Red Wine Quality Data Analytics using Numpy

This project analyzes the **Red and White Wine Quality datasets** using **NumPy**. The goal is to perform exploratory data analysis and demonstrate NumPy operations such as aggregation, filtering, reshaping, sorting, and combining datasets.

Importing modules for numpy

In [1]:
```
import numpy as np
```

np.genfromtxt() - numpy func used to load data from text files like CSV,TSV, TXT into array

In [2]:
```
# loading the csv into array
wines = np.genfromtxt("winequality-red.csv", delimiter=",", skip_header=1)
```

In [3]:
```
# size
wines.shape
```

Out[3]: (1599, 12)

In [4]:
```
# dimension
wines.ndim
```

Out[4]: 2

In [5]:
```
# rows and column
row, col = wines.shape
print(row)
print(col)
```

```
1599
12
```

In [6]:
```
# data type
wines.dtype
```

Out[6]: dtype('float64')

In [7]:
```
# type of wines
type(wines)
```

Out[7]: numpy.ndarray

In [8]:
```
# top 5 rows
wines[0:5]
```

Out[8]:
```
array([[7.400e+00, 7.000e-01, 0.000e+00, 1.900e+00, 7.600e-02, 1.100e+01,
        3.400e+01, 9.978e-01, 3.510e+00, 5.600e-01, 9.400e+00, 5.000e+00],
       [7.800e+00, 8.800e-01, 0.000e+00, 2.600e+00, 9.800e-02, 2.500e+01,
        6.700e+01, 9.968e-01, 3.200e+00, 6.800e-01, 9.800e+00, 5.000e+00],
       [7.800e+00, 7.600e-01, 4.000e-02, 2.300e+00, 9.200e-02, 1.500e+01,
        5.400e+01, 9.970e-01, 3.260e+00, 6.500e-01, 9.800e+00, 5.000e+00],
       [1.120e+01, 2.800e-01, 5.600e-01, 1.900e+00, 7.500e-02, 1.700e+01,
        6.000e+01, 9.980e-01, 3.160e+00, 5.800e-01, 9.800e+00, 6.000e+00],
       [7.400e+00, 7.000e-01, 0.000e+00, 1.900e+00, 7.600e-02, 1.100e+01,
        3.400e+01, 9.978e-01, 3.510e+00, 5.600e-01, 9.400e+00, 5.000e+0
0]])
```

In [9]:
```
# value at 3rd row 4th col
wines[2,3]
```

Out[9]:
```
2.3
```

In [10]:
```
# first 3 items in 4th col
wines[0:3,3]
```

Out[10]:
```
array([1.9, 2.6, 2.3])
```

In [12]:
```
# first column
wines[:,0]
```

Out[12]:
```
array([7.4, 7.8, 7.8, ..., 6.3, 5.9, 6. ])
```

In [13]:
```
# second row
wines[1,:]
```

Out[13]:
```
array([ 7.8   ,  0.88  ,  0.    ,  2.6   ,  0.098 , 25.    , 67.    ,
        0.9968,  3.2   ,  0.68  ,  9.8   ,  5.    ])
```

In [15]:
```
# items from rows 1 to 3 and 5th column
wines[1:4,4]
```

Out[15]:
```
array([0.098, 0.092, 0.075])
```

In [16]:
```
# entire array
print(wines)
```

```
[[ 7.4    0.7    0.    ...  0.56   9.4    5.   ]
 [ 7.8    0.88   0.    ...  0.68   9.8    5.   ]
 [ 7.8    0.76   0.04  ...  0.65   9.8    5.   ]
 ...
 [ 6.3    0.51   0.13  ...  0.75  11.     6.   ]
 [ 5.9    0.645  0.12  ...  0.71  10.2    5.   ]
 [ 6.     0.31   0.47  ...  0.66  11.     6.   ]]
```

In [18]:
```
# changing 1st value into 100
print(wines[0,0])
wines[0,0] = 100
print(wines[0,0])
```

```
7.4
100.0
```

In [19]:
```
# changing back to 7.4
wines[0,0] = 7.4
print(wines[0,0])
```

```
7.4
```

1-Dimensional numpy arrays

In [20]:
```
# 4th row all columns
wines[3,:]
```

Out[20]:
```
array([11.2  ,  0.28 ,  0.56 ,  1.9  ,  0.075, 17.   , 60.   ,  0.998,
        3.16 ,  0.58 ,  9.8  ,  6.   ])
```

In [21]:
```
# 2nd value in 4th row
wines[3,1]
```

Out[21]:
```
0.28
```

In [22]:
```
# converting wine data into integer values
wines.astype(int)
```

Out[22]:
```
array([[ 7,  0,  0, ...,  0,  9,  5],
       [ 7,  0,  0, ...,  0,  9,  5],
       [ 7,  0,  0, ...,  0,  9,  5],
       ...,
       [ 6,  0,  0, ...,  0, 11,  6],
       [ 5,  0,  0, ...,  0, 10,  5],
       [ 6,  0,  0, ...,  0, 11,  6]])
```

Vectorization operations

In [23]:
```
# increasing wine quality score by 10
print(wines[:,11])
```

```
[5. 5. 5. ... 6. 5. 6.]
```

In [24]:
```
wines[:,11]+10
```

Out[24]:
```
array([15., 15., 15., ..., 16., 15., 16.])
```

In [25]:
```
# multiplying alcohol of all wine data by 3
print(wines[:,10])
```

```
[ 9.4  9.8  9.8 ... 11.  10.2 11. ]
```

In [26]:
```
wines[:,10]*3
```

Out[26]:
```
array([28.2, 29.4, 29.4, ..., 33. , 30.6, 33. ])
```

Broadcasting

In [27]:
```
# adding every row of wines data with random array of values
```

In [28]:
```
num_cols = col
random_arr = np.random.rand(num_cols)
result = wines + random_arr
```

In [29]:
```
print(random_arr)
```

```
[0.48344637 0.99978372 0.84358806 0.46444581 0.21458066 0.7993041
 0.31131623 0.70924103 0.34359435 0.30167519 0.5198527  0.63798177]
```

In [31]:
```
print(result)
```

```
[[ 7.88344637  1.69978372  0.84358806 ...  0.86167519  9.9198527
   5.63798177]
 [ 8.28344637  1.87978372  0.84358806 ...  0.98167519 10.3198527
   5.63798177]
 [ 8.28344637  1.75978372  0.88358806 ...  0.95167519 10.3198527
   5.63798177]
 ...
 [ 6.78344637  1.50978372  0.97358806 ...  1.05167519 11.5198527
   6.63798177]
 [ 6.38344637  1.64478372  0.96358806 ...  1.01167519 10.7198527
   5.63798177]
 [ 6.48344637  1.30978372  1.31358806 ...  0.96167519 11.5198527
   6.63798177]]
```

Numpy Aggregation methods

In [32]:
```
# sum of all residual sugar values
np.sum(wines[:,3])
```

Out[32]:
```
4059.55
```

In [33]:
```
# sum of every feature(column) value
np.sum(wines, axis=0)
```

Out[33]:
```
array([13303.1    ,   843.985  ,   433.29   ,  4059.55   ,   139.859  ,
       25384.     , 74302.     ,  1593.79794,  5294.47   ,  1052.38   ,
       16666.35   ,  9012.     ])
```

In [34]:
```
# sum of every row
np.sum(wines, axis=1)
```

Out[34]:
```
array([ 74.5438 , 123.0548 ,  99.699  , ..., 100.48174, 105.21547,
        92.49249])
```

In [35]:
```
# maximum residual sugar value in integer
np.max(wines[:,3]).astype(int)
```

Out[35]:
```
15
```

In [36]:
```
# minimum residual sugar value in integer
np.min(wines[:,3]).astype(int)
```

Out[36]:
```
0
```

In [37]:
```
# average residual sugar value
np.mean(wines[:,3])
```

Out[37]:
```
2.53880550343965
```

In [38]:
```
# 25 percentile of residual sugar
np.percentile(wines[:,3],25)
```

Out[38]:
```
1.9
```

In [39]:
```
# 25 percentile of residual sugar
np.percentile(wines[:,3],75)
```

Out[39]:
```
2.6
```

In [40]:
```
# average of each feature value
np.mean(wines, axis=0)
```

Out[40]:
```
array([ 8.31963727,  0.52782051,  0.27097561,  2.5388055 ,  0.08746654,
       15.87492183, 46.46779237,  0.99674668,  3.3111132 ,  0.65814884,
       10.42298311,  5.63602251])
```

Numpy Array comparisons

In [41]:
```
# wines with quality >5
wines[:,11]>5
```

Out[41]:
```
array([False, False, False, ...,  True, False,  True])
```

In [42]:
```
# wines with quality >7
wines[:,11]>7
```

Out[42]:
```
array([False, False, False, ..., False, False, False])
```

In [43]:
```
# checking any wine value is true for the condition quality > 7
np.any(wines[:,11]>7)
```

Out[43]:
```
True
```

In [45]:
```
# top 3 rows and all columns of wine quality that is > 7
wines[wines[:,11]>7][:3,:]
```

Out[45]:
```
array([[7.900e+00, 3.500e-01, 4.600e-01, 3.600e+00, 7.800e-02, 1.500e+01,
        3.700e+01, 9.973e-01, 3.350e+00, 8.600e-01, 1.280e+01, 8.000e+00],
       [1.030e+01, 3.200e-01, 4.500e-01, 6.400e+00, 7.300e-02, 5.000e+00,
        1.300e+01, 9.976e-01, 3.230e+00, 8.200e-01, 1.260e+01, 8.000e+00],
       [5.600e+00, 8.500e-01, 5.000e-02, 1.400e+00, 4.500e-02, 1.200e+01,
        8.800e+01, 9.924e-01, 3.560e+00, 8.200e-01, 1.290e+01, 8.000e+0
0]])
```

In [46]:
```
# wines with lot of alcohol >10 and wine quality >7
wines[(wines[:,10]>10)&(wines[:,11]>7)]
```

Out[46]:
```
array([[7.9000e+00, 3.5000e-01, 4.6000e-01, 3.6000e+00, 7.8000e-02,
        1.5000e+01, 3.7000e+01, 9.9730e-01, 3.3500e+00, 8.6000e-01,
        1.2800e+01, 8.0000e+00],
       [1.0300e+01, 3.2000e-01, 4.5000e-01, 6.4000e+00, 7.3000e-02,
        5.0000e+00, 1.3000e+01, 9.9760e-01, 3.2300e+00, 8.2000e-01,
        1.2600e+01, 8.0000e+00],
       [5.6000e+00, 8.5000e-01, 5.0000e-02, 1.4000e+00, 4.5000e-02,
        1.2000e+01, 8.8000e+01, 9.9240e-01, 3.5600e+00, 8.2000e-01,
        1.2900e+01, 8.0000e+00],
       [1.1300e+01, 6.2000e-01, 6.7000e-01, 5.2000e+00, 8.6000e-02,
        6.0000e+00, 1.9000e+01, 9.9880e-01, 3.2200e+00, 6.9000e-01,
        1.3400e+01, 8.0000e+00],
       [9.4000e+00, 3.0000e-01, 5.6000e-01, 2.8000e+00, 8.0000e-02,
        6.0000e+00, 1.7000e+01, 9.9640e-01, 3.1500e+00, 9.2000e-01,
        1.1700e+01, 8.0000e+00],
       [1.0700e+01, 3.5000e-01, 5.3000e-01, 2.6000e+00, 7.0000e-02,
        5.0000e+00, 1.6000e+01, 9.9720e-01, 3.1500e+00, 6.5000e-01,
        1.1000e+01, 8.0000e+00],
       [1.0700e+01, 3.5000e-01, 5.3000e-01, 2.6000e+00, 7.0000e-02,
        5.0000e+00, 1.6000e+01, 9.9720e-01, 3.1500e+00, 6.5000e-01,
        1.1000e+01, 8.0000e+00],
       [5.0000e+00, 4.2000e-01, 2.4000e-01, 2.0000e+00, 6.0000e-02,
        1.9000e+01, 5.0000e+01, 9.9170e-01, 3.7200e+00, 7.4000e-01,
        1.4000e+01, 8.0000e+00],
       [7.8000e+00, 5.7000e-01, 9.0000e-02, 2.3000e+00, 6.5000e-02,
        3.4000e+01, 4.5000e+01, 9.9417e-01, 3.4600e+00, 7.4000e-01,
        1.2700e+01, 8.0000e+00],
       [9.1000e+00, 4.0000e-01, 5.0000e-01, 1.8000e+00, 7.1000e-02,
        7.0000e+00, 1.6000e+01, 9.9462e-01, 3.2100e+00, 6.9000e-01,
        1.2500e+01, 8.0000e+00],
       [1.0000e+01, 2.6000e-01, 5.4000e-01, 1.9000e+00, 8.3000e-02,
        4.2000e+01, 7.4000e+01, 9.9451e-01, 2.9800e+00, 6.3000e-01,
        1.1800e+01, 8.0000e+00],
       [7.9000e+00, 5.4000e-01, 3.4000e-01, 2.5000e+00, 7.6000e-02,
        8.0000e+00, 1.7000e+01, 9.9235e-01, 3.2000e+00, 7.2000e-01,
        1.3100e+01, 8.0000e+00],
       [8.6000e+00, 4.2000e-01, 3.9000e-01, 1.8000e+00, 6.8000e-02,
        6.0000e+00, 1.2000e+01, 9.9516e-01, 3.3500e+00, 6.9000e-01,
        1.1700e+01, 8.0000e+00],
       [5.5000e+00, 4.9000e-01, 3.0000e-02, 1.8000e+00, 4.4000e-02,
        2.8000e+01, 8.7000e+01, 9.9080e-01, 3.5000e+00, 8.2000e-01,
        1.4000e+01, 8.0000e+00],
       [7.2000e+00, 3.8000e-01, 3.1000e-01, 2.0000e+00, 5.6000e-02,
        1.5000e+01, 2.9000e+01, 9.9472e-01, 3.2300e+00, 7.6000e-01,
        1.1300e+01, 8.0000e+00],
       [7.4000e+00, 3.6000e-01, 3.0000e-01, 1.8000e+00, 7.4000e-02,
        1.7000e+01, 2.4000e+01, 9.9419e-01, 3.2400e+00, 7.0000e-01,
        1.1400e+01, 8.0000e+00]])
```

In [47]:
```
# only alcohol and quality columns from above
wines[(wines[:,10]>10)&(wines[:,11]>7)][:,[10,11]]
```

Out[47]:
```
array([[12.8,  8. ],
       [12.6,  8. ],
       [12.9,  8. ],
       [13.4,  8. ],
       [11.7,  8. ],
       [11. ,  8. ],
       [11. ,  8. ],
       [14. ,  8. ],
       [12.7,  8. ],
       [12.5,  8. ],
       [11.8,  8. ],
       [13.1,  8. ],
       [11.7,  8. ],
       [14. ,  8. ],
       [11.3,  8. ],
       [11.4,  8. ]])
```

Combining Numpy arrays

In [48]:
```
# combine red wine and white wine data
```

In [52]:
```
white_wines = np.genfromtxt("winequality-white.csv", delimiter =";", skip_he
```

In [53]:
```
white_wines
```

Out[53]:
```
array([[ 7.  ,  0.27,  0.36, ...,  0.45,  8.8 ,  6.  ],
       [ 6.3 ,  0.3 ,  0.34, ...,  0.49,  9.5 ,  6.  ],
       [ 8.1 ,  0.28,  0.4 , ...,  0.44, 10.1 ,  6.  ],
       ...,
       [ 6.5 ,  0.24,  0.19, ...,  0.46,  9.4 ,  6.  ],
       [ 5.5 ,  0.29,  0.3 , ...,  0.38, 12.8 ,  7.  ],
       [ 6.  ,  0.21,  0.38, ...,  0.32, 11.8 ,  6.  ]])
```

In [55]:
```
# combine wines and white wines using vstack
all_wines = np.vstack((wines,white_wines))
all_wines
```

Out[55]:
```
array([[ 7.4 ,  0.7 ,  0.  , ...,  0.56,  9.4 ,  5.  ],
       [ 7.8 ,  0.88,  0.  , ...,  0.68,  9.8 ,  5.  ],
       [ 7.8 ,  0.76,  0.04, ...,  0.65,  9.8 ,  5.  ],
       ...,
       [ 6.5 ,  0.24,  0.19, ...,  0.46,  9.4 ,  6.  ],
       [ 5.5 ,  0.29,  0.3 , ...,  0.38, 12.8 ,  7.  ],
       [ 6.  ,  0.21,  0.38, ...,  0.32, 11.8 ,  6.  ]])
```

In [56]:
```
# combine using concatenate
data = np.concatenate((wines,white_wines))
data
```

Out[56]:
```
array([[ 7.4 ,  0.7 ,  0.  , ...,  0.56,  9.4 ,  5.  ],
       [ 7.8 ,  0.88,  0.  , ...,  0.68,  9.8 ,  5.  ],
       [ 7.8 ,  0.76,  0.04, ...,  0.65,  9.8 ,  5.  ],
       ...,
       [ 6.5 ,  0.24,  0.19, ...,  0.46,  9.4 ,  6.  ],
       [ 5.5 ,  0.29,  0.3 , ...,  0.38, 12.8 ,  7.  ],
       [ 6.  ,  0.21,  0.38, ...,  0.32, 11.8 ,  6.  ]])
```

Matrix operations and reshape

In [57]:
```
# transpose of wines and its size
wines.T.shape
```

Out[57]:
```
(12, 1599)
```

In [58]:
```
# wines data into 1D array
wines.flatten()
```

Out[58]:
```
array([ 7.4 ,  0.7 ,  0.  , ...,  0.66, 11.  ,  6.  ])
```

In [59]:
```
wines.size
```

Out[59]:
```
19188
```

In [60]:
```
# reshaping 2nd row of wines into 2D with 2 rows and 6 columns
wines[1,:].reshape(2,6)
```

Out[60]:
```
array([[ 7.8   ,  0.88  ,  0.    ,  2.6   ,  0.098 , 25.    ],
       [67.    ,  0.9968,  3.2   ,  0.68  ,  9.8   ,  5.    ]])
```

In [62]:
```
# sort alcohol column in ascending order
sort_alcohol = wines[:,10][np.argsort(wines[:,10])]
sort_alcohol
```

Out[62]:
```
array([ 8.4,  8.4,  8.5, ..., 14. , 14. , 14.9])
```

In [64]:
```
sort_alcohol[:10]
```

Out[64]:
```
array([8.4, 8.4, 8.5, 8.7, 8.7, 8.8, 8.8, 9. , 9. , 9. ])
```

In [65]:
```
# sort alcohol column in descending order
sort_alcohol_des = wines[:,10][np.argsort(wines[:,10])][::-1]
sort_alcohol_des
```

Out[65]:
```
array([14.9, 14. , 14. , ...,  8.5,  8.4,  8.4])
```

In [ ]:
```

```