Wine Quality Analysis and Prediction

In [1]:
```
#importing the Libery
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

Data Collection

In [2]:
```
#loding dataset in pandas dataframe
data = pd.read_csv('/content/winequality.csv')
```

In [3]:
```
#check first five rows of the dataset
data.head()
```

Out[3]:

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

In [4]:
```
#check last five rows of the dataset
data.tail()
```

Out[4]:

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

In [5]:
```
#check shape of the dataset
data.shape
```

Out[5]: (6497, 13)

In [6]:
```
#check infomation of the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   type                  6497 non-null   object 
 1   fixed acidity         6487 non-null   float64
 2   volatile acidity      6489 non-null   float64
 3   citric acid           6494 non-null   float64
 4   residual sugar        6495 non-null   float64
 5   chlorides             6495 non-null   float64
 6   free sulfur dioxide   6497 non-null   float64
 7   total sulfur dioxide  6497 non-null   float64
 8   density               6497 non-null   float64
 9   pH                    6488 non-null   float64
 10  sulphates             6493 non-null   float64
 11  alcohol               6497 non-null   float64
 12  quality               6497 non-null   int64  
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

In [7]:
```
#check columns of the dataset
data.columns
```

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

In [8]:
```
#check mathamatic describe
data.describe()
```

Out[8]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	tot
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	649
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	11
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	5
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	7
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	11
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	15
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	44

In [10]:
```
#check coreation of the dataset
data.corr()
```

Out[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	de
fixed acidity	1.000000	0.220172	0.323736	-0.112319	0.298421	-0.283317	-0.329747	0.45
volatile acidity	0.220172	1.000000	-0.378061	-0.196702	0.377167	-0.353230	-0.414928	0.27
citric acid	0.323736	-0.378061	1.000000	0.142486	0.039315	0.133437	0.195218	0.09
residual sugar	-0.112319	-0.196702	0.142486	1.000000	-0.128902	0.403439	0.495820	0.55
chlorides	0.298421	0.377167	0.039315	-0.128902	1.000000	-0.195042	-0.279580	0.36
free sulfur dioxide	-0.283317	-0.353230	0.133437	0.403439	-0.195042	1.000000	0.720934	0.02
total sulfur dioxide	-0.329747	-0.414928	0.195218	0.495820	-0.279580	0.720934	1.000000	0.03
density	0.459204	0.271193	0.096320	0.552498	0.362594	0.025717	0.032395	1.00
pH	-0.251814	0.260660	-0.328689	-0.267050	0.044806	-0.145191	-0.237687	0.01
sulphates	0.300380	0.225476	0.057613	-0.185745	0.395332	-0.188489	-0.275381	0.25
alcohol	-0.095603	-0.038248	-0.010433	-0.359706	-0.256861	-0.179838	-0.265740	-0.68
quality	-0.077031	-0.265953	0.085706	-0.036825	-0.200886	0.055463	-0.041385	-0.30

In [11]:
```
#check missing value of the dataset
data.isnull().sum()
```

Out[11]:
```
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 [12]:
```
#check duplicated value
data.duplicated().sum()
```

Out[12]:
```
1168
```

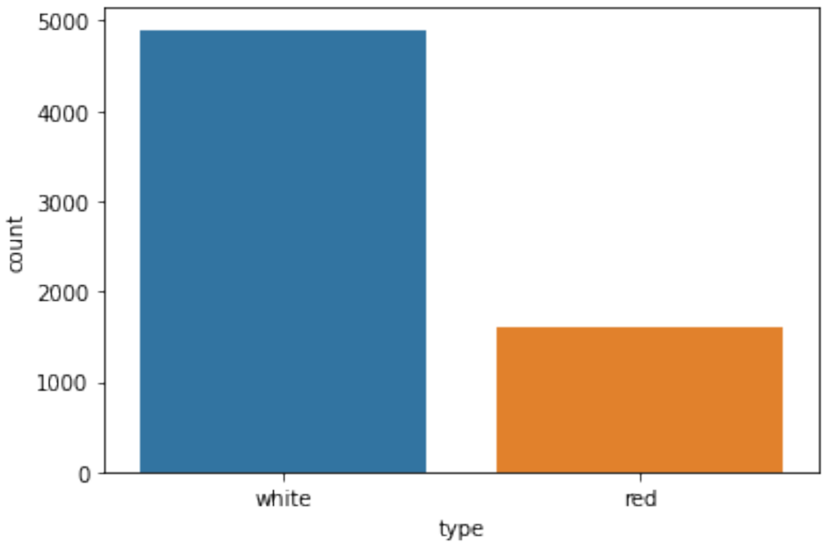
EDA of The Dataset

In [13]:
```
#count the value of type
data['type'].value_counts()
```

Out[13]:
```
white    4898
red      1599
Name: type, dtype: int64
```

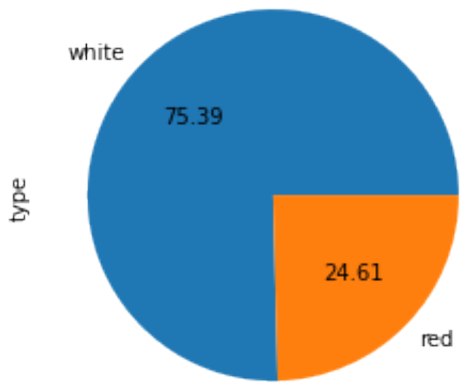
In [14]:
```
#plot conutplot
sns.countplot(data['type'])
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6c0137a60>



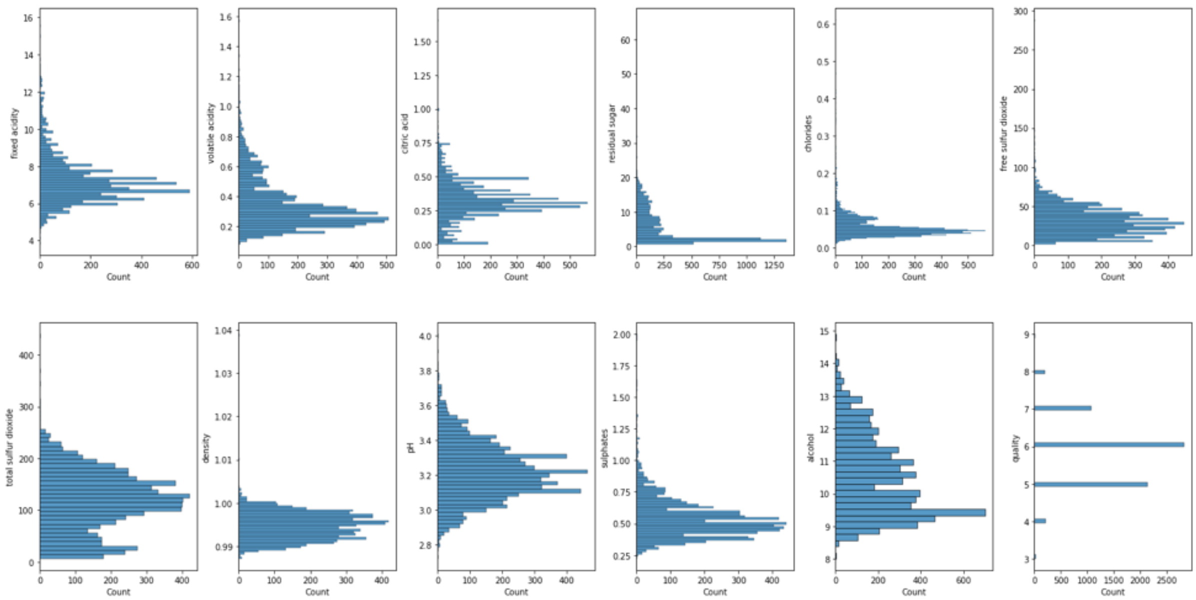
In [15]:
```
#plot pie plot
data['type'].value_counts().plot(kind='pie',autopct='%.2f')
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf8c8e50>
```



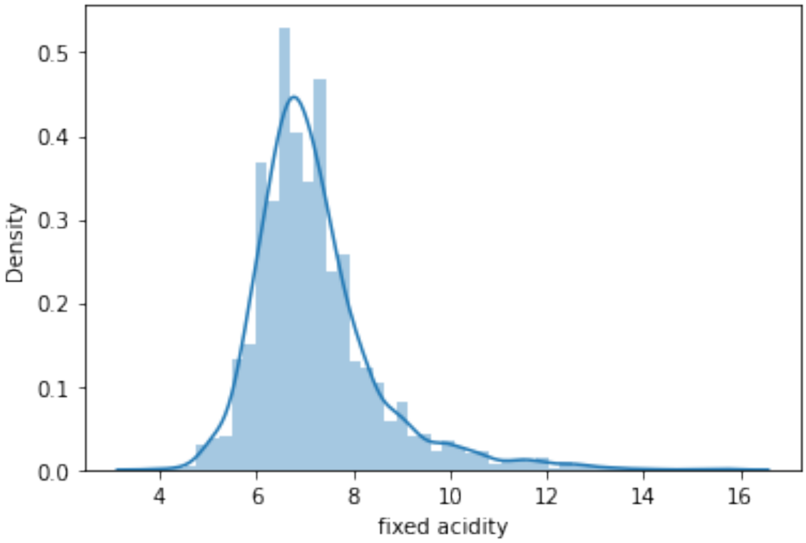
In [16]:
```
# create histplot plots
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in data.items():
    if col != 'type':
        sns.histplot(y=col, data=data, ax=ax[index])
        index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



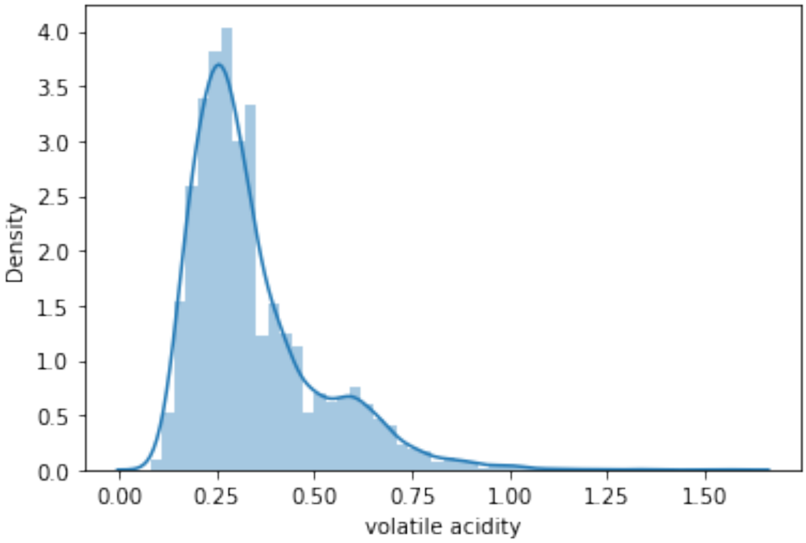
In [23]:
```
#distplot for fixed acidity
sns.distplot(data['fixed acidity'])
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bec86520>



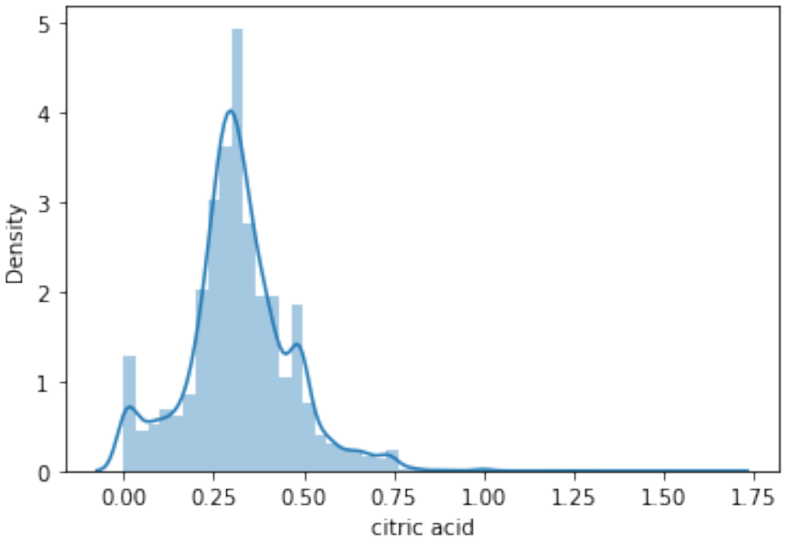
In [24]:
```
#distplot for volatile acidity
sns.distplot(data['volatile acidity'])
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bee81ac0>



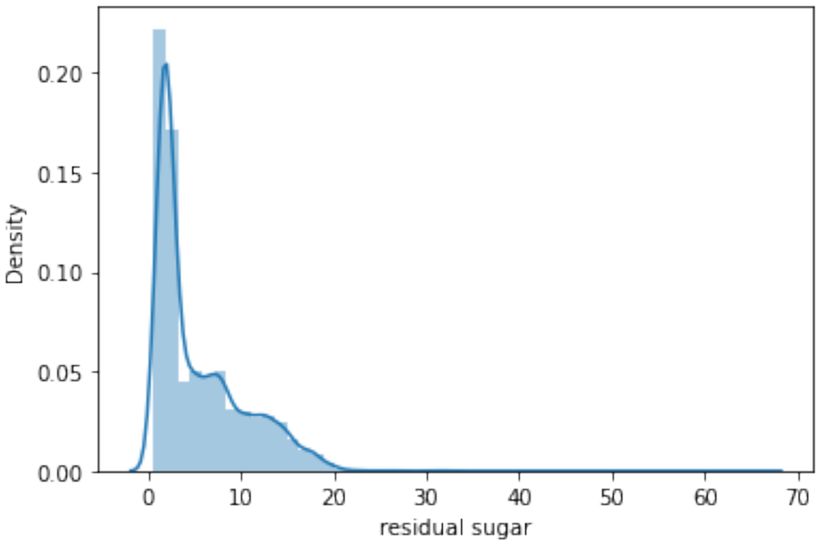
In [25]:
```
#distplot for citric acid
sns.distplot(data['citric acid'])
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf4aa970>



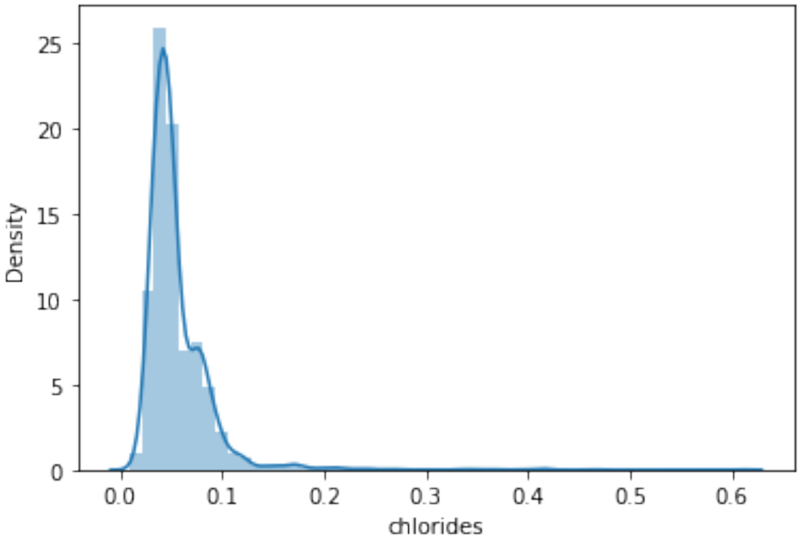
In [26]:
```
#distplot for residual sugar
sns.distplot(data['residual sugar'])
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf0d6fa0>



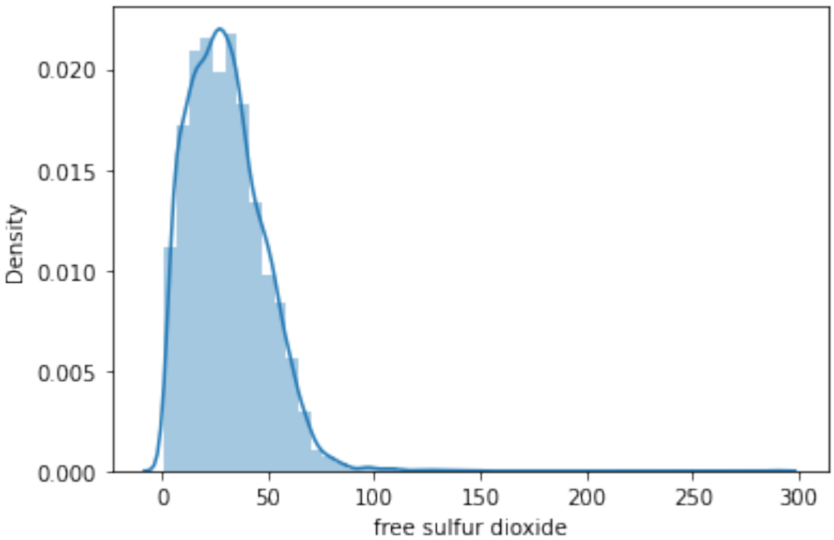
In [27]:
```
#distplot for chlorides
sns.distplot(data['chlorides'])
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf5531c0>



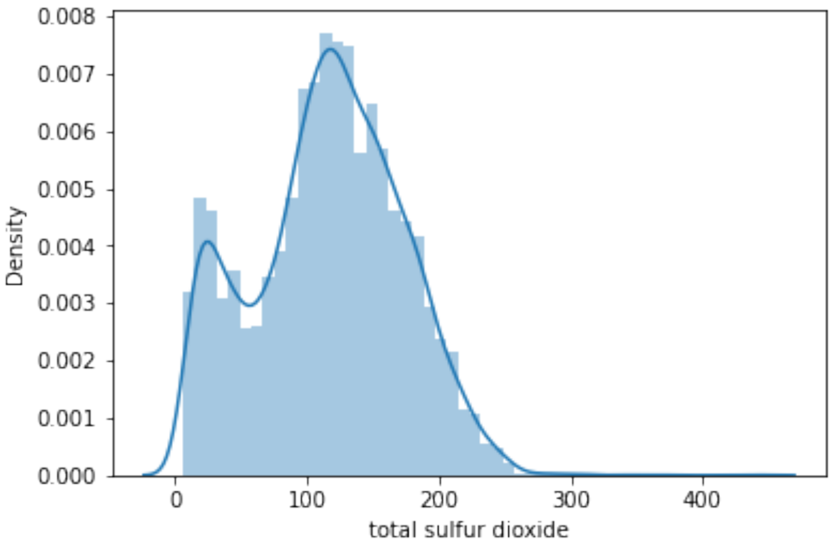
In [28]:
```
#distplot for free sulfur dioxide
sns.distplot(data['free sulfur dioxide'])
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf28f100>



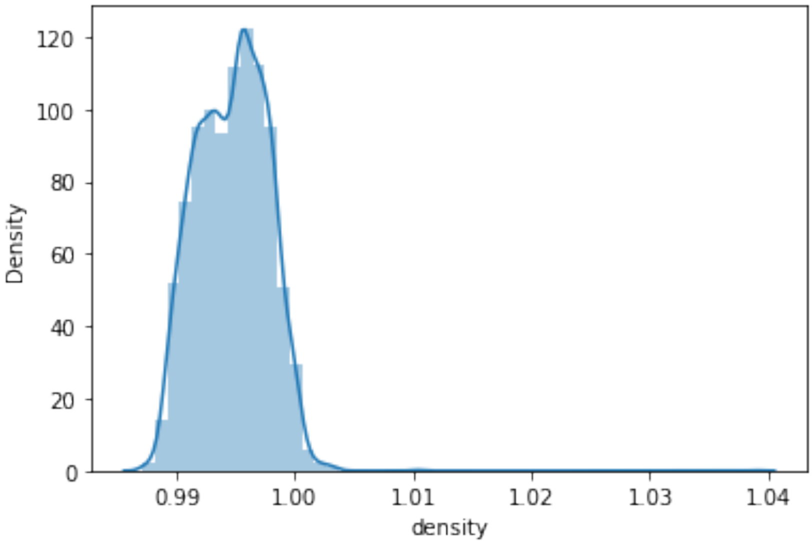
In [29]:
```
#distplot for total sulfur dioxide
sns.distplot(data['total sulfur dioxide'])
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf22d580>



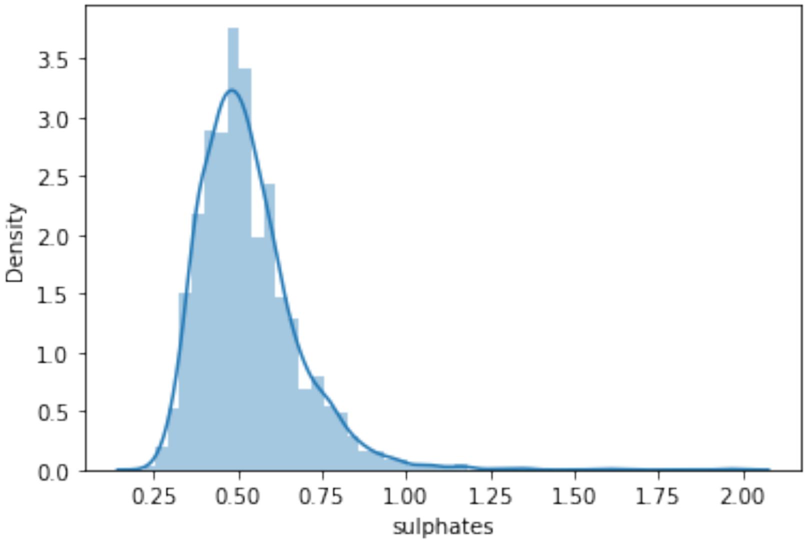
In [31]:
```
#distplot for density
sns.distplot(data['density'])
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bfd46a90>



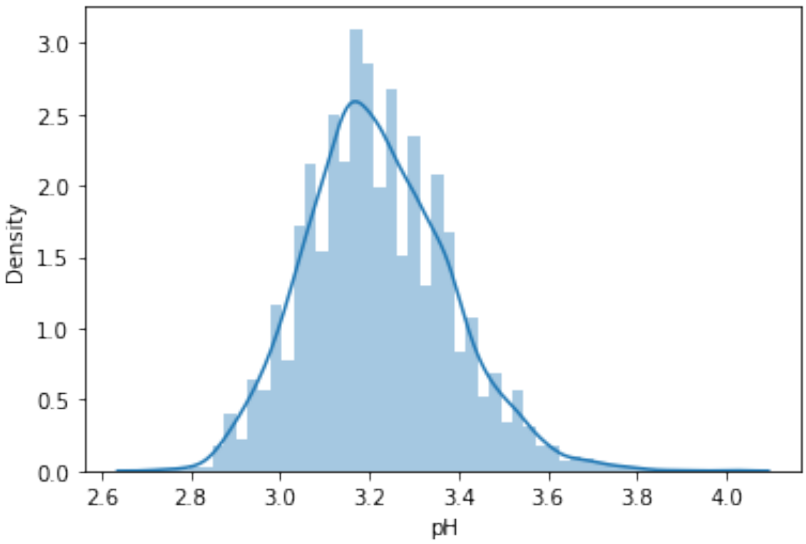
In [32]:
```
#distplot for sulphates
sns.distplot(data['sulphates'])
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6ba7148b0>



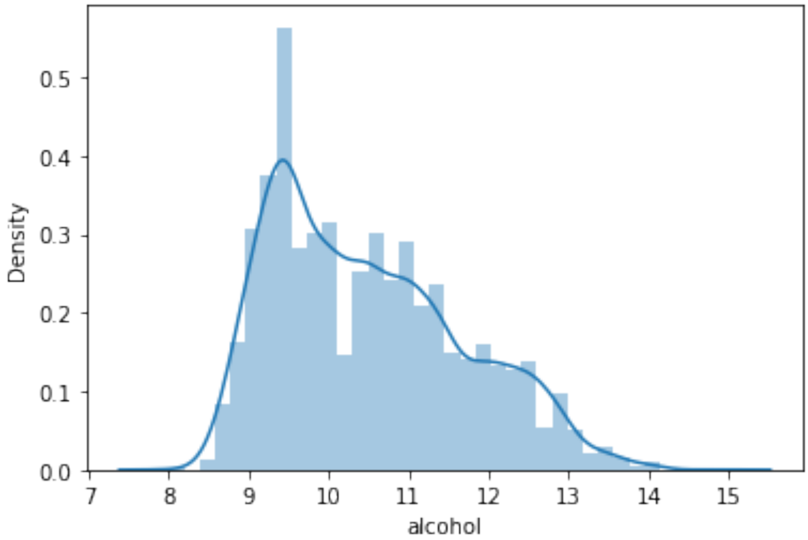
In [35]:
```
#distplot for PH
sns.distplot(data['pH'])
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6ba1dea90>



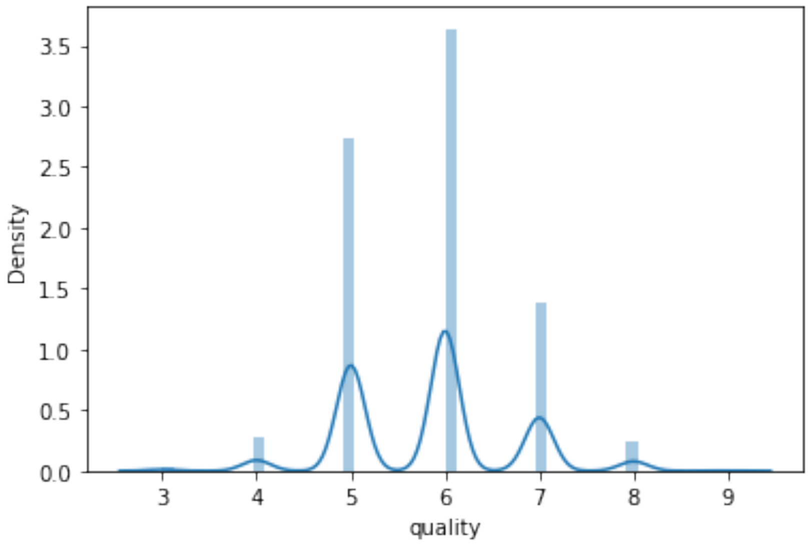
In [36]:
```
#distplot for alcohol
sns.distplot(data['alcohol'])
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bec61bb0>



In [37]:
```
#distplot for quality
sns.distplot(data['quality'])
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bed212b0>

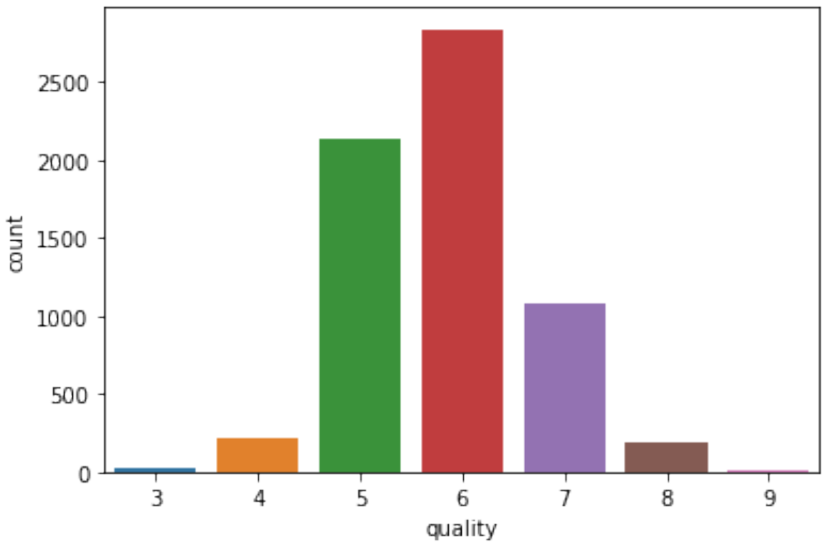


In [38]:
```
#count the value quality
data['quality'].value_counts()
```

Out[38]:
```
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

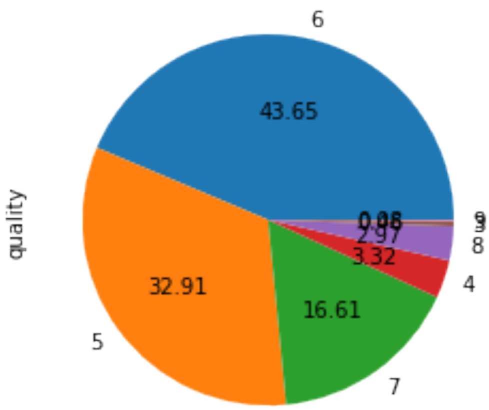
In [39]:
```
#countplot value quality
sns.countplot(data['quality'])
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6beca4160>



In [40]:
```
#plot pie plot
data['quality'].value_counts().plot(kind='pie',autopct='%.2f')
```

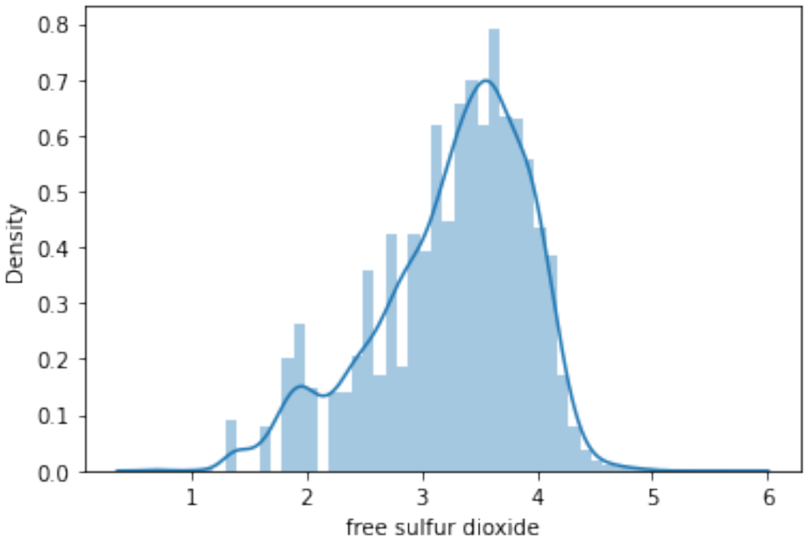
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf7aefd0>



In [41]:
```
# log transformation
data['free sulfur dioxide'] = np.log(1 + data['free sulfur dioxide'
```

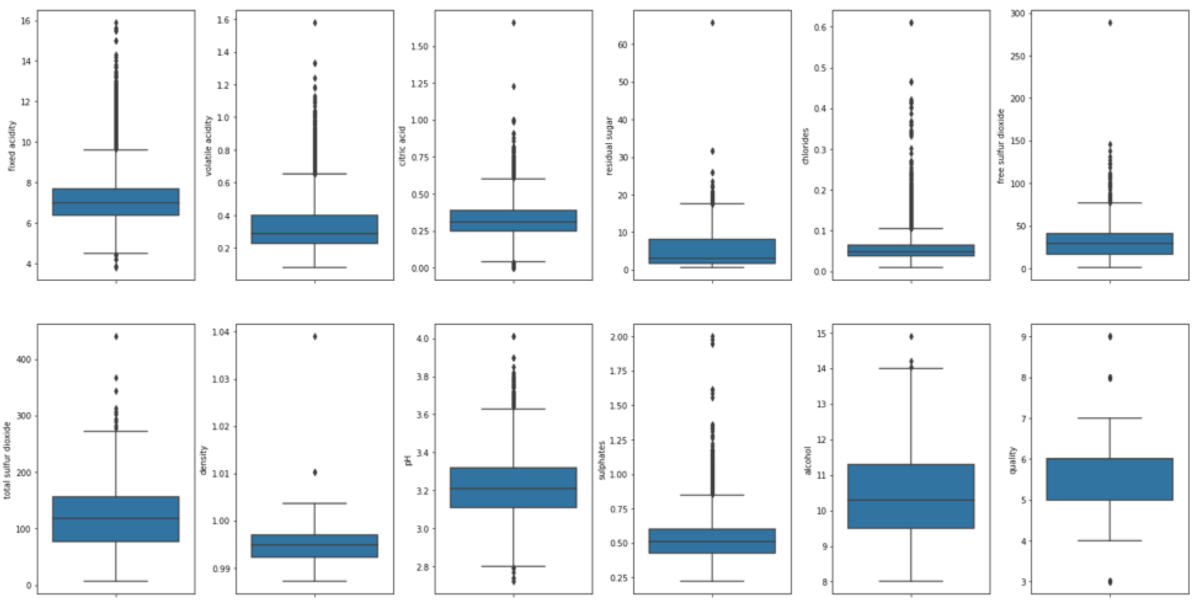
In [42]:
```
sns.distplot(data['free sulfur dioxide'])
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bc0dedc0>



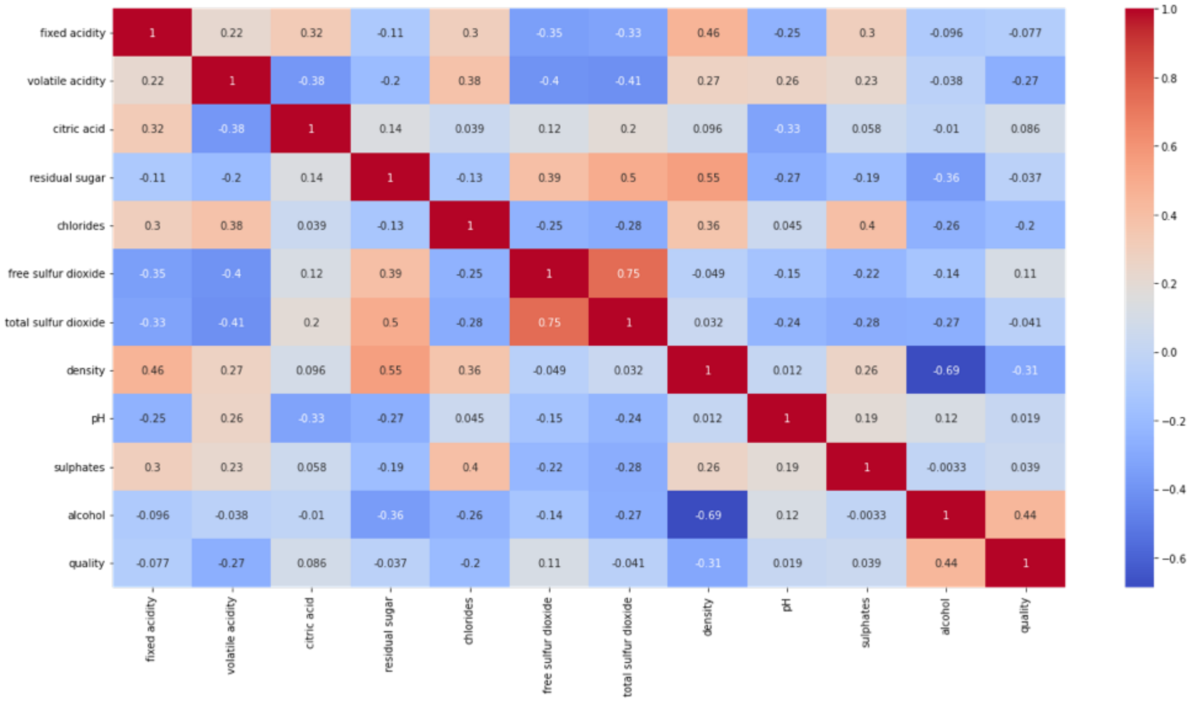
In [18]:
```
# create box plots
fig, ax = plt.subplots(ncols=6, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()

for col, value in data.items():
    if col != 'type':
        sns.boxplot(y=col, data=data, ax=ax[index])
        index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```



In [43]:
```
#correaction of the dataset
corr = data.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bf1856a0>

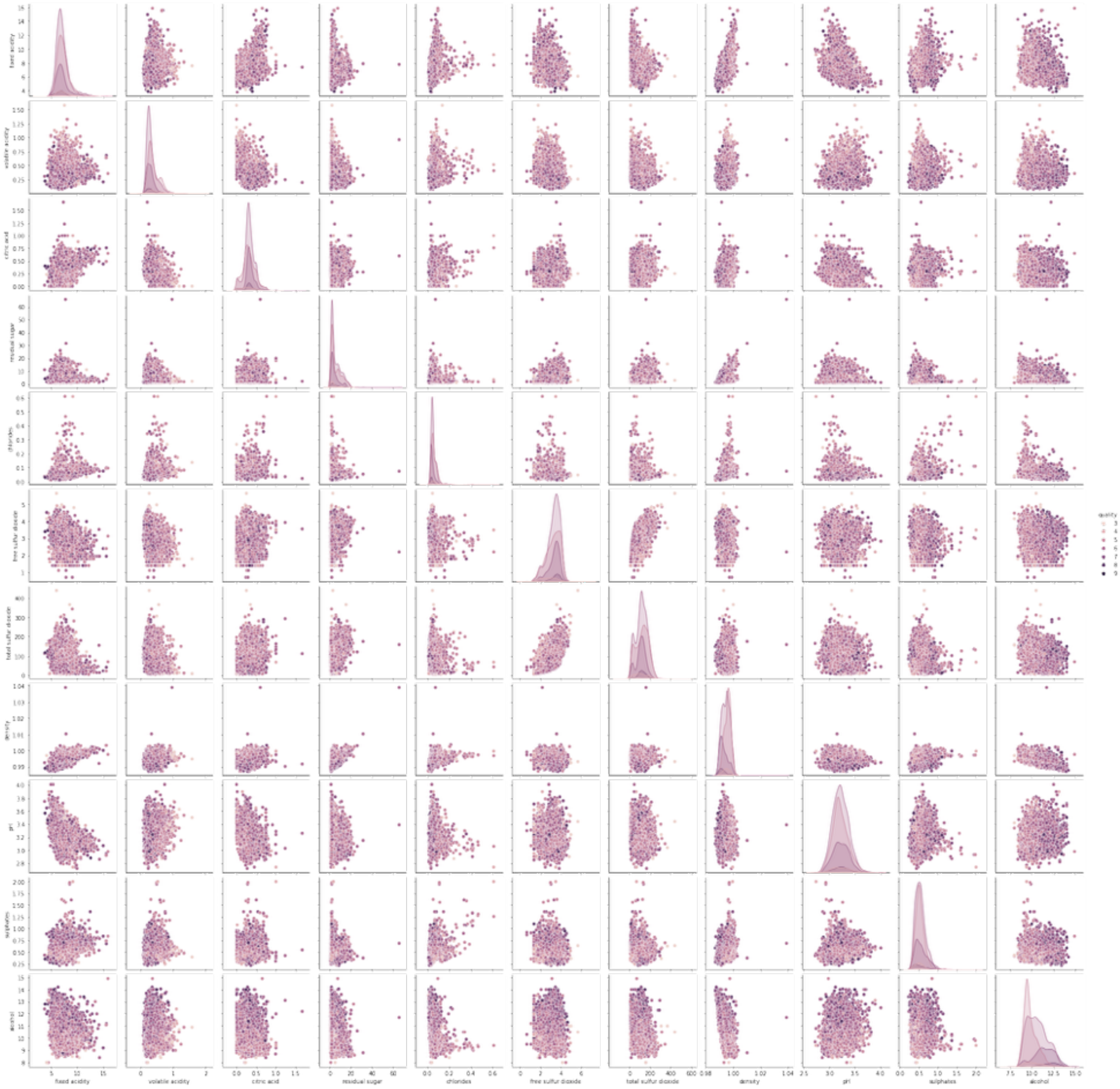


In [46]:
```
desc = data.describe()
plt.figure(figsize=(20,10))
sns.heatmap(desc, annot=True, cmap='coolwarm')
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe6bec22b50>

In [48]:
```
#pairplot of the dataset
sns.pairplot(data,hue='quality')
```

Out[48]: <seaborn.axisgrid.PairGrid at 0x7fe6b64a2460>



In [55]:
```
# fill the missing values
for col, value in data.items():
    if col != 'type':
        data[col] = data[col].fillna(data[col].mean())
```

In [56]:
```
#after fill missing value check missing value
data.isnull().sum()
```

Out[56]:
```
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 [57]:
```
#spliting the dataset in X and Y
X = data.drop(columns=['type', 'quality'])
Y = data['quality']
```

In [58]:
```
print(X)
print(Y)
```

```
      fixed acidity  volatile acidity  citric acid  residual sugar
chlorides  \
0               7.0             0.270         0.36            20.7
0.045
1               6.3             0.300         0.34             1.6
0.049
2               8.1             0.280         0.40             6.9
0.050
3               7.2             0.230         0.32             8.5
0.058
4               7.2             0.230         0.32             8.5
0.058
...             ...               ...          ...             ...
...
6492            6.2             0.600         0.08             2.0
0.090
6493            5.9             0.550         0.10             2.2
0.062
6494            6.3             0.510         0.13             2.3
0.076
6495            5.9             0.645         0.12             2.0
0.075
6496            6.0             0.310         0.47             3.6
0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  su
lphates  \
```

```
0               3.828641                   170.0  1.00100  3.00   0
.450000
1               2.708050                   132.0  0.99400  3.30   0
.490000
2               3.433987                    97.0  0.99510  3.26   0
.440000
3               3.871201                   186.0  0.99560  3.19   0
.400000
4               3.871201                   186.0  0.99560  3.19   0
.400000
...                  ...                     ...      ...   ...
...
6492            3.496508                    44.0  0.99490  3.45   0
.580000
6493            3.688879                    51.0  0.99512  3.52   0
.531215
6494            3.401197                    40.0  0.99574  3.42   0
.750000
6495            3.496508                    44.0  0.99547  3.57   0
.710000
6496            2.944439                    42.0  0.99549  3.39   0
.660000

      alcohol
0         8.8
1         9.5
2        10.1
3         9.9
4         9.9
...       ...
6492     10.5
6493     11.2
6494     11.0
6495     10.2
6496     11.0

[6497 rows x 11 columns]
0       6
1       6
2       6
3       6
4       6
       ..
6492    5
6493    6
6494    6
6495    5
6496    6
Name: quality, Length: 6497, dtype: int64
```

In [63]:
```
Y.value_counts()
```

Out[63]:
```
6    2836
5    2138
7    1079
4     216
8     193
3      30
9       5
Name: quality, dtype: int64
```

In [64]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=4)
# transform the dataset
X, Y = oversample.fit_resample(X, Y)
```

In [68]:
```
Y.value_counts()
```

Out[68]:
```
6    2836
5    2836
7    2836
8    2836
4    2836
3    2836
9    2836
Name: quality, dtype: int64
```

In [59]:
```
#spliting the dataset in X_train and Y_train
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [65]:
```
# split the data to train and test set
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,train_size=0.8
```

In [66]:
```
#print X_train and Y_train
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(16874, 11) (2978, 11) (16874,) (2978,)
```

In [67]:
```
#usingg standardscaler
scaler = StandardScaler()
X_train =scaler.fit_transform(X_train)
```

Model Training

In [69]:
```
# classify function
from sklearn.model_selection import cross_val_score, train_test_spl
def classify(model, X_train, Y_train):
    # train the model
    model.fit(X_train, Y_train)
    print("Accuracy:", model.score(X_test, Y_test) * 100)

    # cross-validation
    score = cross_val_score(model, X_train, Y_train, cv=5)
    print("CV Score:", np.mean(score)*100)
```

In [70]:
```
#using LogisticRegression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X_train, Y_train)
```

```
Accuracy: 8.059100067159166
CV Score: 52.275715383433216
```

In [71]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, X_train, Y_train)
```

```
Accuracy: 13.834788448623236
CV Score: 79.01506860743376
```

In [72]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model, X_train, Y_train)
```

```
Accuracy: 15.547347212894561
CV Score: 87.37703308524885
```

In [73]:
```
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
classify(model, X_train, Y_train)
```

```
Accuracy: 14.707857622565479
CV Score: 88.68078772311137
```

In [ ]: