Red Wine Quality

April 22, 2018

```
In [1]: import pandas as pd   # for data manipulation and analysis
        import numpy as np    # for scientific computing
        import matplotlib.pyplot as plt   # 2D plotting library
        %matplotlib inline
        from sklearn import tree    # sklearn is the machine learning library
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split  #for splitting data into train a
        import seaborn as sns                                 # seaborn is a visualization lib
        from sklearn.datasets import load_iris
        from sklearn import tree

        from sklearn.tree import export_graphviz
        import graphviz    # for graph visualization
        import pydotplus
        import io         #to access files and streams

        from scipy import misc   #for scientific computing
```

0.1 Importing -Red Wine Quality Dataset

```
In [2]: wine=pd.read_csv('./Red wine quality.csv',sep=',')  #Reading dataset
        df=pd.DataFrame(wine)                               #dataset converted to dataframe
        df                                                  #display dataframe
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.700	0.00	1.9	0.076	
1	7.8	0.880	0.00	2.6	0.098	
2	7.8	0.760	0.04	2.3	0.092	
3	11.2	0.280	0.56	1.9	0.075	
4	7.4	0.700	0.00	1.9	0.076	
5	7.4	0.660	0.00	1.8	0.075	
6	7.9	0.600	0.06	1.6	0.069	
7	7.3	0.650	0.00	1.2	0.065	
8	7.8	0.580	0.02	2.0	0.073	
9	7.5	0.500	0.36	6.1	0.071	
10	6.7	0.580	0.08	1.8	0.097	
11	7.5	0.500	0.36	6.1	0.071	

12	5.6	0.615	0.00	1.6	0.089
13	7.8	0.610	0.29	1.6	0.114
14	8.9	0.620	0.18	3.8	0.176
15	8.9	0.620	0.19	3.9	0.170
16	8.5	0.280	0.56	1.8	0.092
17	8.1	0.560	0.28	1.7	0.368
18	7.4	0.590	0.08	4.4	0.086
19	7.9	0.320	0.51	1.8	0.341
20	8.9	0.220	0.48	1.8	0.077
21	7.6	0.390	0.31	2.3	0.082
22	7.9	0.430	0.21	1.6	0.106
23	8.5	0.490	0.11	2.3	0.084
24	6.9	0.400	0.14	2.4	0.085
25	6.3	0.390	0.16	1.4	0.080
26	7.6	0.410	0.24	1.8	0.080
27	7.9	0.430	0.21	1.6	0.106
28	7.1	0.710	0.00	1.9	0.080
29	7.8	0.645	0.00	2.0	0.082
...	...	...	...	...	...
1569	6.2	0.510	0.14	1.9	0.056
1570	6.4	0.360	0.53	2.2	0.230
1571	6.4	0.380	0.14	2.2	0.038
1572	7.3	0.690	0.32	2.2	0.069
1573	6.0	0.580	0.20	2.4	0.075
1574	5.6	0.310	0.78	13.9	0.074
1575	7.5	0.520	0.40	2.2	0.060
1576	8.0	0.300	0.63	1.6	0.081
1577	6.2	0.700	0.15	5.1	0.076
1578	6.8	0.670	0.15	1.8	0.118
1579	6.2	0.560	0.09	1.7	0.053
1580	7.4	0.350	0.33	2.4	0.068
1581	6.2	0.560	0.09	1.7	0.053
1582	6.1	0.715	0.10	2.6	0.053
1583	6.2	0.460	0.29	2.1	0.074
1584	6.7	0.320	0.44	2.4	0.061
1585	7.2	0.390	0.44	2.6	0.066
1586	7.5	0.310	0.41	2.4	0.065
1587	5.8	0.610	0.11	1.8	0.066
1588	7.2	0.660	0.33	2.5	0.068
1589	6.6	0.725	0.20	7.8	0.073
1590	6.3	0.550	0.15	1.8	0.077
1591	5.4	0.740	0.09	1.7	0.089
1592	6.3	0.510	0.13	2.3	0.076
1593	6.8	0.620	0.08	1.9	0.068
1594	6.2	0.600	0.08	2.0	0.090
1595	5.9	0.550	0.10	2.2	0.062
1596	6.3	0.510	0.13	2.3	0.076
1597	5.9	0.645	0.12	2.0	0.075

```
1598            6.0             0.310         0.47             3.6       0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
5                    13.0                  40.0  0.99780  3.51       0.56
6                    15.0                  59.0  0.99640  3.30       0.46
7                    15.0                  21.0  0.99460  3.39       0.47
8                     9.0                  18.0  0.99680  3.36       0.57
9                    17.0                 102.0  0.99780  3.35       0.80
10                   15.0                  65.0  0.99590  3.28       0.54
11                   17.0                 102.0  0.99780  3.35       0.80
12                   16.0                  59.0  0.99430  3.58       0.52
13                    9.0                  29.0  0.99740  3.26       1.56
14                   52.0                 145.0  0.99860  3.16       0.88
15                   51.0                 148.0  0.99860  3.17       0.93
16                   35.0                 103.0  0.99690  3.30       0.75
17                   16.0                  56.0  0.99680  3.11       1.28
18                    6.0                  29.0  0.99740  3.38       0.50
19                   17.0                  56.0  0.99690  3.04       1.08
20                   29.0                  60.0  0.99680  3.39       0.53
21                   23.0                  71.0  0.99820  3.52       0.65
22                   10.0                  37.0  0.99660  3.17       0.91
23                    9.0                  67.0  0.99680  3.17       0.53
24                   21.0                  40.0  0.99680  3.43       0.63
25                   11.0                  23.0  0.99550  3.34       0.56
26                    4.0                  11.0  0.99620  3.28       0.59
27                   10.0                  37.0  0.99660  3.17       0.91
28                   14.0                  35.0  0.99720  3.47       0.55
29                    8.0                  16.0  0.99640  3.38       0.59
...                   ...                   ...      ...   ...        ...
1569                 15.0                  34.0  0.99396  3.48       0.57
1570                 19.0                  35.0  0.99340  3.37       0.93
1571                 15.0                  25.0  0.99514  3.44       0.65
1572                 35.0                 104.0  0.99632  3.33       0.51
1573                 15.0                  50.0  0.99467  3.58       0.67
1574                 23.0                  92.0  0.99677  3.39       0.48
1575                 12.0                  20.0  0.99474  3.26       0.64
1576                 16.0                  29.0  0.99588  3.30       0.78
1577                 13.0                  27.0  0.99622  3.54       0.60
1578                 13.0                  20.0  0.99540  3.42       0.67
1579                 24.0                  32.0  0.99402  3.54       0.60
1580                  9.0                  26.0  0.99470  3.36       0.60
1581                 24.0                  32.0  0.99402  3.54       0.60
1582                 13.0                  27.0  0.99362  3.57       0.50
```

1583	32.0	98.0	0.99578	3.33	0.62
1584	24.0	34.0	0.99484	3.29	0.80
1585	22.0	48.0	0.99494	3.30	0.84
1586	34.0	60.0	0.99492	3.34	0.85
1587	18.0	28.0	0.99483	3.55	0.66
1588	34.0	102.0	0.99414	3.27	0.78
1589	29.0	79.0	0.99770	3.29	0.54
1590	26.0	35.0	0.99314	3.32	0.82
1591	16.0	26.0	0.99402	3.67	0.56
1592	29.0	40.0	0.99574	3.42	0.75
1593	28.0	38.0	0.99651	3.42	0.82
1594	32.0	44.0	0.99490	3.45	0.58
1595	39.0	51.0	0.99512	3.52	0.76
1596	29.0	40.0	0.99574	3.42	0.75
1597	32.0	44.0	0.99547	3.57	0.71
1598	18.0	42.0	0.99549	3.39	0.66

	alcohol	quality
0	9.4	5
1	9.8	5
2	9.8	5
3	9.8	6
4	9.4	5
5	9.4	5
6	9.4	5
7	10.0	7
8	9.5	7
9	10.5	5
10	9.2	5
11	10.5	5
12	9.9	5
13	9.1	5
14	9.2	5
15	9.2	5
16	10.5	7
17	9.3	5
18	9.0	4
19	9.2	6
20	9.4	6
21	9.7	5
22	9.5	5
23	9.4	5
24	9.7	6
25	9.3	5
26	9.5	5
27	9.5	5
28	9.4	5
29	9.8	6

```
...       ...       ...
1569     11.5         6
1570     12.4         6
1571     11.1         6
1572      9.5         5
1573     12.5         6
1574     10.5         6
1575     11.8         6
1576     10.8         6
1577     11.9         6
1578     11.3         6
1579     11.3         5
1580     11.9         6
1581     11.3         5
1582     11.9         5
1583      9.8         5
1584     11.6         7
1585     11.5         6
1586     11.4         6
1587     10.9         6
1588     12.8         6
1589      9.2         5
1590     11.6         6
1591     11.6         6
1592     11.0         6
1593      9.5         6
1594     10.5         5
1595     11.2         6
1596     11.0         6
1597     10.2         5
1598     11.0         6

[1599 rows x 12 columns]
```

```
In [3]: df.shape   #Gives the number of rows and columns in a dataframe
```

```
Out[3]: (1599, 12)
```

```
In [4]: df.head()   #gives the first five rows of the dataframe
```

```
Out[4]:    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
```

```
1                 25.0                 67.0    0.9968  3.20       0.68
2                 15.0                 54.0    0.9970  3.26       0.65
3                 17.0                 60.0    0.9980  3.16       0.58
4                 11.0                 34.0    0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

0.2 Data Cleaning Steps

```
In [5]: df.isnull().any()  #To check if any of the columns of the dataframe have null values-
                           #this dataframe has no null values

Out[5]: fixed acidity           False
        volatile acidity        False
        citric acid             False
        residual sugar          False
        chlorides               False
        free sulfur dioxide     False
        total sulfur dioxide    False
        density                 False
        pH                      False
        sulphates               False
        alcohol                 False
        quality                 False
        dtype: bool

In [218]: df.duplicated() # To check for duplicated rows in the dataframe-
                          #it can be seen that their are quite a few duplicate rows

Out[218]: 0       False
          1       False
          2       False
          3       False
          5       False
          6       False
          7       False
          8       False
          9       False
          10      False
          12      False
          13      False
          14      False
          15      False
          16      False
```

```
17      False
18      False
19      False
20      False
21      False
22      False
23      False
24      False
25      False
26      False
28      False
29      False
30      False
31      False
32      False
        ...
1566    False
1568    False
1569    False
1570    False
1571    False
1572    False
1573    False
1574    False
1575    False
1576    False
1577    False
1578    False
1579    False
1580    False
1582    False
1583    False
1584    False
1585    False
1586    False
1587    False
1588    False
1589    False
1590    False
1591    False
1592    False
1593    False
1594    False
1595    False
1597    False
1598    False
Length: 1359, dtype: bool
```

```
In [7]: np.dtype(df.chlorides)
        np.dtype(df.quality)   #to check the data type of a column

Out[7]: dtype('int64')

In [8]: df.dtypes   #gives the data types of all columns in a table

Out[8]: fixed acidity           float64
        volatile acidity        float64
        citric acid             float64
        residual sugar          float64
        chlorides               float64
        free sulfur dioxide     float64
        total sulfur dioxide    float64
        density                 float64
        pH                      float64
        sulphates               float64
        alcohol                 float64
        quality                   int64
        dtype: object

In [9]: df.columns  #displays names of all the columns in the dataframe

Out[9]: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
               'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
               'pH', 'sulphates', 'alcohol', 'quality'],
              dtype='object')

In [10]: df.describe() #Generates descriptive statistics that summarize the
         #central tendency, dispersion and shape of a datasets distribution, excluding NaN val

Out[10]:        fixed acidity  volatile acidity  citric acid  residual sugar  \
         count    1599.000000       1599.000000  1599.000000     1599.000000
         mean        8.319637          0.527821     0.270976        2.538806
         std         1.741096          0.179060     0.194801        1.409928
         min         4.600000          0.120000     0.000000        0.900000
         25%         7.100000          0.390000     0.090000        1.900000
         50%         7.900000          0.520000     0.260000        2.200000
         75%         9.200000          0.640000     0.420000        2.600000
         max        15.900000          1.580000     1.000000       15.500000

                   chlorides  free sulfur dioxide  total sulfur dioxide      density  \
         count  1599.000000          1599.000000           1599.000000  1599.000000
         mean      0.087467            15.874922             46.467792     0.996747
         std       0.047065            10.460157             32.895324     0.001887
         min       0.012000             1.000000              6.000000     0.990070
         25%       0.070000             7.000000             22.000000     0.995600
         50%       0.079000            14.000000             38.000000     0.996750
         75%       0.090000            21.000000             62.000000     0.997835
```

```
max       0.611000            72.000000            289.000000     1.003690

                pH    sulphates      alcohol      quality
count  1599.000000  1599.000000  1599.000000  1599.000000
mean      3.311113     0.658149    10.422983     5.636023
std       0.154386     0.169507     1.065668     0.807569
min       2.740000     0.330000     8.400000     3.000000
25%       3.210000     0.550000     9.500000     5.000000
50%       3.310000     0.620000    10.200000     6.000000
75%       3.400000     0.730000    11.100000     6.000000
max       4.010000     2.000000    14.900000     8.000000
```

```
In [11]: df.quality.min() # to find the minimum value of the target variable(quality)

Out[11]: 3

In [12]: df.quality.max() # to find the maximum value of the target variable(quality)

Out[12]: 8
```

Replacing the column values of 'quality' with a new set of values

```
In [13]: df['quality'].replace((3),1,inplace=True)
         df['quality'].replace((4),2,inplace=True)
         df['quality'].replace((5),3,inplace=True)
         df['quality'].replace((6),4,inplace=True)
         df['quality'].replace((7),5,inplace=True)
         df['quality'].replace((8),6,inplace=True)

In [14]: df

Out[14]:       fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
         0               7.4             0.700         0.00             1.9      0.076
         1               7.8             0.880         0.00             2.6      0.098
         2               7.8             0.760         0.04             2.3      0.092
         3              11.2             0.280         0.56             1.9      0.075
         4               7.4             0.700         0.00             1.9      0.076
         5               7.4             0.660         0.00             1.8      0.075
         6               7.9             0.600         0.06             1.6      0.069
         7               7.3             0.650         0.00             1.2      0.065
         8               7.8             0.580         0.02             2.0      0.073
         9               7.5             0.500         0.36             6.1      0.071
         10              6.7             0.580         0.08             1.8      0.097
         11              7.5             0.500         0.36             6.1      0.071
         12              5.6             0.615         0.00             1.6      0.089
         13              7.8             0.610         0.29             1.6      0.114
         14              8.9             0.620         0.18             3.8      0.176
         15              8.9             0.620         0.19             3.9      0.170
         16              8.5             0.280         0.56             1.8      0.092
```

17	8.1	0.560	0.28	1.7	0.368
18	7.4	0.590	0.08	4.4	0.086
19	7.9	0.320	0.51	1.8	0.341
20	8.9	0.220	0.48	1.8	0.077
21	7.6	0.390	0.31	2.3	0.082
22	7.9	0.430	0.21	1.6	0.106
23	8.5	0.490	0.11	2.3	0.084
24	6.9	0.400	0.14	2.4	0.085
25	6.3	0.390	0.16	1.4	0.080
26	7.6	0.410	0.24	1.8	0.080
27	7.9	0.430	0.21	1.6	0.106
28	7.1	0.710	0.00	1.9	0.080
29	7.8	0.645	0.00	2.0	0.082
...	...	...	...	...	...
1569	6.2	0.510	0.14	1.9	0.056
1570	6.4	0.360	0.53	2.2	0.230
1571	6.4	0.380	0.14	2.2	0.038
1572	7.3	0.690	0.32	2.2	0.069
1573	6.0	0.580	0.20	2.4	0.075
1574	5.6	0.310	0.78	13.9	0.074
1575	7.5	0.520	0.40	2.2	0.060
1576	8.0	0.300	0.63	1.6	0.081
1577	6.2	0.700	0.15	5.1	0.076
1578	6.8	0.670	0.15	1.8	0.118
1579	6.2	0.560	0.09	1.7	0.053
1580	7.4	0.350	0.33	2.4	0.068
1581	6.2	0.560	0.09	1.7	0.053
1582	6.1	0.715	0.10	2.6	0.053
1583	6.2	0.460	0.29	2.1	0.074
1584	6.7	0.320	0.44	2.4	0.061
1585	7.2	0.390	0.44	2.6	0.066
1586	7.5	0.310	0.41	2.4	0.065
1587	5.8	0.610	0.11	1.8	0.066
1588	7.2	0.660	0.33	2.5	0.068
1589	6.6	0.725	0.20	7.8	0.073
1590	6.3	0.550	0.15	1.8	0.077
1591	5.4	0.740	0.09	1.7	0.089
1592	6.3	0.510	0.13	2.3	0.076
1593	6.8	0.620	0.08	1.9	0.068
1594	6.2	0.600	0.08	2.0	0.090
1595	5.9	0.550	0.10	2.2	0.062
1596	6.3	0.510	0.13	2.3	0.076
1597	5.9	0.645	0.12	2.0	0.075
1598	6.0	0.310	0.47	3.6	0.067

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.99780	3.51	0.56	
1	25.0	67.0	0.99680	3.20	0.68	

2	15.0	54.0	0.99700	3.26	0.65
3	17.0	60.0	0.99800	3.16	0.58
4	11.0	34.0	0.99780	3.51	0.56
5	13.0	40.0	0.99780	3.51	0.56
6	15.0	59.0	0.99640	3.30	0.46
7	15.0	21.0	0.99460	3.39	0.47
8	9.0	18.0	0.99680	3.36	0.57
9	17.0	102.0	0.99780	3.35	0.80
10	15.0	65.0	0.99590	3.28	0.54
11	17.0	102.0	0.99780	3.35	0.80
12	16.0	59.0	0.99430	3.58	0.52
13	9.0	29.0	0.99740	3.26	1.56
14	52.0	145.0	0.99860	3.16	0.88
15	51.0	148.0	0.99860	3.17	0.93
16	35.0	103.0	0.99690	3.30	0.75
17	16.0	56.0	0.99680	3.11	1.28
18	6.0	29.0	0.99740	3.38	0.50
19	17.0	56.0	0.99690	3.04	1.08
20	29.0	60.0	0.99680	3.39	0.53
21	23.0	71.0	0.99820	3.52	0.65
22	10.0	37.0	0.99660	3.17	0.91
23	9.0	67.0	0.99680	3.17	0.53
24	21.0	40.0	0.99680	3.43	0.63
25	11.0	23.0	0.99550	3.34	0.56
26	4.0	11.0	0.99620	3.28	0.59
27	10.0	37.0	0.99660	3.17	0.91
28	14.0	35.0	0.99720	3.47	0.55
29	8.0	16.0	0.99640	3.38	0.59
...	...	...	...	...	...
1569	15.0	34.0	0.99396	3.48	0.57
1570	19.0	35.0	0.99340	3.37	0.93
1571	15.0	25.0	0.99514	3.44	0.65
1572	35.0	104.0	0.99632	3.33	0.51
1573	15.0	50.0	0.99467	3.58	0.67
1574	23.0	92.0	0.99677	3.39	0.48
1575	12.0	20.0	0.99474	3.26	0.64
1576	16.0	29.0	0.99588	3.30	0.78
1577	13.0	27.0	0.99622	3.54	0.60
1578	13.0	20.0	0.99540	3.42	0.67
1579	24.0	32.0	0.99402	3.54	0.60
1580	9.0	26.0	0.99470	3.36	0.60
1581	24.0	32.0	0.99402	3.54	0.60
1582	13.0	27.0	0.99362	3.57	0.50
1583	32.0	98.0	0.99578	3.33	0.62
1584	24.0	34.0	0.99484	3.29	0.80
1585	22.0	48.0	0.99494	3.30	0.84
1586	34.0	60.0	0.99492	3.34	0.85
1587	18.0	28.0	0.99483	3.55	0.66

1588	34.0	102.0	0.99414	3.27	0.78
1589	29.0	79.0	0.99770	3.29	0.54
1590	26.0	35.0	0.99314	3.32	0.82
1591	16.0	26.0	0.99402	3.67	0.56
1592	29.0	40.0	0.99574	3.42	0.75
1593	28.0	38.0	0.99651	3.42	0.82
1594	32.0	44.0	0.99490	3.45	0.58
1595	39.0	51.0	0.99512	3.52	0.76
1596	29.0	40.0	0.99574	3.42	0.75
1597	32.0	44.0	0.99547	3.57	0.71
1598	18.0	42.0	0.99549	3.39	0.66

	alcohol	quality
0	9.4	3
1	9.8	3
2	9.8	3
3	9.8	4
4	9.4	3
5	9.4	3
6	9.4	3
7	10.0	5
8	9.5	5
9	10.5	3
10	9.2	3
11	10.5	3
12	9.9	3
13	9.1	3
14	9.2	3
15	9.2	3
16	10.5	5
17	9.3	3
18	9.0	2
19	9.2	4
20	9.4	4
21	9.7	3
22	9.5	3
23	9.4	3
24	9.7	4
25	9.3	3
26	9.5	3
27	9.5	3
28	9.4	3
29	9.8	4
...	...	...
1569	11.5	4
1570	12.4	4
1571	11.1	4
1572	9.5	3

```
1573     12.5        4
1574     10.5        4
1575     11.8        4
1576     10.8        4
1577     11.9        4
1578     11.3        4
1579     11.3        3
1580     11.9        4
1581     11.3        3
1582     11.9        3
1583      9.8        3
1584     11.6        5
1585     11.5        4
1586     11.4        4
1587     10.9        4
1588     12.8        4
1589      9.2        3
1590     11.6        4
1591     11.6        4
1592     11.0        4
1593      9.5        4
1594     10.5        3
1595     11.2        4
1596     11.0        4
1597     10.2        3
1598     11.0        4

[1599 rows x 12 columns]
```

```
In [15]: df.quality.min() #new minimum of target variable
```

```
Out[15]: 1
```

```
In [16]: df.quality.max() #new maximum of target variable
```

```
Out[16]: 6
```

```
In [17]: df.shape  #to get the number of rows and columns originally
```

```
Out[17]: (1599, 12)
```

```
In [18]: df=df.drop_duplicates() #to drop the duplicate rows
```

```
In [19]: df.shape  #to get the number of rows and columns after removing duplicates
```

```
Out[19]: (1359, 12)
```

```
In [20]: df
```

Out[20]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	7.4	0.700	0.00	1.9	0.076	
1	7.8	0.880	0.00	2.6	0.098	
2	7.8	0.760	0.04	2.3	0.092	
3	11.2	0.280	0.56	1.9	0.075	
5	7.4	0.660	0.00	1.8	0.075	
6	7.9	0.600	0.06	1.6	0.069	
7	7.3	0.650	0.00	1.2	0.065	
8	7.8	0.580	0.02	2.0	0.073	
9	7.5	0.500	0.36	6.1	0.071	
10	6.7	0.580	0.08	1.8	0.097	
12	5.6	0.615	0.00	1.6	0.089	
13	7.8	0.610	0.29	1.6	0.114	
14	8.9	0.620	0.18	3.8	0.176	
15	8.9	0.620	0.19	3.9	0.170	
16	8.5	0.280	0.56	1.8	0.092	
17	8.1	0.560	0.28	1.7	0.368	
18	7.4	0.590	0.08	4.4	0.086	
19	7.9	0.320	0.51	1.8	0.341	
20	8.9	0.220	0.48	1.8	0.077	
21	7.6	0.390	0.31	2.3	0.082	
22	7.9	0.430	0.21	1.6	0.106	
23	8.5	0.490	0.11	2.3	0.084	
24	6.9	0.400	0.14	2.4	0.085	
25	6.3	0.390	0.16	1.4	0.080	
26	7.6	0.410	0.24	1.8	0.080	
28	7.1	0.710	0.00	1.9	0.080	
29	7.8	0.645	0.00	2.0	0.082	
30	6.7	0.675	0.07	2.4	0.089	
31	6.9	0.685	0.00	2.5	0.105	
32	8.3	0.655	0.12	2.3	0.083	
...	...	...	...	...	...	
1566	6.7	0.160	0.64	2.1	0.059	
1568	7.0	0.560	0.13	1.6	0.077	
1569	6.2	0.510	0.14	1.9	0.056	
1570	6.4	0.360	0.53	2.2	0.230	
1571	6.4	0.380	0.14	2.2	0.038	
1572	7.3	0.690	0.32	2.2	0.069	
1573	6.0	0.580	0.20	2.4	0.075	
1574	5.6	0.310	0.78	13.9	0.074	
1575	7.5	0.520	0.40	2.2	0.060	
1576	8.0	0.300	0.63	1.6	0.081	
1577	6.2	0.700	0.15	5.1	0.076	
1578	6.8	0.670	0.15	1.8	0.118	
1579	6.2	0.560	0.09	1.7	0.053	
1580	7.4	0.350	0.33	2.4	0.068	
1582	6.1	0.715	0.10	2.6	0.053	
1583	6.2	0.460	0.29	2.1	0.074	

1584	6.7	0.320	0.44	2.4	0.061
1585	7.2	0.390	0.44	2.6	0.066
1586	7.5	0.310	0.41	2.4	0.065
1587	5.8	0.610	0.11	1.8	0.066
1588	7.2	0.660	0.33	2.5	0.068
1589	6.6	0.725	0.20	7.8	0.073
1590	6.3	0.550	0.15	1.8	0.077
1591	5.4	0.740	0.09	1.7	0.089
1592	6.3	0.510	0.13	2.3	0.076
1593	6.8	0.620	0.08	1.9	0.068
1594	6.2	0.600	0.08	2.0	0.090
1595	5.9	0.550	0.10	2.2	0.062
1597	5.9	0.645	0.12	2.0	0.075
1598	6.0	0.310	0.47	3.6	0.067

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	11.0	34.0	0.99780	3.51	0.56	
1	25.0	67.0	0.99680	3.20	0.68	
2	15.0	54.0	0.99700	3.26	0.65	
3	17.0	60.0	0.99800	3.16	0.58	
5	13.0	40.0	0.99780	3.51	0.56	
6	15.0	59.0	0.99640	3.30	0.46	
7	15.0	21.0	0.99460	3.39	0.47	
8	9.0	18.0	0.99680	3.36	0.57	
9	17.0	102.0	0.99780	3.35	0.80	
10	15.0	65.0	0.99590	3.28	0.54	
12	16.0	59.0	0.99430	3.58	0.52	
13	9.0	29.0	0.99740	3.26	1.56	
14	52.0	145.0	0.99860	3.16	0.88	
15	51.0	148.0	0.99860	3.17	0.93	
16	35.0	103.0	0.99690	3.30	0.75	
17	16.0	56.0	0.99680	3.11	1.28	
18	6.0	29.0	0.99740	3.38	0.50	
19	17.0	56.0	0.99690	3.04	1.08	
20	29.0	60.0	0.99680	3.39	0.53	
21	23.0	71.0	0.99820	3.52	0.65	
22	10.0	37.0	0.99660	3.17	0.91	
23	9.0	67.0	0.99680	3.17	0.53	
24	21.0	40.0	0.99680	3.43	0.63	
25	11.0	23.0	0.99550	3.34	0.56	
26	4.0	11.0	0.99620	3.28	0.59	
28	14.0	35.0	0.99720	3.47	0.55	
29	8.0	16.0	0.99640	3.38	0.59	
30	17.0	82.0	0.99580	3.35	0.54	
31	22.0	37.0	0.99660	3.46	0.57	
32	15.0	113.0	0.99660	3.17	0.66	
...	...	...	...	...	...	
1566	24.0	52.0	0.99494	3.34	0.71	

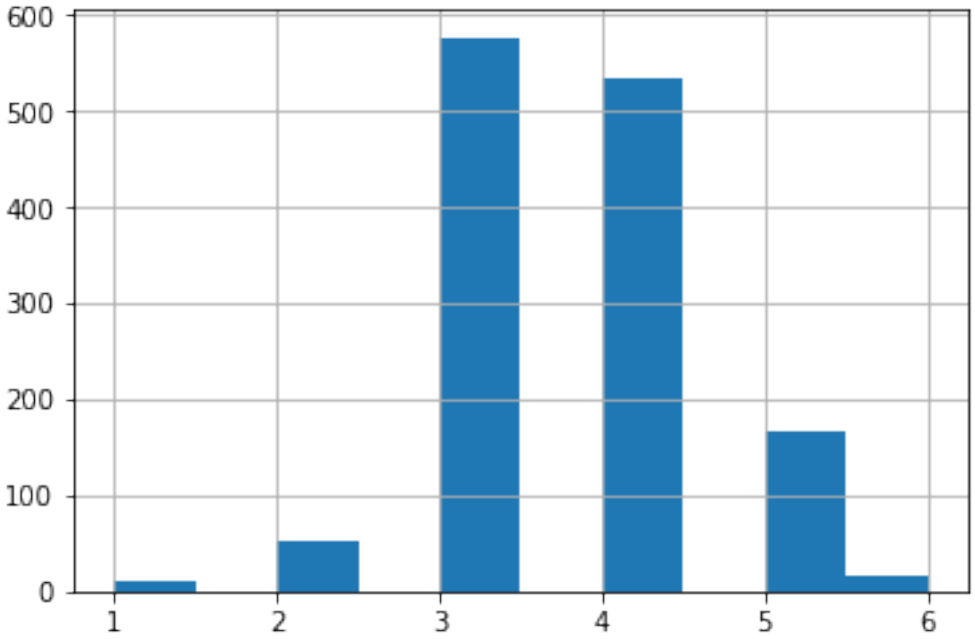
1568	25.0	42.0	0.99629	3.34	0.59
1569	15.0	34.0	0.99396	3.48	0.57
1570	19.0	35.0	0.99340	3.37	0.93
1571	15.0	25.0	0.99514	3.44	0.65
1572	35.0	104.0	0.99632	3.33	0.51
1573	15.0	50.0	0.99467	3.58	0.67
1574	23.0	92.0	0.99677	3.39	0.48
1575	12.0	20.0	0.99474	3.26	0.64
1576	16.0	29.0	0.99588	3.30	0.78
1577	13.0	27.0	0.99622	3.54	0.60
1578	13.0	20.0	0.99540	3.42	0.67
1579	24.0	32.0	0.99402	3.54	0.60
1580	9.0	26.0	0.99470	3.36	0.60
1582	13.0	27.0	0.99362	3.57	0.50
1583	32.0	98.0	0.99578	3.33	0.62
1584	24.0	34.0	0.99484	3.29	0.80
1585	22.0	48.0	0.99494	3.30	0.84
1586	34.0	60.0	0.99492	3.34	0.85
1587	18.0	28.0	0.99483	3.55	0.66
1588	34.0	102.0	0.99414	3.27	0.78
1589	29.0	79.0	0.99770	3.29	0.54
1590	26.0	35.0	0.99314	3.32	0.82
1591	16.0	26.0	0.99402	3.67	0.56
1592	29.0	40.0	0.99574	3.42	0.75
1593	28.0	38.0	0.99651	3.42	0.82
1594	32.0	44.0	0.99490	3.45	0.58
1595	39.0	51.0	0.99512	3.52	0.76
1597	32.0	44.0	0.99547	3.57	0.71
1598	18.0	42.0	0.99549	3.39	0.66

	alcohol	quality
0	9.4	3
1	9.8	3
2	9.8	3
3	9.8	4
5	9.4	3
6	9.4	3
7	10.0	5
8	9.5	5
9	10.5	3
10	9.2	3
12	9.9	3
13	9.1	3
14	9.2	3
15	9.2	3
16	10.5	5
17	9.3	3
18	9.0	2

```
19        9.2        4
20        9.4        4
21        9.7        3
22        9.5        3
23        9.4        3
24        9.7        4
25        9.3        3
26        9.5        3
28        9.4        3
29        9.8        4
30       10.1        3
31       10.6        4
32        9.8        3
...       ...      ...
1566     11.2        4
1568      9.2        3
1569     11.5        4
1570     12.4        4
1571     11.1        4
1572      9.5        3
1573     12.5        4
1574     10.5        4
1575     11.8        4
1576     10.8        4
1577     11.9        4
1578     11.3        4
1579     11.3        3
1580     11.9        4
1582     11.9        3
1583      9.8        3
1584     11.6        5
1585     11.5        4
1586     11.4        4
1587     10.9        4
1588     12.8        4
1589      9.2        3
1590     11.6        4
1591     11.6        4
1592     11.0        4
1593      9.5        4
1594     10.5        3
1595     11.2        4
1597     10.2        3
1598     11.0        4

[1359 rows x 12 columns]
```

In [21]:
```
df.quality.mode()
```

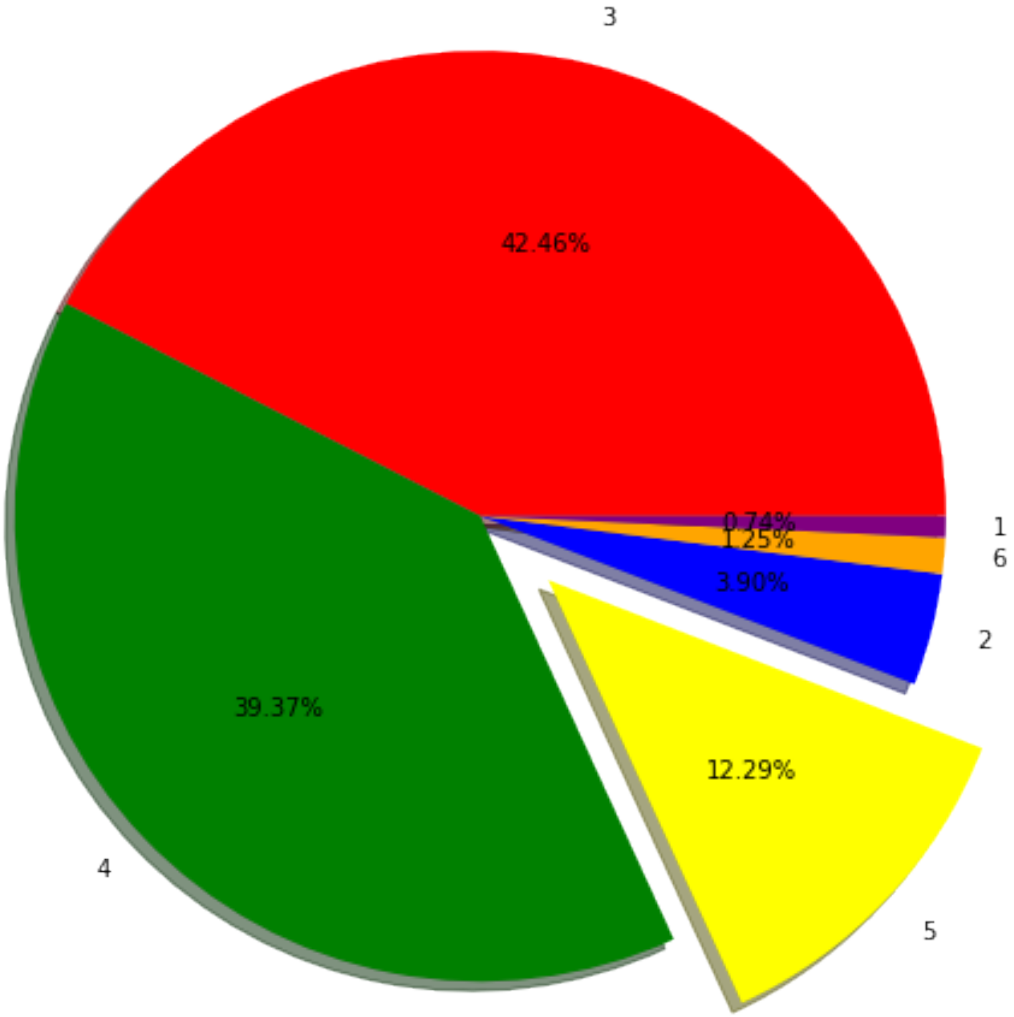
```
Out[21]: 0    3
         dtype: int64

In [22]: df.quality.hist()

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x106cb6dd8>
```




```
In [23]: df.loc[df['quality'] == 1] #to locate rows with particular values

Out[23]:       fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
         459             11.6             0.580         0.66            2.20      0.074
         517             10.4             0.610         0.49            2.10      0.200
         690              7.4             1.185         0.00            4.25      0.097
         832             10.4             0.440         0.42            1.50      0.145
         899              8.3             1.020         0.02            3.40      0.084
         1299             7.6             1.580         0.00            2.10      0.137
         1374             6.8             0.815         0.00            1.20      0.267
         1469             7.3             0.980         0.05            2.10      0.061
         1478             7.1             0.875         0.05            5.70      0.082
         1505             6.7             0.760         0.02            1.80      0.078

               free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
         459                  10.0                  47.0  1.00080  3.25       0.57
         517                   5.0                  16.0  0.99940  3.16       0.63
         690                   5.0                  14.0  0.99660  3.63       0.54
         832                  34.0                  48.0  0.99832  3.38       0.86
```

```
899                    6.0                  11.0  0.99892  3.48       0.49
1299                   5.0                   9.0  0.99476  3.50       0.40
1374                  16.0                  29.0  0.99471  3.32       0.51
1469                  20.0                  49.0  0.99705  3.31       0.55
1478                   3.0                  14.0  0.99808  3.40       0.52
1505                   6.0                  12.0  0.99600  3.55       0.63

      alcohol  quality
459      9.00        1
517      8.40        1
690     10.70        1
832      9.90        1
899     11.00        1
1299    10.90        1
1374     9.80        1
1469     9.70        1
1478    10.20        1
1505     9.95        1
```

In [233]:
```
labels = '3','4','5','2','6','1'
colors = ['red','green','yellow','blue','orange', 'purple']
plt.pie(df["quality"].value_counts(), labels=labels, colors=colors, autopct='%0.2f%%
plt.axis('equal')
fig = plt.gcf()
fig.set_size_inches(8,8)
plt.show()
```




```
In [228]: df.groupby('quality').count()
```

Out[228]:

quality	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
1	10	10	10	10	10	10	10	10	10
2	53	53	53	53					
3	577	577	577	577					
4	535	535	535	535					
5	167	167	167	167					
6	17	17	17	17					

```
2               53                  53                  53       53   53
3              577                 577                 577      577  577
4              535                 535                 535      535  535
5              167                 167                 167      167  167
6               17                  17                  17       17   17

           sulphates   ...     ca   rs   cl  free_sd  total_sd    d    p  sul  \
quality                ...
1                 10   ...     10   10   10       10        10   10   10   10
2                 53   ...     53   53   53       53        53   53   53   53
3                577   ...    577  577  577      577       577  577  577  577
4                535   ...    535  535  535      535       535  535  535  535
5                167   ...    167  167  167      167       167  167  167  167
6                 17   ...     17   17   17       17        17   17   17   17

            al  status
quality
1           10      10
2           53      53
3          577     577
4          535     535
5          167     167
6           17      17

[6 rows x 23 columns]
```

```
In [224]: df.sort_values(['volatile acidity'],ascending=True, inplace=False)
```

```
Out[224]:       fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
948               8.9             0.120         0.45            1.80      0.075
1286              7.1             0.160         0.44            2.50      0.068
1566              6.7             0.160         0.64            2.10      0.059
269              11.5             0.180         0.51            4.00      0.104
1017              8.0             0.180         0.37            0.90      0.049
1156              8.5             0.180         0.51            1.75      0.071
1230              7.7             0.180         0.34            2.70      0.066
1429              7.9             0.180         0.40            2.20      0.049
1509              7.9             0.180         0.40            1.80      0.062
301              11.1             0.180         0.48            1.50      0.068
1087              7.9             0.190         0.42            1.60      0.057
1131              5.9             0.190         0.21            1.70      0.045
291              11.0             0.200         0.48            2.00      0.343
1459              7.9             0.200         0.35            1.70      0.054
1145              8.2             0.200         0.43            2.50      0.076
873               9.1             0.210         0.37            1.60      0.067
243              15.0             0.210         0.44            2.20      0.075
354               6.1             0.210         0.40            1.40      0.066
955               8.5             0.210         0.52            1.90      0.090
```

518	10.9	0.210	0.49	2.80	0.088
1143	7.0	0.220	0.30	1.80	0.065
1490	7.1	0.220	0.49	1.80	0.039
925	8.6	0.220	0.36	1.90	0.064
20	8.9	0.220	0.48	1.80	0.077
530	9.1	0.220	0.24	2.10	0.078
1233	10.2	0.230	0.37	2.20	0.057
454	7.0	0.230	0.40	1.60	0.063
1426	7.7	0.230	0.37	1.80	0.046
1106	8.2	0.230	0.42	1.90	0.069
1060	11.6	0.230	0.57	1.80	0.074
...	...	...	...	...	...
422	7.7	0.960	0.20	2.00	0.047
1040	7.4	0.965	0.00	2.20	0.088
735	7.7	0.965	0.10	2.10	0.112
261	7.0	0.975	0.04	2.00	0.087
756	6.3	0.980	0.01	2.00	0.057
684	9.8	0.980	0.32	2.30	0.078
1469	7.3	0.980	0.05	2.10	0.061
775	7.2	1.000	0.00	3.00	0.102
234	8.2	1.000	0.09	2.30	0.065
1012	7.7	1.005	0.15	2.10	0.102
861	5.8	1.010	0.66	2.00	0.039
1261	6.3	1.020	0.00	2.00	0.083
700	10.6	1.020	0.43	2.90	0.076
94	5.0	1.020	0.04	1.40	0.045
899	8.3	1.020	0.02	3.40	0.084
710	10.6	1.025	0.43	2.80	0.080
705	8.4	1.035	0.15	6.00	0.073
1467	6.7	1.040	0.08	2.30	0.067
134	7.9	1.040	0.05	2.20	0.084
553	5.0	1.040	0.24	1.60	0.050
120	7.3	1.070	0.09	1.70	0.178
199	6.9	1.090	0.06	2.10	0.061
724	7.5	1.115	0.10	3.10	0.086
38	5.7	1.130	0.09	1.50	0.172
1312	8.0	1.180	0.21	1.90	0.083
690	7.4	1.185	0.00	4.25	0.097
672	9.8	1.240	0.34	2.00	0.079
126	8.2	1.330	0.00	1.70	0.081
127	8.1	1.330	0.00	1.80	0.082
1299	7.6	1.580	0.00	2.10	0.137

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
948	10.0	21.0	0.99552	3.41	0.76	
1286	17.0	31.0	0.99328	3.35	0.54	
1566	24.0	52.0	0.99494	3.34	0.71	
269	4.0	23.0	0.99960	3.28	0.97	

1017	36.0	109.0	0.99007	2.89	0.44
1156	45.0	88.0	0.99524	3.33	0.76
1230	15.0	58.0	0.99470	3.37	0.78
1429	38.0	67.0	0.99600	3.33	0.93
1509	7.0	20.0	0.99410	3.28	0.70
301	7.0	15.0	0.99730	3.22	0.64
1087	18.0	30.0	0.99400	3.29	0.69
1131	57.0	135.0	0.99341	3.32	0.44
291	6.0	18.0	0.99790	3.30	0.71
1459	7.0	15.0	0.99458	3.32	0.80
1145	31.0	51.0	0.99672	3.53	0.81
873	6.0	10.0	0.99552	3.23	0.58
243	10.0	24.0	1.00005	3.07	0.84
354	40.5	165.0	0.99120	3.25	0.59
955	9.0	23.0	0.99648	3.36	0.67
518	11.0	32.0	0.99720	3.22	0.68
1143	16.0	20.0	0.99672	3.61	0.82
1490	8.0	18.0	0.99344	3.39	0.56
925	53.0	77.0	0.99604	3.47	0.87
20	29.0	60.0	0.99680	3.39	0.53
530	1.0	28.0	0.99900	3.41	0.87
1233	14.0	36.0	0.99614	3.23	0.49
454	21.0	67.0	0.99520	3.50	0.63
1426	23.0	60.0	0.99710	3.41	0.71
1106	9.0	17.0	0.99376	3.21	0.54
1060	3.0	8.0	0.99810	3.14	0.70
...	...	...	...	...	...
422	15.0	60.0	0.99550	3.36	0.44
1040	16.0	32.0	0.99756	3.58	0.67
735	11.0	22.0	0.99630	3.26	0.50
261	12.0	67.0	0.99565	3.35	0.60
756	15.0	33.0	0.99488	3.60	0.46
684	35.0	152.0	0.99800	3.25	0.48
1469	20.0	49.0	0.99705	3.31	0.55
775	7.0	16.0	0.99586	3.43	0.46
234	7.0	37.0	0.99685	3.32	0.55
1012	11.0	32.0	0.99604	3.23	0.48
861	15.0	88.0	0.99357	3.66	0.60
1261	17.0	24.0	0.99437	3.59	0.55
700	26.0	88.0	0.99840	3.08	0.57
94	41.0	85.0	0.99380	3.75	0.48
899	6.0	11.0	0.99892	3.48	0.49
710	21.0	84.0	0.99850	3.06	0.57
705	11.0	54.0	0.99900	3.37	0.49
1467	19.0	32.0	0.99648	3.52	0.57
134	13.0	29.0	0.99590	3.22	0.55
553	32.0	96.0	0.99340	3.74	0.62
120	10.0	89.0	0.99620	3.30	0.57

```
199                   12.0                   31.0  0.99480  3.51       0.43
724                    5.0                   12.0  0.99580  3.54       0.60
38                     7.0                   19.0  0.99400  3.50       0.48
1312                  14.0                   41.0  0.99532  3.34       0.47
690                    5.0                   14.0  0.99660  3.63       0.54
672                   32.0                  151.0  0.99800  3.15       0.53
126                    3.0                   12.0  0.99640  3.53       0.49
127                    3.0                   12.0  0.99640  3.54       0.48
1299                   5.0                    9.0  0.99476  3.50       0.40

      ...    ca  rs  cl  free_sd  total_sd  d  p  sul  al  status
948   ...     2   1   1        1         1  4  2    2   2       1
1286  ...     2   1   1        1         1  4  2    2   3       1
1566  ...     3   1   1        1         1  4  2    2   2       1
269   ...     3   1   1        1         1  4  2    2   2       1
1017  ...     2   1   1        2         2  4  2    1   3       1
1156  ...     3   1   1        2         2  4  2    2   2       1
1230  ...     2   1   1        1         1  4  2    2   2       1
1429  ...     2   1   1        2         1  4  2    2   2       0
1509  ...     2   1   1        1         1  4  2    2   2       0
301   ...     2   1   1        1         1  4  2    2   2       1
1087  ...     2   1   1        1         1  4  2    2   2       1
1131  ...     1   1   1        3         2  4  2    1   1       0
291   ...     2   1   2        1         1  4  2    2   2       0
1459  ...     2   1   1        1         1  4  2    2   2       1
1145  ...     2   1   1        2         1  4  3    2   2       1
873   ...     2   1   1        1         1  4  2    2   2       1
243   ...     2   1   1        1         1  4  2    2   1       1
354   ...     2   1   1        2         3  4  2    2   2       1
955   ...     3   1   1        1         1  4  2    2   2       0
518   ...     2   1   1        1         1  4  2    2   2       1
1143  ...     2   1   1        1         1  4  3    2   1       1
1490  ...     2   1   1        1         1  4  2    2   3       1
925   ...     2   1   1        3         1  4  2    2   2       1
20    ...     2   1   1        2         1  4  2    2   1       1
530   ...     1   1   1        1         1  4  2    2   2       1
1233  ...     2   1   1        1         1  4  2    1   1       0
454   ...     2   1   1        1         1  4  2    2   2       0
1426  ...     2   1   1        1         1  4  2    2   3       1
1106  ...     2   1   1        1         1  4  2    2   3       1
1060  ...     3   1   1        1         1  4  2    2   1       1
...   ...    ..  ..  ..      ...       ... .. ..   ..  ..     ...
422   ...     1   1   1        1         1  4  2    1   2       0
1040  ...     1   1   1        1         1  4  3    2   2       0
735   ...     1   1   1        1         1  4  2    1   1       0
261   ...     1   1   1        1         1  4  2    2   1       0
756   ...     1   1   1        1         1  4  3    1   2       1
684   ...     2   1   1        2         2  4  2    1   1       0
```

```
1469  ...     1   1  1        1         1  4  2   2  1       0
775   ...     1   1  1        1         1  4  2   1  1       0
234   ...     1   1  1        1         1  4  2   2  1       1
1012  ...     1   1  1        1         1  4  2   1  1       0
861   ...     3   1  1        1         2  4  3   2  2       1
1261  ...     1   1  1        1         1  4  3   2  2       0
700   ...     2   1  1        2         2  4  2   2  2       1
94    ...     1   1  1        2         2  4  3   1  2       0
899   ...     1   1  1        1         1  4  2   1  2       0
710   ...     2   1  1        1         2  4  2   2  2       0
705   ...     1   2  1        1         1  4  2   1  1       0
1467  ...     1   1  1        1         1  4  3   2  2       0
134   ...     1   1  1        1         1  4  2   2  1       1
553   ...     1   1  1        2         2  4  3   2  2       0
120   ...     1   1  1        1         2  4  2   2  1       0
199   ...     1   1  1        1         1  4  3   1  2       0
724   ...     1   1  1        1         1  4  3   2  2       0
38    ...     1   1  1        1         1  4  2   1  1       0
1312  ...     1   1  1        1         1  4  2   1  2       0
690   ...     1   1  1        1         1  4  3   2  2       0
672   ...     2   1  1        2         2  4  2   2  1       0
126   ...     1   1  1        1         1  4  3   1  2       0
127   ...     1   1  1        1         1  4  3   1  2       0
1299  ...     1   1  1        1         1  4  2   1  2       0

[1359 rows x 24 columns]
```

```
In [27]: df.info()   # to get the info of all te columns of the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1359 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1359 non-null float64
volatile acidity        1359 non-null float64
citric acid             1359 non-null float64
residual sugar          1359 non-null float64
chlorides               1359 non-null float64
free sulfur dioxide     1359 non-null float64
total sulfur dioxide    1359 non-null float64
density                 1359 non-null float64
pH                      1359 non-null float64
sulphates               1359 non-null float64
alcohol                 1359 non-null float64
quality                 1359 non-null int64
dtypes: float64(11), int64(1)
memory usage: 138.0 KB
```

0.2.1 Grouping column values using bins

```
In [28]: bin1=[1,5,10,15,20]
         label1=[1,2,3,4]
         df['fa']=pd.cut(df['fixed acidity'],bins=bin1,labels=label1)
         print(df[['fixed acidity','fa']])
```

```
      fixed acidity fa
0               7.4  2
1               7.8  2
2               7.8  2
3              11.2  3
5               7.4  2
6               7.9  2
7               7.3  2
8               7.8  2
9               7.5  2
10              6.7  2
12              5.6  2
13              7.8  2
14              8.9  2
15              8.9  2
16              8.5  2
17              8.1  2
18              7.4  2
19              7.9  2
20              8.9  2
21              7.6  2
22              7.9  2
23              8.5  2
24              6.9  2
25              6.3  2
26              7.6  2
28              7.1  2
29              7.8  2
30              6.7  2
31              6.9  2
32              8.3  2
...             ... ..
1566            6.7  2
1568            7.0  2
1569            6.2  2
1570            6.4  2
1571            6.4  2
1572            7.3  2
1573            6.0  2
1574            5.6  2
1575            7.5  2
1576            8.0  2
```

```
1577            6.2  2
1578            6.8  2
1579            6.2  2
1580            7.4  2
1582            6.1  2
1583            6.2  2
1584            6.7  2
1585            7.2  2
1586            7.5  2
1587            5.8  2
1588            7.2  2
1589            6.6  2
1590            6.3  2
1591            5.4  2
1592            6.3  2
1593            6.8  2
1594            6.2  2
1595            5.9  2
1597            5.9  2
1598            6.0  2

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


In [29]: print(df[['fixed acidity','fa']].head(5))

   fixed acidity fa
0            7.4  2
1            7.8  2
2            7.8  2
3           11.2  3
5            7.4  2


In [30]: bin2=[0,0.5,1,1.5,2]
         label2=[1,2,3,4]
         df['va']=pd.cut(df['volatile acidity'],bins=bin2,labels=label2)
         print(df[['volatile acidity','va']])

      volatile acidity va
0                0.700  2
```

1	0.880	2
2	0.760	2
3	0.280	1
5	0.660	2
6	0.600	2
7	0.650	2
8	0.580	2
9	0.500	1
10	0.580	2
12	0.615	2
13	0.610	2
14	0.620	2
15	0.620	2
16	0.280	1
17	0.560	2
18	0.590	2
19	0.320	1
20	0.220	1
21	0.390	1
22	0.430	1
23	0.490	1
24	0.400	1
25	0.390	1
26	0.410	1
28	0.710	2
29	0.645	2
30	0.675	2
31	0.685	2
32	0.655	2
...	...	..
1566	0.160	1
1568	0.560	2
1569	0.510	2
1570	0.360	1
1571	0.380	1
1572	0.690	2
1573	0.580	2
1574	0.310	1
1575	0.520	2
1576	0.300	1
1577	0.700	2
1578	0.670	2
1579	0.560	2
1580	0.350	1
1582	0.715	2
1583	0.460	1
1584	0.320	1
1585	0.390	1

```
1586             0.310  1
1587             0.610  2
1588             0.660  2
1589             0.725  2
1590             0.550  2
1591             0.740  2
1592             0.510  2
1593             0.620  2
1594             0.600  2
1595             0.550  2
1597             0.645  2
1598             0.310  1

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [31]: bin3=[-0.25,0.25,0.5,0.75,1]
         label3=[1,2,3,4]
         df['ca']=pd.cut(df['citric acid'],bins=bin3,labels=label3)
         print(df[['citric acid','ca']])
```

```
      citric acid ca
0            0.00  1
1            0.00  1
2            0.04  1
3            0.56  3
5            0.00  1
6            0.06  1
7            0.00  1
8            0.02  1
9            0.36  2
10           0.08  1
12           0.00  1
13           0.29  2
14           0.18  1
15           0.19  1
16           0.56  3
17           0.28  2
18           0.08  1
19           0.51  3
```

```
20           0.48  2
21           0.31  2
22           0.21  1
23           0.11  1
24           0.14  1
25           0.16  1
26           0.24  1
28           0.00  1
29           0.00  1
30           0.07  1
31           0.00  1
32           0.12  1
...           ... ..
1566         0.64  3
1568         0.13  1
1569         0.14  1
1570         0.53  3
1571         0.14  1
1572         0.32  2
1573         0.20  1
1574         0.78  4
1575         0.40  2
1576         0.63  3
1577         0.15  1
1578         0.15  1
1579         0.09  1
1580         0.33  2
1582         0.10  1
1583         0.29  2
1584         0.44  2
1585         0.44  2
1586         0.41  2
1587         0.11  1
1588         0.33  2
1589         0.20  1
1590         0.15  1
1591         0.09  1
1592         0.13  1
1593         0.08  1
1594         0.08  1
1595         0.10  1
1597         0.12  1
1598         0.47  2

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [32]: df['ca'].isnull().any()
```

```
Out[32]: False
```

```
In [33]: bin4=[-5,5,10,15,20]
         label4=[1,2,3,4]
         df['rs']=pd.cut(df['residual sugar'],bins=bin4,labels=label4)
         print(df[['residual sugar','rs']])
```

```
      residual sugar rs
0                1.9  1
1                2.6  1
2                2.3  1
3                1.9  1
5                1.8  1
6                1.6  1
7                1.2  1
8                2.0  1
9                6.1  2
10               1.8  1
12               1.6  1
13               1.6  1
14               3.8  1
15               3.9  1
16               1.8  1
17               1.7  1
18               4.4  1
19               1.8  1
20               1.8  1
21               2.3  1
22               1.6  1
23               2.3  1
24               2.4  1
25               1.4  1
26               1.8  1
28               1.9  1
29               2.0  1
30               2.4  1
31               2.5  1
32               2.3  1
...              ... ..
1566             2.1  1
```

```
1568            1.6  1
1569            1.9  1
1570            2.2  1
1571            2.2  1
1572            2.2  1
1573            2.4  1
1574           13.9  3
1575            2.2  1
1576            1.6  1
1577            5.1  2
1578            1.8  1
1579            1.7  1
1580            2.4  1
1582            2.6  1
1583            2.1  1
1584            2.4  1
1585            2.6  1
1586            2.4  1
1587            1.8  1
1588            2.5  1
1589            7.8  2
1590            1.8  1
1591            1.7  1
1592            2.3  1
1593            1.9  1
1594            2.0  1
1595            2.2  1
1597            2.0  1
1598            3.6  1

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [34]: bin5=[0,0.2,0.4,0.6,0.8]
         label5=[1,2,3,4]
         df['cl']=pd.cut(df['chlorides'],bins=bin5,labels=label5)
         print(df[['chlorides','cl']])
```

```
      chlorides cl
0         0.076  1
```

1	0.098	1
2	0.092	1
3	0.075	1
5	0.075	1
6	0.069	1
7	0.065	1
8	0.073	1
9	0.071	1
10	0.097	1
12	0.089	1
13	0.114	1
14	0.176	1
15	0.170	1
16	0.092	1
17	0.368	2
18	0.086	1
19	0.341	2
20	0.077	1
21	0.082	1
22	0.106	1
23	0.084	1
24	0.085	1
25	0.080	1
26	0.080	1
28	0.080	1
29	0.082	1
30	0.089	1
31	0.105	1
32	0.083	1
...	...	..
1566	0.059	1
1568	0.077	1
1569	0.056	1
1570	0.230	2
1571	0.038	1
1572	0.069	1
1573	0.075	1
1574	0.074	1
1575	0.060	1
1576	0.081	1
1577	0.076	1
1578	0.118	1
1579	0.053	1
1580	0.068	1
1582	0.053	1
1583	0.074	1
1584	0.061	1
1585	0.066	1

```
1586      0.065  1
1587      0.066  1
1588      0.068  1
1589      0.073  1
1590      0.077  1
1591      0.089  1
1592      0.076  1
1593      0.068  1
1594      0.090  1
1595      0.062  1
1597      0.075  1
1598      0.067  1

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [35]: bin6=[0,25,50,75,100]
         label6=[1,2,3,4]
         df['free_sd']=pd.cut(df['free sulfur dioxide'],bins=bin6,labels=label6)
         print(df[['free sulfur dioxide','free_sd']])
```

```
      free sulfur dioxide free_sd
0                    11.0       1
1                    25.0       1
2                    15.0       1
3                    17.0       1
5                    13.0       1
6                    15.0       1
7                    15.0       1
8                     9.0       1
9                    17.0       1
10                   15.0       1
12                   16.0       1
13                    9.0       1
14                   52.0       3
15                   51.0       3
16                   35.0       2
17                   16.0       1
18                    6.0       1
19                   17.0       1
```

```
20                  29.0       2
21                  23.0       1
22                  10.0       1
23                   9.0       1
24                  21.0       1
25                  11.0       1
26                   4.0       1
28                  14.0       1
29                   8.0       1
30                  17.0       1
31                  22.0       1
32                  15.0       1
...                  ...     ...
1566                24.0       1
1568                25.0       1
1569                15.0       1
1570                19.0       1
1571                15.0       1
1572                35.0       2
1573                15.0       1
1574                23.0       1
1575                12.0       1
1576                16.0       1
1577                13.0       1
1578                13.0       1
1579                24.0       1
1580                 9.0       1
1582                13.0       1
1583                32.0       2
1584                24.0       1
1585                22.0       1
1586                34.0       2
1587                18.0       1
1588                34.0       2
1589                29.0       2
1590                26.0       2
1591                16.0       1
1592                29.0       2
1593                28.0       2
1594                32.0       2
1595                39.0       2
1597                32.0       2
1598                18.0       1

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [36]: bin7=[5,80,155,230,305]
         label7=[1,2,3,4]
         df['total_sd']=pd.cut(df['total sulfur dioxide'],bins=bin7,labels=label7)
         print(df[['total sulfur dioxide','total_sd']])
```

	total sulfur dioxide	total_sd
0	34.0	1
1	67.0	1
2	54.0	1
3	60.0	1
5	40.0	1
6	59.0	1
7	21.0	1
8	18.0	1
9	102.0	2
10	65.0	1
12	59.0	1
13	29.0	1
14	145.0	2
15	148.0	2
16	103.0	2
17	56.0	1
18	29.0	1
19	56.0	1
20	60.0	1
21	71.0	1
22	37.0	1
23	67.0	1
24	40.0	1
25	23.0	1
26	11.0	1
28	35.0	1
29	16.0	1
30	82.0	2
31	37.0	1
32	113.0	2
...	...	...
1566	52.0	1
1568	42.0	1
1569	34.0	1
1570	35.0	1

```
1571                  25.0        1
1572                 104.0        2
1573                  50.0        1
1574                  92.0        2
1575                  20.0        1
1576                  29.0        1
1577                  27.0        1
1578                  20.0        1
1579                  32.0        1
1580                  26.0        1
1582                  27.0        1
1583                  98.0        2
1584                  34.0        1
1585                  48.0        1
1586                  60.0        1
1587                  28.0        1
1588                 102.0        2
1589                  79.0        1
1590                  35.0        1
1591                  26.0        1
1592                  40.0        1
1593                  38.0        1
1594                  44.0        1
1595                  51.0        1
1597                  44.0        1
1598                  42.0        1

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [37]: bin8=[0,0.3,0.6,0.9,1.2]
         label8=[1,2,3,4]
         df['d']=pd.cut(df['density'],bins=bin8,labels=label8)
         print(df[['density','d']])
```

```
      density  d
0     0.99780  4
1     0.99680  4
2     0.99700  4
3     0.99800  4
```

```
5     0.99780  4
6     0.99640  4
7     0.99460  4
8     0.99680  4
9     0.99780  4
10    0.99590  4
12    0.99430  4
13    0.99740  4
14    0.99860  4
15    0.99860  4
16    0.99690  4
17    0.99680  4
18    0.99740  4
19    0.99690  4
20    0.99680  4
21    0.99820  4
22    0.99660  4
23    0.99680  4
24    0.99680  4
25    0.99550  4
26    0.99620  4
28    0.99720  4
29    0.99640  4
30    0.99580  4
31    0.99660  4
32    0.99660  4
...       ... ..
1566  0.99494  4
1568  0.99629  4
1569  0.99396  4
1570  0.99340  4
1571  0.99514  4
1572  0.99632  4
1573  0.99467  4
1574  0.99677  4
1575  0.99474  4
1576  0.99588  4
1577  0.99622  4
1578  0.99540  4
1579  0.99402  4
1580  0.99470  4
1582  0.99362  4
1583  0.99578  4
1584  0.99484  4
1585  0.99494  4
1586  0.99492  4
1587  0.99483  4
1588  0.99414  4
```

```
1589  0.99770  4
1590  0.99314  4
1591  0.99402  4
1592  0.99574  4
1593  0.99651  4
1594  0.99490  4
1595  0.99512  4
1597  0.99547  4
1598  0.99549  4

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


In [38]: bin9=[2,2.75,3.5,4.25,5]
         label9=[1,2,3,4]
         df['p']=pd.cut(df['pH'],bins=bin9,labels=label9)
         print(df[['pH','p']])

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


       pH  p
0    3.51  3
1    3.20  2
2    3.26  2
3    3.16  2
5    3.51  3
6    3.30  2
7    3.39  2
8    3.36  2
9    3.35  2
10   3.28  2
12   3.58  3
13   3.26  2
14   3.16  2
15   3.17  2
```

```
16    3.30  2
17    3.11  2
18    3.38  2
19    3.04  2
20    3.39  2
21    3.52  3
22    3.17  2
23    3.17  2
24    3.43  2
25    3.34  2
26    3.28  2
28    3.47  2
29    3.38  2
30    3.35  2
31    3.46  2
32    3.17  2
...    ... ..
1566  3.34  2
1568  3.34  2
1569  3.48  2
1570  3.37  2
1571  3.44  2
1572  3.33  2
1573  3.58  3
1574  3.39  2
1575  3.26  2
1576  3.30  2
1577  3.54  3
1578  3.42  2
1579  3.54  3
1580  3.36  2
1582  3.57  3
1583  3.33  2
1584  3.29  2
1585  3.30  2
1586  3.34  2
1587  3.55  3
1588  3.27  2
1589  3.29  2
1590  3.32  2
1591  3.67  3
1592  3.42  2
1593  3.42  2
1594  3.45  2
1595  3.52  3
1597  3.57  3
1598  3.39  2
```

```
[1359 rows x 2 columns]
```

```
In [39]: bin10=[0,0.5,1,1.5,2]
         label10=[1,2,3,4]
         df['sul']=pd.cut(df['sulphates'],bins=bin10,labels=label10)
         print(df[['sulphates','sul']])
```

```
      sulphates sul
0          0.56   2
1          0.68   2
2          0.65   2
3          0.58   2
5          0.56   2
6          0.46   1
7          0.47   1
8          0.57   2
9          0.80   2
10         0.54   2
12         0.52   2
13         1.56   4
14         0.88   2
15         0.93   2
16         0.75   2
17         1.28   3
18         0.50   1
19         1.08   3
20         0.53   2
21         0.65   2
22         0.91   2
23         0.53   2
24         0.63   2
25         0.56   2
26         0.59   2
28         0.55   2
29         0.59   2
30         0.54   2
31         0.57   2
32         0.66   2
...         ...  ..
1566       0.71   2
1568       0.59   2
1569       0.57   2
1570       0.93   2
1571       0.65   2
1572       0.51   2
1573       0.67   2
1574       0.48   1
```

```
1575       0.64   2
1576       0.78   2
1577       0.60   2
1578       0.67   2
1579       0.60   2
1580       0.60   2
1582       0.50   1
1583       0.62   2
1584       0.80   2
1585       0.84   2
1586       0.85   2
1587       0.66   2
1588       0.78   2
1589       0.54   2
1590       0.82   2
1591       0.56   2
1592       0.75   2
1593       0.82   2
1594       0.58   2
1595       0.76   2
1597       0.71   2
1598       0.66   2

[1359 rows x 2 columns]


/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


In [40]: bin11=[8,10,12,14,16]
         label11=[1,2,3,4]
         df['al']=pd.cut(df['alcohol'],bins=bin11,labels=label11)
         print(df[['alcohol','al']])

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until


      alcohol al
0         9.4  1
```

1	9.8	1
2	9.8	1
3	9.8	1
5	9.4	1
6	9.4	1
7	10.0	1
8	9.5	1
9	10.5	2
10	9.2	1
12	9.9	1
13	9.1	1
14	9.2	1
15	9.2	1
16	10.5	2
17	9.3	1
18	9.0	1
19	9.2	1
20	9.4	1
21	9.7	1
22	9.5	1
23	9.4	1
24	9.7	1
25	9.3	1
26	9.5	1
28	9.4	1
29	9.8	1
30	10.1	2
31	10.6	2
32	9.8	1
...	...	..
1566	11.2	2
1568	9.2	1
1569	11.5	2
1570	12.4	3
1571	11.1	2
1572	9.5	1
1573	12.5	3
1574	10.5	2
1575	11.8	2
1576	10.8	2
1577	11.9	2
1578	11.3	2
1579	11.3	2
1580	11.9	2
1582	11.9	2
1583	9.8	1
1584	11.6	2
1585	11.5	2

```
1586     11.4  2
1587     10.9  2
1588     12.8  3
1589      9.2  1
1590     11.6  2
1591     11.6  2
1592     11.0  2
1593      9.5  1
1594     10.5  2
1595     11.2  2
1597     10.2  2
1598     11.0  2

[1359 rows x 2 columns]
```

```
In [41]: df.head()
```

```
Out[41]:    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
         0            7.4              0.70         0.00             1.9      0.076
         1            7.8              0.88         0.00             2.6      0.098
         2            7.8              0.76         0.04             2.3      0.092
         3           11.2              0.28         0.56             1.9      0.075
         5            7.4              0.66         0.00             1.8      0.075

            free sulfur dioxide  total sulfur dioxide  density    pH  sulphates ...  \
         0                 11.0                  34.0   0.9978  3.51       0.56 ...
         1                 25.0                  67.0   0.9968  3.20       0.68 ...
         2                 15.0                  54.0   0.9970  3.26       0.65 ...
         3                 17.0                  60.0   0.9980  3.16       0.58 ...
         5                 13.0                  40.0   0.9978  3.51       0.56 ...

            va  ca rs cl free_sd total_sd  d  p sul al
         0   2   1  1  1       1        1  4  3   2  1
         1   2   1  1  1       1        1  4  2   2  1
         2   2   1  1  1       1        1  4  2   2  1
         3   1   3  1  1       1        1  4  2   2  1
         5   2   1  1  1       1        1  4  3   2  1

         [5 rows x 23 columns]
```

0.3 Classifying the target variable by grouping values into two distinct values - 0 and 1

```
In [42]: df.loc[ df.quality > 3, 'status' ] = '1'    # 1 will represent good quality
         df.loc[ df.quality <= 3, 'status' ] = '0'   # 0 will represent bad quality
         print(df[['quality','status']])
```

```
/anaconda3/lib/python3.6/site-packages/pandas/core/indexing.py:357: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[key] = _infer_fill_value(value)
/anaconda3/lib/python3.6/site-packages/pandas/core/indexing.py:537: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

	quality	status
0	3	0
1	3	0
2	3	0
3	4	1
5	3	0
6	3	0
7	5	1
8	5	1
9	3	0
10	3	0
12	3	0
13	3	0
14	3	0
15	3	0
16	5	1
17	3	0
18	2	0
19	4	1
20	4	1
21	3	0
22	3	0
23	3	0
24	4	1
25	3	0
26	3	0
28	3	0
29	4	1
30	3	0
31	4	1
32	3	0
...	...	...
1566	4	1
1568	3	0

```
1569        4      1
1570        4      1
1571        4      1
1572        3      0
1573        4      1
1574        4      1
1575        4      1
1576        4      1
1577        4      1
1578        4      1
1579        3      0
1580        4      1
1582        3      0
1583        3      0
1584        5      1
1585        4      1
1586        4      1
1587        4      1
1588        4      1
1589        3      0
1590        4      1
1591        4      1
1592        4      1
1593        4      1
1594        3      0
1595        4      1
1597        3      0
1598        4      1

[1359 rows x 2 columns]
```

1 Data Visualizations

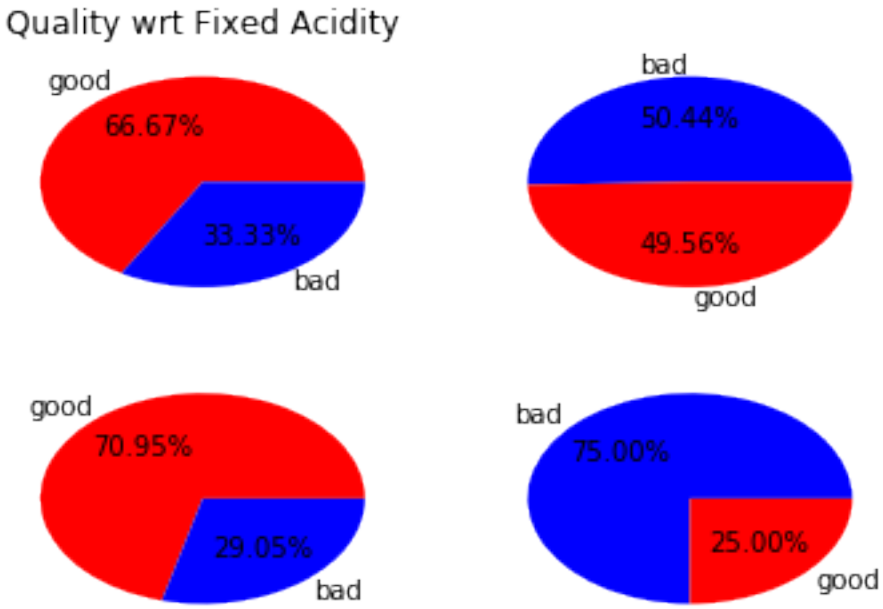
```
In [47]: df.status[df["fa"]==4].value_counts()
```

```
Out[47]: 0    3
         1    1
         Name: status, dtype: int64
```

1.0.1 Quality wrt Fixed Acidity

```
In [48]: plt.subplot(2,2,1)
         plt.title('Quality wrt Fixed Acidity')
         plt.pie(df.status[df["fa"]==1].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,2)
         plt.pie(df.status[df["fa"]==2].value_counts(),colors=("blue","red"), labels=("bad","g
```

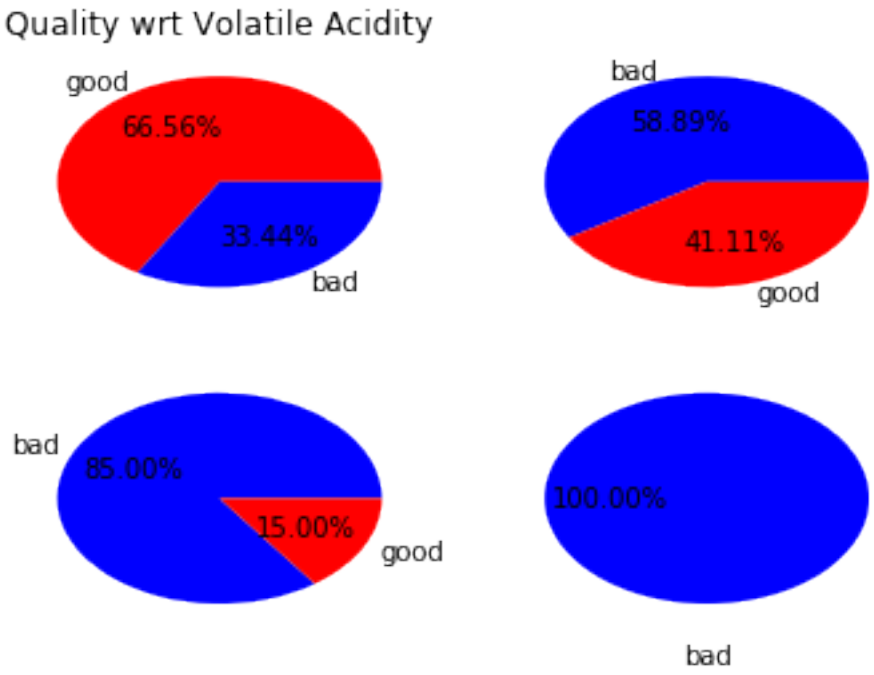
```
plt.subplot(2,2,3)
plt.pie(df.status[df["fa"]==3].value_counts(),colors=("red","blue"), labels=("good","
plt.subplot(2,2,4)
plt.pie(df.status[df["fa"]==4].value_counts(),colors=("blue","red"), labels=("bad","g
plt.show()
```



1.0.2 Quality wrt Volatile Acidity

```
In [49]: df.status[df["va"]==4].value_counts()

Out[49]: 0    1
         Name: status, dtype: int64

In [50]: plt.subplot(2,2,1)
         plt.title('Quality wrt Volatile Acidity')
         plt.pie(df.status[df["va"]==1].value_counts(),colors=("red","blue"), labels=("good","
         plt.subplot(2,2,2)
         plt.pie(df.status[df["va"]==2].value_counts(),colors=("blue","red"), labels=("bad","g
         plt.subplot(2,2,3)
         plt.pie(df.status[df["va"]==3].value_counts(),colors=("blue","red"), labels=("bad","g
```

```
plt.subplot(2,2,4)
plt.pie(df.status[df["va"]==4].value_counts(),colors=("blue"), autopct='%0.2f%%')
plt.xlabel('bad')


plt.show()
```

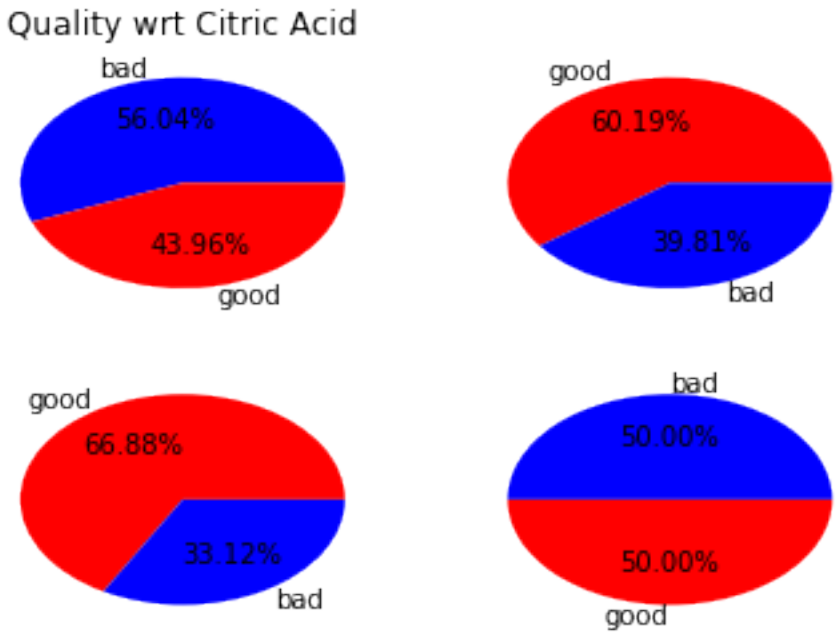


Quality wrt Citric Acid

```
In [51]: df.status[df["ca"]==4].value_counts()

Out[51]: 0    3
         1    3
         Name: status, dtype: int64

In [52]: plt.subplot(2,2,1)
         plt.title('Quality wrt Citric Acid')
         plt.pie(df.status[df["ca"]==1].value_counts(),colors=("blue","red"), labels=("bad","g

         plt.subplot(2,2,2)
         plt.pie(df.status[df["ca"]==2].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,3)
         plt.pie(df.status[df["ca"]==3].value_counts(),colors=("red","blue"), labels=("good","
```

```
plt.subplot(2,2,4)
plt.pie(df.status[df["ca"]==4].value_counts(),colors=("blue","red"), labels=("bad","g
```

```
plt.show()
```



1.0.3 Quality wrt Residual Sugar

In [53]:
```
df.status[df["rs"]==2].value_counts()
```

Out[53]:
```
1    29
0    29
Name: status, dtype: int64
```

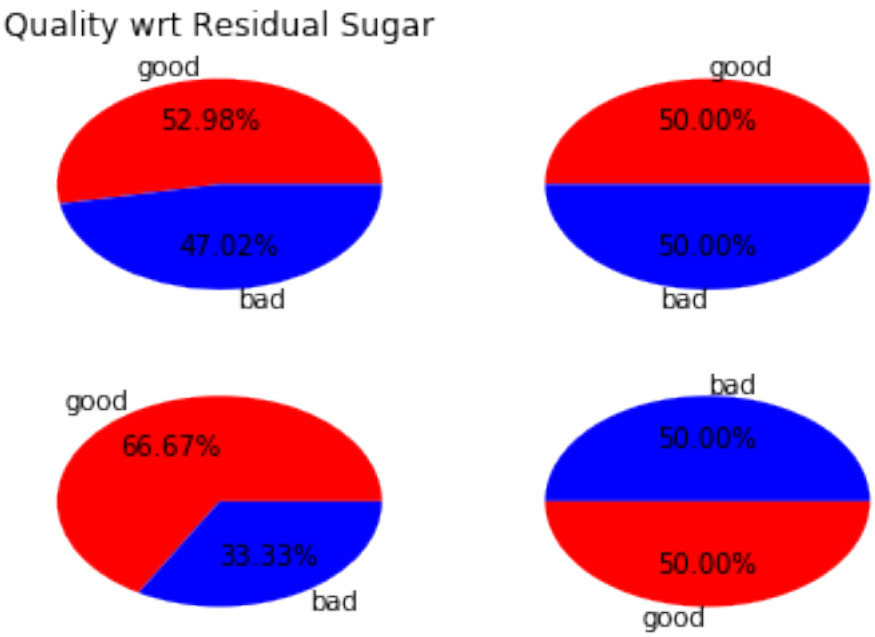
In [54]:
```
plt.subplot(2,2,1)
plt.title('Quality wrt Residual Sugar')
plt.pie(df.status[df["rs"]==1].value_counts(),colors=("red","blue"), labels=("good","
```

```
plt.subplot(2,2,2)
plt.pie(df.status[df["rs"]==2].value_counts(),colors=("red","blue"), labels=("good","
```

```
plt.subplot(2,2,3)
plt.pie(df.status[df["rs"]==3].value_counts(),colors=("red","blue"), labels=("good","
```

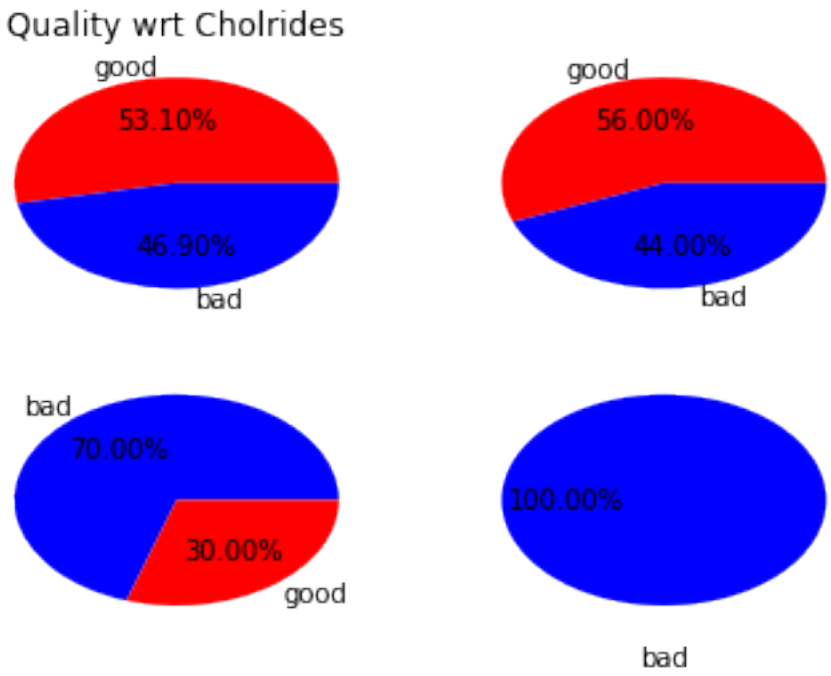
```
plt.subplot(2,2,4)
plt.pie(df.status[df["rs"]==4].value_counts(),colors=("blue","red"), labels=("bad","g
```

```
plt.show()
```



1.0.4 Quality wrt Cholrides

```
In [55]: df.status[df["cl"]==4].value_counts()

Out[55]: 0    2
         Name: status, dtype: int64

In [56]: plt.subplot(2,2,1)
         plt.title('Quality wrt Cholrides')
         plt.pie(df.status[df["cl"]==1].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,2)
         plt.pie(df.status[df["cl"]==2].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,3)
         plt.pie(df.status[df["cl"]==3].value_counts(),colors=("blue","red"), labels=("bad","g

         plt.subplot(2,2,4)
         plt.pie(df.status[df["cl"]==4].value_counts(),colors=("blue"),autopct='%0.2f%%')
         plt.xlabel('bad')

         plt.show()
```

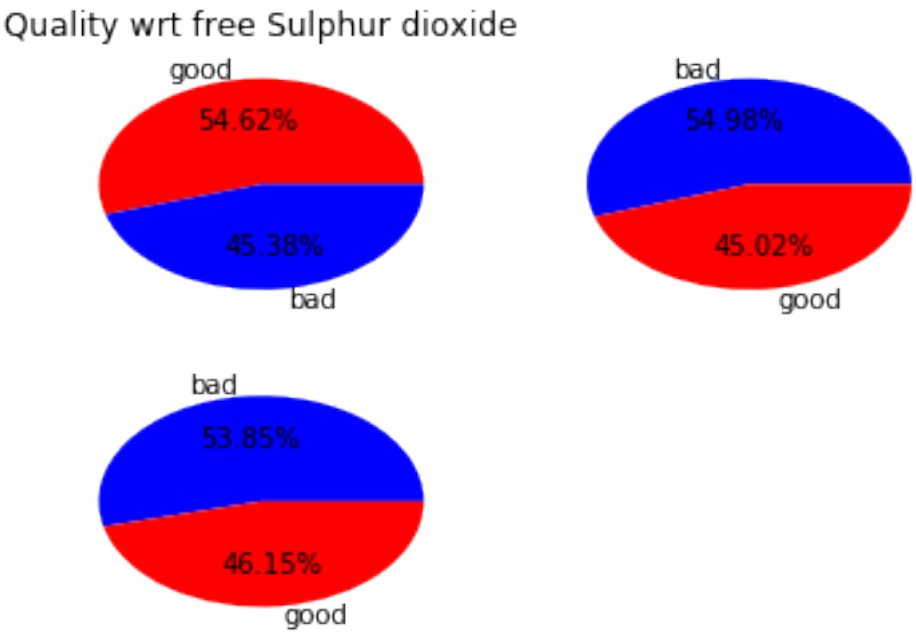


1.0.5 Quality wrt free Sulphur dioxide

```
In [63]: df.status[df["free_sd"]==4].value_counts()
```

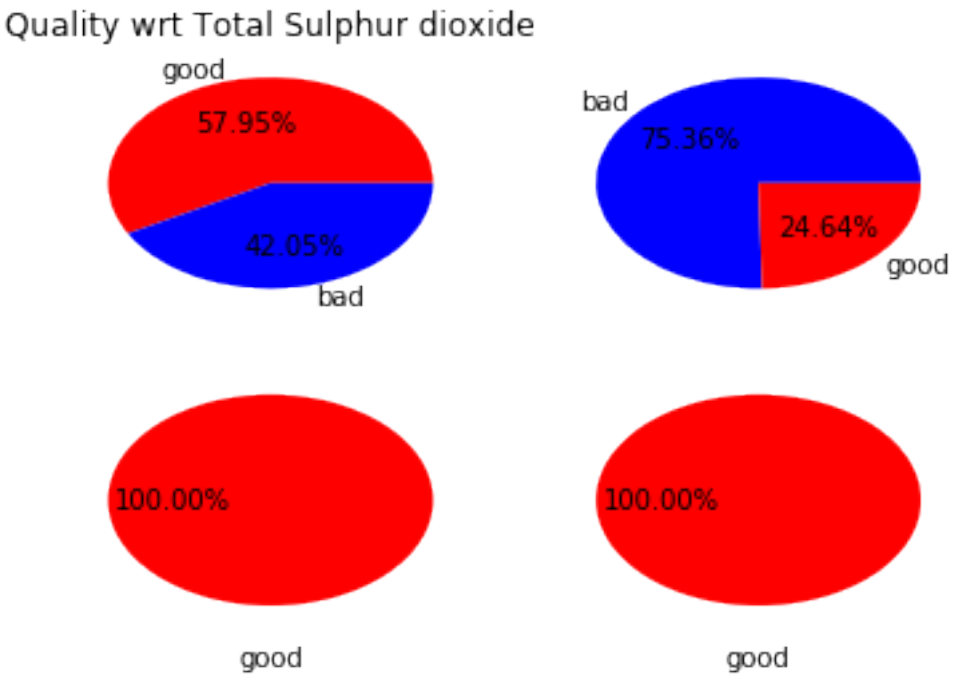
```
Out[63]: Series([], Name: status, dtype: int64)
```

```
In [58]: plt.subplot(2,2,1)
         plt.title('Quality wrt free Sulphur dioxide')
         plt.pie(df.status[df["free_sd"]==1].value_counts(),colors=("red","blue"), labels=("goo

         plt.subplot(2,2,2)
         plt.pie(df.status[df["free_sd"]==2].value_counts(),colors=("blue","red"), labels=("bad

         plt.subplot(2,2,3)
         plt.pie(df.status[df["free_sd"]==3].value_counts(),colors=("blue","red"), labels=("bad

         plt.show()
```



1.0.6 Quality wrt Total Sulphur dioxide

```
In [59]: df.status[df["total_sd"]==4].value_counts()

Out[59]: 1    2
         Name: status, dtype: int64

In [60]: plt.subplot(2,2,1)
         plt.title('Quality wrt Total Sulphur dioxide')
         plt.pie(df.status[df["total_sd"]==1].value_counts(),colors=("red","blue"), labels=("g

         plt.subplot(2,2,2)
         plt.pie(df.status[df["total_sd"]==2].value_counts(),colors=("blue","red"), labels=("ba

         plt.subplot(2,2,3)
         plt.pie(df.status[df["total_sd"]==3].value_counts(),colors=("red"), autopct='%0.2f%%'
         plt.xlabel('good')

         plt.subplot(2,2,4)
         plt.pie(df.status[df["total_sd"]==4].value_counts(),colors=("red"), autopct='%0.2f%%'
         plt.xlabel('good')

         plt.show()
```



1.0.7 Quality wrt Density

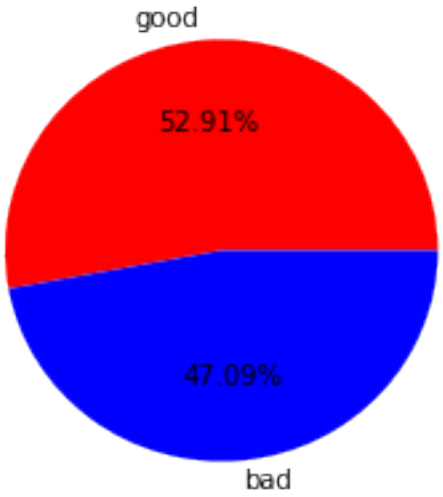
```
In [61]: df.status[df["d"]==4].value_counts()
         df.d
```

```
Out[61]: 0     4
         1     4
         2     4
         3     4
         5     4
         6     4
         7     4
         8     4
         9     4
         10    4
         12    4
         13    4
         14    4
         15    4
         16    4
         17    4
         18    4
         19    4
         20    4
         21    4
```

```
22      4
23      4
24      4
25      4
26      4
28      4
29      4
30      4
31      4
32      4
       ..
1566    4
1568    4
1569    4
1570    4
1571    4
1572    4
1573    4
1574    4
1575    4
1576    4
1577    4
1578    4
1579    4
1580    4
1582    4
1583    4
1584    4
1585    4
1586    4
1587    4
1588    4
1589    4
1590    4
1591    4
1592    4
1593    4
1594    4
1595    4
1597    4
1598    4
Name: d, Length: 1359, dtype: category
Categories (4, int64): [1 < 2 < 3 < 4]
```

```
In [237]: #plt.subplot(2,2,1)
          #plt.title('Quality wrt Density')
          #plt.pie(df.status[df["d"]==1].value_counts(),colors=("red","blue"), labels=("good",
```

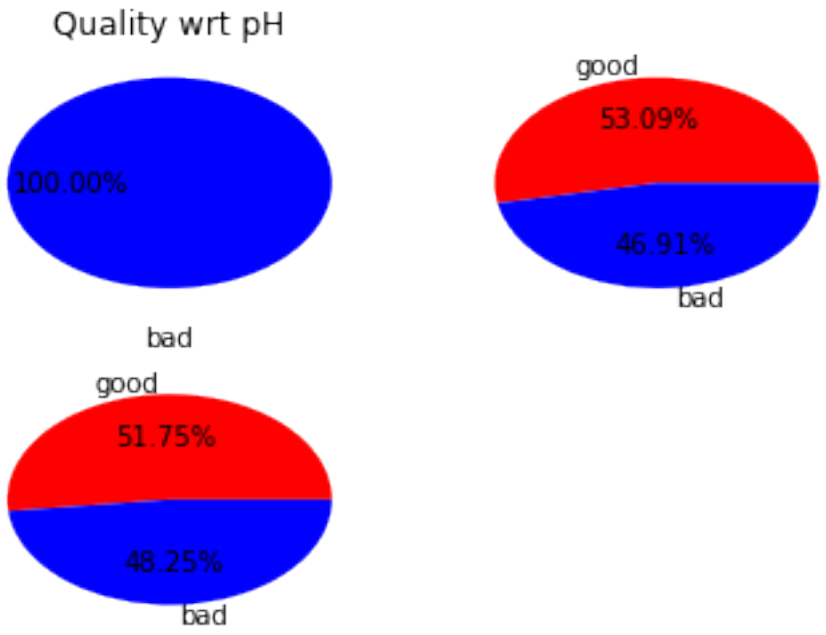
```
#plt.subplot(2,2,2)
#plt.pie(df.status[df["d"]==2].value_counts(),colors=("red","blue"), labels=("good",

#plt.subplot(2,2,3)
#plt.pie(df.status[df["d"]==3].value_counts(),colors=("red","blue"), labels=("good",

plt.subplot(2,2,4)
plt.pie(df.status[df["d"]==4].value_counts(),colors=("red","blue"), labels=("good","
fig = plt.gcf()
fig.set_size_inches(8,8)
plt.show()
```



1.0.8 Quality wrt pH

```
In [69]: df.status[df["p"]==1].value_counts()

Out[69]: 0    1
         Name: status, dtype: int64

In [67]: plt.subplot(2,2,1)
         plt.title('Quality wrt pH')
         plt.pie(df.status[df["p"]==1].value_counts(),colors=("blue"),autopct='%0.2f%%')
         plt.xlabel('bad')


         plt.subplot(2,2,2)
         plt.pie(df.status[df["p"]==2].value_counts(),colors=("red","blue"), labels=("good","ba
```

```
plt.subplot(2,2,3)
plt.pie(df.status[df["p"]==3].value_counts(),colors=("red","blue"), labels=("good","ba

#plt.subplot(2,2,4)
#plt.pie(df.status[df["p"]==4].value_counts(),colors=("blue","red"), labels=("bad","g

plt.show()
```

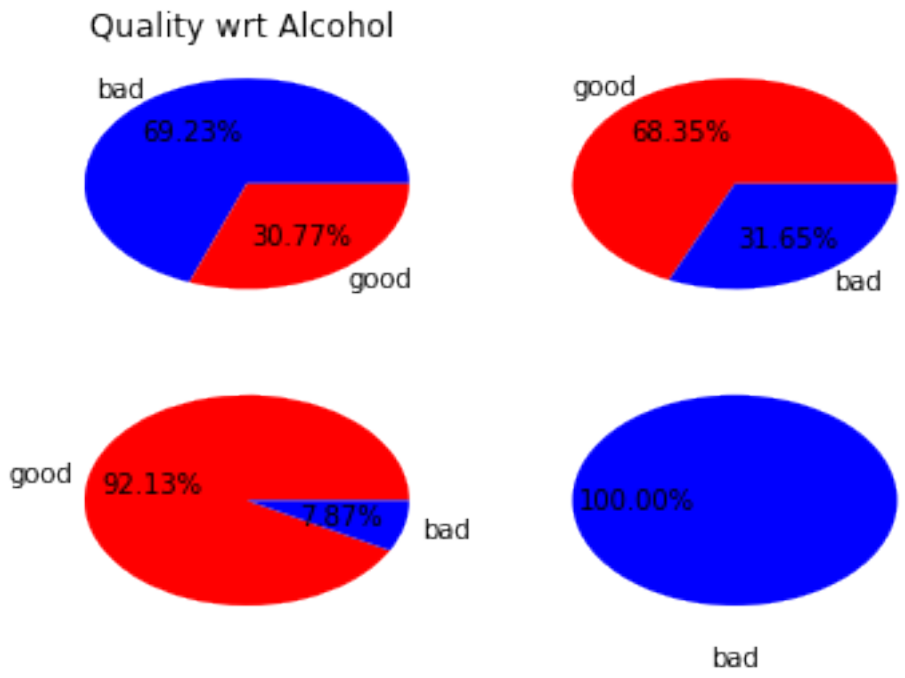


1.0.9 Quality wrt Alcohol

```
In [ ]: df.status[df["al"]==4].value_counts()
```

```
In [70]: plt.subplot(2,2,1)
         plt.title('Quality wrt Alcohol ')
         plt.pie(df.status[df["al"]==1].value_counts(),colors=("blue","red"), labels=("bad","g

         plt.subplot(2,2,2)
         plt.pie(df.status[df["al"]==2].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,3)
         plt.pie(df.status[df["al"]==3].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,4)
         plt.pie(df.status[df["al"]==4].value_counts(),colors=("blue"),autopct='%0.2f%%')
         plt.xlabel('bad')
```

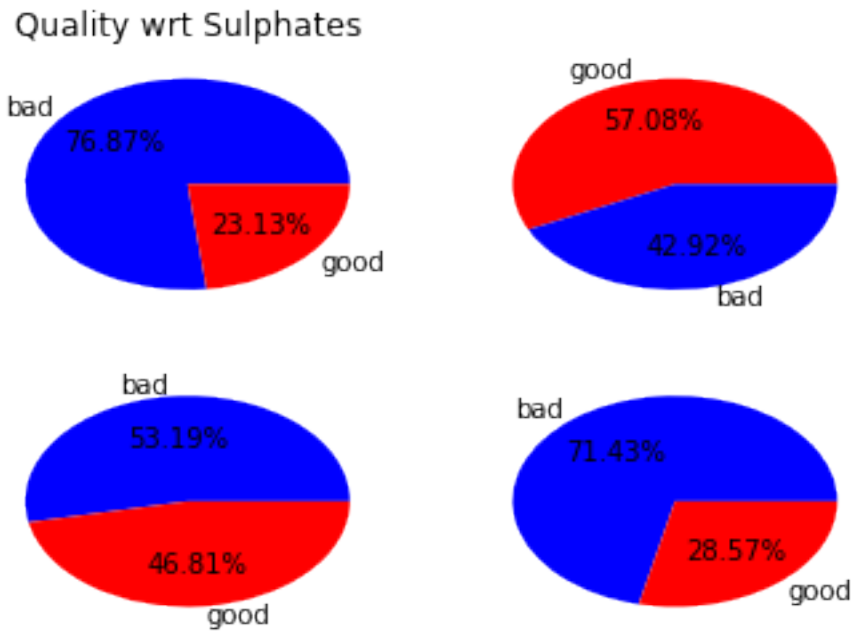
```
plt.show()
```



1.0.10 Quality wrt Sulphates

```
In [ ]: df.status[df["sul"]==4].value_counts()
```

```
In [71]: plt.subplot(2,2,1)
         plt.title('Quality wrt Sulphates')
         plt.pie(df.status[df["sul"]==1].value_counts(),colors=("blue","red"), labels=("bad","

         plt.subplot(2,2,2)
         plt.pie(df.status[df["sul"]==2].value_counts(),colors=("red","blue"), labels=("good","

         plt.subplot(2,2,3)
         plt.pie(df.status[df["sul"]==3].value_counts(),colors=("blue","red"), labels=("bad","

         plt.subplot(2,2,4)
         plt.pie(df.status[df["sul"]==4].value_counts(),colors=("blue","red"), labels=("bad","

         plt.show()
```



1.1 Machine Learning algorithm - Decision Tree

1.1.1 Splitting dataset into- Training set and Testing set

```
In [202]: train,test= train_test_split(df, test_size= 0.15)

In [203]: print("Training size: {}; Test size: {}".format(len(train),len(test)))

Training size: 1155; Test size: 204


In [204]: train.shape

Out[204]: (1155, 24)

In [205]: test.shape

Out[205]: (204, 24)
```

1.2 Generating Decision Tree

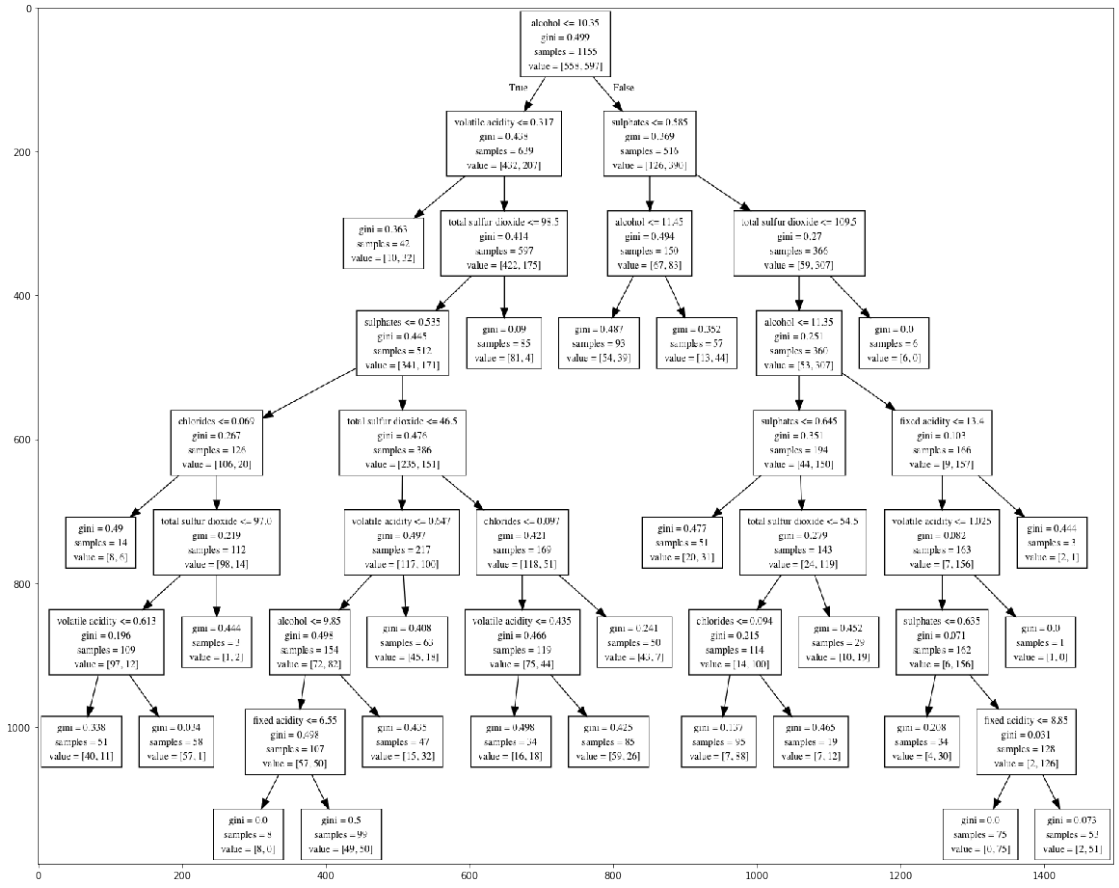
```
In [206]: c= DecisionTreeClassifier(min_samples_split=100)

In [207]: df.head()

Out[207]:    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
          0            7.4              0.70         0.00             1.9      0.076
```

```
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
5            7.4              0.66         0.00             1.8      0.075

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  ...    \
0                 11.0                  34.0   0.9978  3.51       0.56  ...
1                 25.0                  67.0   0.9968  3.20       0.68  ...
2                 15.0                  54.0   0.9970  3.26       0.65  ...
3                 17.0                  60.0   0.9980  3.16       0.58  ...
5                 13.0                  40.0   0.9978  3.51       0.56  ...

   ca  rs cl free_sd total_sd  d  p sul al status
0   1   1  1       1        1  4  3   2  1      0
1   1   1  1       1        1  4  2   2  1      0
2   1   1  1       1        1  4  2   2  1      0
3   3   1  1       1        1  4  2   2  1      1
5   1   1  1       1        1  4  3   2  1      0

[5 rows x 24 columns]
```

1.3 Feature Selection and Extraction

```
In [208]: #features=["fixed acidity","volatile acidity","citric acid","residual sugar", "chlor
          #features=["fa","va","ca","rs", "cl","free_sd","total_sd","d","p","sul","al"]
          #features=["fa","va","cl","total_sd","sul","al"]
          features=["fixed acidity","volatile acidity", "chlorides","total sulfur dioxide","su

In [209]: x_train=train[features]
          y_train=train["status"]

          x_test=test[features]
          y_test=test["status"]

In [210]: dt= c.fit(x_train,y_train)

In [211]: def show_tree(tree,features,path):
              f=io.StringIO()
              export_graphviz(tree, out_file=f,feature_names=features)
              pydotplus.graph_from_dot_data(f.getvalue()).write_png(path)
              img=misc.imread(path)
              plt.rcParams["figure.figsize"]=(20,20)
              plt.imshow(img)

In [212]: show_tree(dt,features,'dec_tree_01.png')

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5: DeprecationWarning: `imread` is
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

"""



In [213]: y_pred=c.predict(x_test)

In [214]: y_pred

Out[214]: array(['0', '1', '0', '1', '1', '0', '1', '0', '1', '1', '0', '1', '1',
'1', '1', '0', '0', '1', '1', '1', '0', '0', '0', '1', '1', '1',
'1', '1', '1', '0', '1', '1', '1', '1', '1', '0', '1', '1', '0',
'0', '0', '1', '1', '1', '1', '1', '1', '0', '0', '0', '1', '1',
'0', '1', '0', '1', '1', '0', '1', '0', '1', '0', '1', '0', '1',
'1', '1', '1', '1', '1', '1', '0', '1', '0', '0', '0', '0', '1',
'0', '1', '0', '1', '0', '1', '0', '1', '0', '1', '1', '1', '1',
'1', '1', '0', '1', '1', '1', '0', '0', '0', '1', '1', '1', '1',
'1', '0', '0', '0', '1', '1', '1', '1', '1', '1', '1', '1', '0',
'1', '0', '1', '1', '1', '1', '0', '1', '1', '0', '0', '0', '1',
'1', '1', '1', '0', '1', '0', '1', '1', '1', '0', '0', '1', '0',
'1', '0', '1', '1', '1', '0', '1', '0', '1', '0', '1', '1', '1',
'1', '0', '0', '0', '1', '0', '0', '1', '1', '1', '1', '0', '0',

```
       '1', '1', '0', '1', '1', '1', '1', '1', '1', '0', '1', '1', '1',
       '1', '1', '0', '0', '1', '1', '0', '1', '0', '0', '0', '1', '1',
       '1', '0', '1', '0', '0', '1', '0', '0', '0'], dtype=object)
```

```
In [215]: from sklearn.metrics import accuracy_score
```

```
In [216]: score= accuracy_score(y_test, y_pred)*100
```

```
In [217]: print("Accuracy using Decision tree",round(score,2),"%")
```

```
Accuracy using Decision tree 75.0 %
```