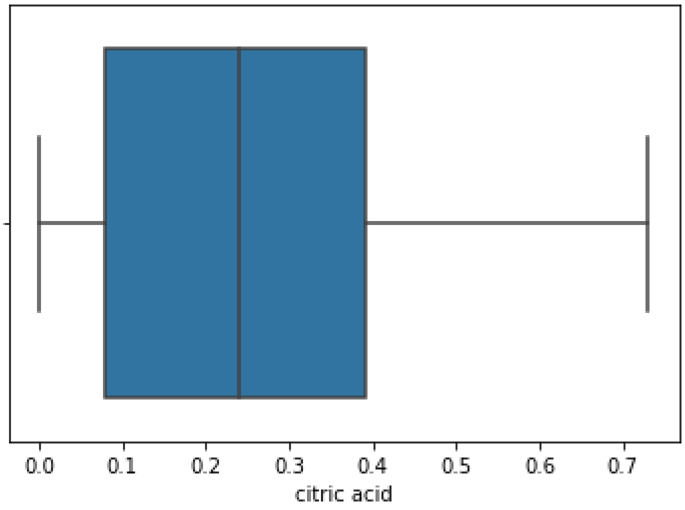
```
(1179, 12)
```

In [19]:

```
sns.boxplot(x = df.iloc[:, 2])
```

Out[19]:

```
<AxesSubplot:xlabel='citric acid'>
```



Removing outliers

In [20]:

```

def Outliers(data, feature):
    IQ1 = data[feature].quantile(0.25)
    IQ3 = data[feature].quantile(0.75)
    IQR = IQ3 - IQ1

    lower_bound = IQ1 - 1.5 * IQR
    upper_bound = IQ3 + 1.5 * IQR

    index = data.index[ (data[feature] < lower_bound) | (data[feature] > upper_b
    return index
```

In [21]:

```
max(df.quality)

```

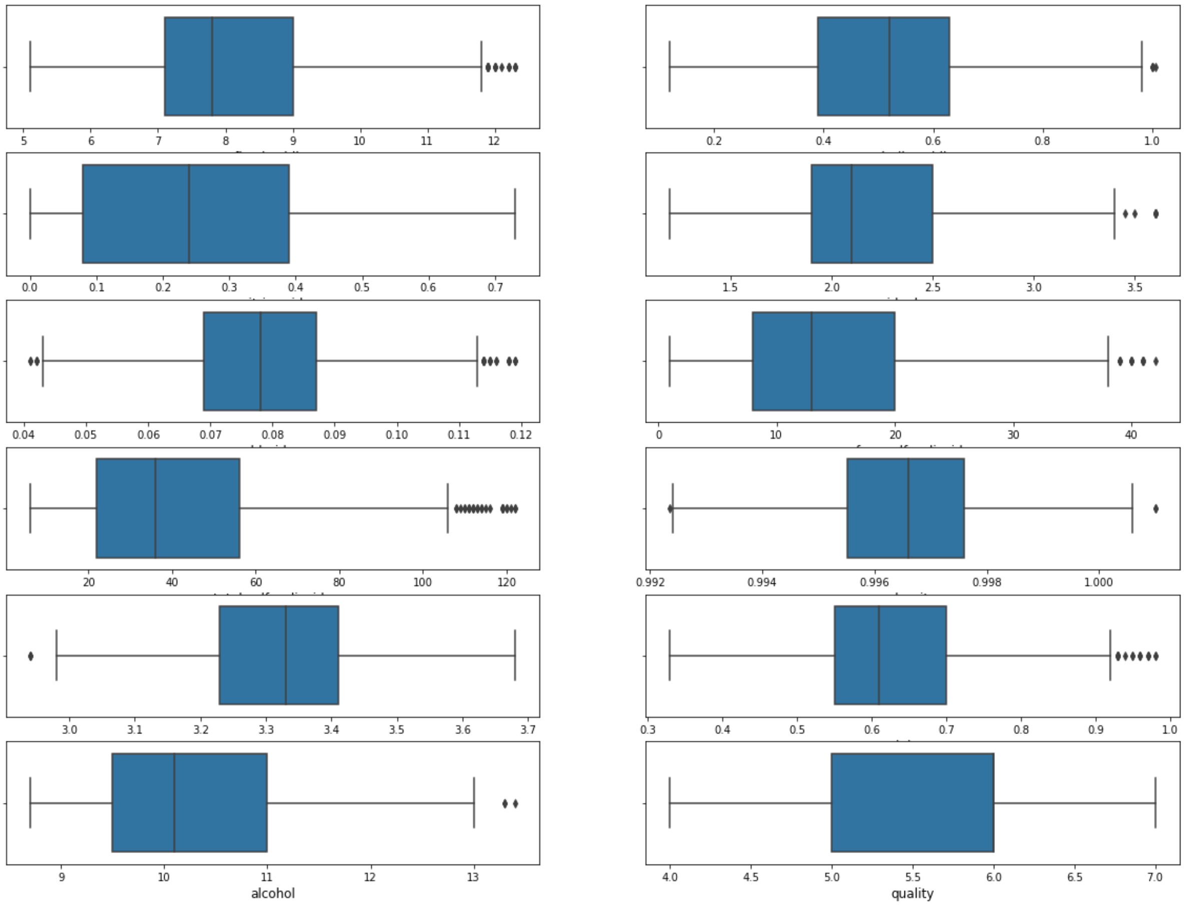
Out[21]:

```
7
```

In [22]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Checking for outliers
plt.figure(figsize = (20, 15))
for i in range (len(df.columns)):
    plt.subplot(6, 2, i+1)
    sns.boxplot(x = df.iloc[:, i])
    plt.xlabel(df.columns[i], size = 12)
```



Removing outliers and getting index of all the outliers

In [35]:

```python
# Removing outliers
def Outliers(data, feature):
    IQ1 = data[feature].quantile(0.25)
    IQ3 = data[feature].quantile(0.75)
    IQR = IQ3 - IQ1

    lower_bound = IQ1 - 1.5 * IQR
    upper_bound = IQ3 + 1.5 * IQR

    index = data.index[ (data[feature] < lower_bound) | (data[feature] > upper_b
    return index



# Getting index of all the outliers
index = []
for i in df.columns:
    index.extend(Outliers(df, i))

index = set(index)
print("Total number of outliers are {}".format(len(index)))

# Dropping all the outliers
df.drop(index, inplace = True, axis = 0)
df.shape
```

```
Total number of outliers are 2
```

Out[35]:

```
(923, 12)
```

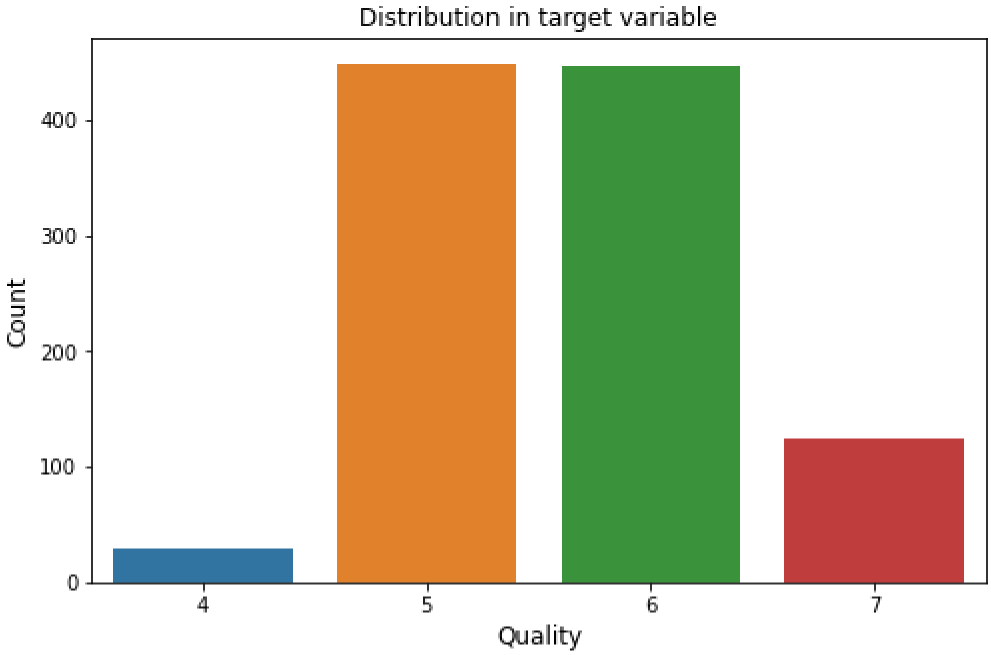
In [25]:

```
print(df['quality'].value_counts())
plt.figure(figsize = (8, 5))
sns.countplot(x = df['quality'])
plt.xlabel('Quality', size = 12)
plt.ylabel("Count", size = 12)
plt.title("Distribution in target variable", size = 12)
```

```
5    449
6    448
7    124
4     30
Name: quality, dtype: int64
```

Out[25]:

```
Text(0.5, 1.0, 'Distribution in target variable')
```



In [2]:

```
!pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /Users/tomisin/opt/anaconda
3/lib/python3.9/site-packages (3.3.3)
Requirement already satisfied: numpy in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from lightgbm) (1.20.3)
Requirement already satisfied: scikit-learn!=0.22.0 in /Users/tomisin/
opt/anaconda3/lib/python3.9/site-packages (from lightgbm) (0.24.2)
Requirement already satisfied: wheel in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from lightgbm) (0.37.0)
Requirement already satisfied: scipy in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from lightgbm) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/tomisin/
opt/anaconda3/lib/python3.9/site-packages (from scikit-learn!=0.22.0->
lightgbm) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /Users/tomisin/opt/anac
onda3/lib/python3.9/site-packages (from scikit-learn!=0.22.0->lightgb
m) (1.1.0)
```

In [3]:

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /Users/tomisin/opt/anaconda
3/lib/python3.9/site-packages (1.7.2)
Requirement already satisfied: numpy in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from xgboost) (1.7.1)
```

In [4]:

```
!pip install catboost
```

```
Requirement already satisfied: catboost in /Users/tomisin/opt/anaconda
3/lib/python3.9/site-packages (1.1.1)
Requirement already satisfied: six in /Users/tomisin/opt/anaconda3/li
b/python3.9/site-packages (from catboost) (1.16.0)
Requirement already satisfied: scipy in /Users/tomisin/opt/anaconda3/l
ib/python3.9/site-packages (from catboost) (1.7.1)
Requirement already satisfied: matplotlib in /Users/tomisin/opt/anacon
da3/lib/python3.9/site-packages (from catboost) (3.4.3)
Requirement already satisfied: graphviz in /Users/tomisin/opt/anaconda
3/lib/python3.9/site-packages (from catboost) (0.20.1)
Requirement already satisfied: pandas>=0.24.0 in /Users/tomisin/opt/an
aconda3/lib/python3.9/site-packages (from catboost) (1.3.4)
Requirement already satisfied: numpy>=1.16.0 in /Users/tomisin/opt/ana
conda3/lib/python3.9/site-packages (from catboost) (1.20.3)
Requirement already satisfied: plotly in /Users/tomisin/opt/anaconda3/
lib/python3.9/site-packages (from catboost) (5.11.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /Users/tomisi
n/opt/anaconda3/lib/python3.9/site-packages (from pandas>=0.24.0->catb
oost) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /Users/tomisin/opt/anac
onda3/lib/python3.9/site-packages (from pandas>=0.24.0->catboost) (202
1.3)
Requirement already satisfied: cycler>=0.10 in /Users/tomisin/opt/anac
onda3/lib/python3.9/site-packages (from matplotlib->catboost) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/tomisin/opt/
anaconda3/lib/python3.9/site-packages (from matplotlib->catboost) (3.
0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/tomisin/op
t/anaconda3/lib/python3.9/site-packages (from matplotlib->catboost)
(1.3.1)
Requirement already satisfied: pillow>=6.2.0 in /Users/tomisin/opt/ana
conda3/lib/python3.9/site-packages (from matplotlib->catboost) (8.4.0)
Requirement already satisfied: tenacity>=6.2.0 in /Users/tomisin/opt/a
naconda3/lib/python3.9/site-packages (from plotly->catboost) (8.1.0)
```

In [8]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, auc, roc_curve, roc_auc_score, mean_
```

In [41]:

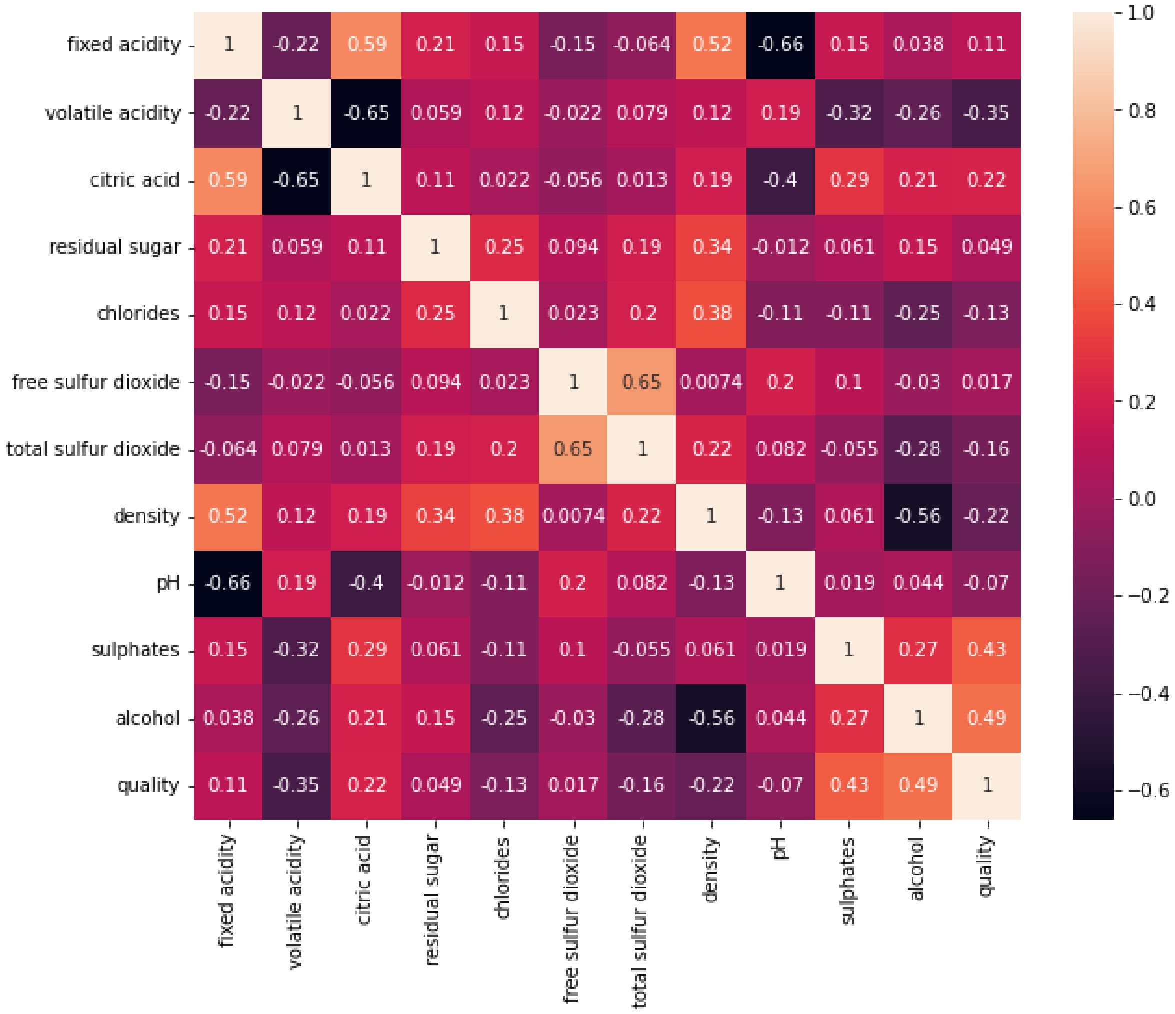
```python
data = df
```

In [42]:

```
cor = data.corr()
plt.figure(figsize = (10,8))
sns.heatmap(cor, annot = True)
```

Out[42]:

```
<AxesSubplot:>
```



In [43]:

```
x = data.iloc[:, :-1]
x.head()
```

Out[43]:

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

In [44]:

```
y = data.iloc[:, -1]
y.head()
```

Out[44]:

```
0    5
1    5
2    5
3    6
4    5
Name: quality, dtype: int64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: