20250531_01

May 31, 2025

[1]:
```
import pandas as pd
```

[9]:
```
# Load the wine dataset from UCI ML Repo
url_wine = "https://archive.ics.uci.edu/ml/machine-learning-databases/
 ↪wine-quality/winequality-white.csv"
wine = pd.read_csv(url_wine, sep = ';')
```

[10]:
```
# Take a look
wine.info()
wine.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

[10]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	4898.000000	4898.000000	4898.000000	4898.000000	
mean	6.854788	0.278241	0.334192	6.391415	
std	0.843868	0.100795	0.121020	5.072058	
min	3.800000	0.080000	0.000000	0.600000	
25%	6.300000	0.210000	0.270000	1.700000	
50%	6.800000	0.260000	0.320000	5.200000	

```
75%         7.300000          0.320000     0.390000        9.900000
max        14.200000          1.100000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  4898.000000          4898.000000           4898.000000  4898.000000
mean      0.045772            35.308085            138.360657     0.994027
std       0.021848            17.007137             42.498065     0.002991
min       0.009000             2.000000              9.000000     0.987110
25%       0.036000            23.000000            108.000000     0.991723
50%       0.043000            34.000000            134.000000     0.993740
75%       0.050000            46.000000            167.000000     0.996100
max       0.346000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol      quality
count  4898.000000  4898.000000  4898.000000  4898.000000
mean      3.188267     0.489847    10.514267     5.877909
std       0.151001     0.114126     1.230621     0.885639
min       2.720000     0.220000     8.000000     3.000000
25%       3.090000     0.410000     9.500000     5.000000
50%       3.180000     0.470000    10.400000     6.000000
75%       3.280000     0.550000    11.400000     6.000000
max       3.820000     1.080000    14.200000     9.000000
```

[11]:
```
from scipy.stats import zscore
import numpy as np
```

[23]:
```
# Z-scores means making (roughly) normal distributino to standard normal␣
 ↪distribution

# Compute Z-scores for all columns
z_scores = zscore(wine)

# Convert to DataFrame for easier handling
z_df = pd.DataFrame(z_scores, columns = wine.columns)

# Find rows with any |Z| > 3 (outliers), since -3 <= Z <= 3 comprise 99.7% of␣
 ↪the datas.
# And if any one of the columns in a row(data) is marked as outlier, make the␣
 ↪entire data as outlier.
# Since all the fectors are kinda related, so it make sense to mark the entire␣
 ↪data as an outlier even if only one fector is off.
outliers = (np.abs(z_df) > 3).any(axis = 1)

# Print outcome
print("Total rows:", len(wine))
print("Outliers detected (Z > 3):", outliers.sum())
```

```
Total rows: 4898
Outliers detected (Z > 3): 411
```

[38]:
```
# I want to check outliers by cols
(np.abs(z_df) > 3).sum()
```

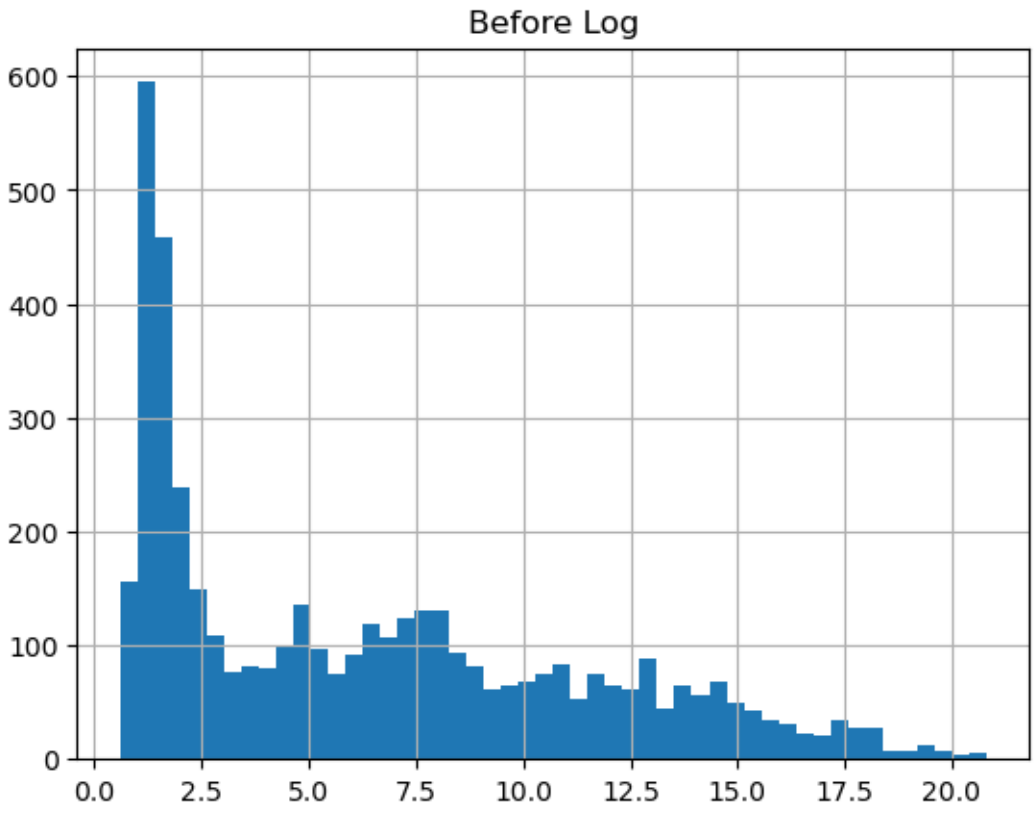
[38]:
```
fixed acidity           46
volatile acidity        81
citric acid             85
residual sugar           9
chlorides              102
free sulfur dioxide     32
total sulfur dioxide    12
density                  3
pH                      32
sulphates               48
alcohol                  0
quality                 25
dtype: int64
```

[25]:
```
# Remove outliers
wine_cleaned = wine[outliers == False]

# Check result
print("Original shape:", wine.shape)
print("Cleaned shape:", wine_cleaned.shape)
```

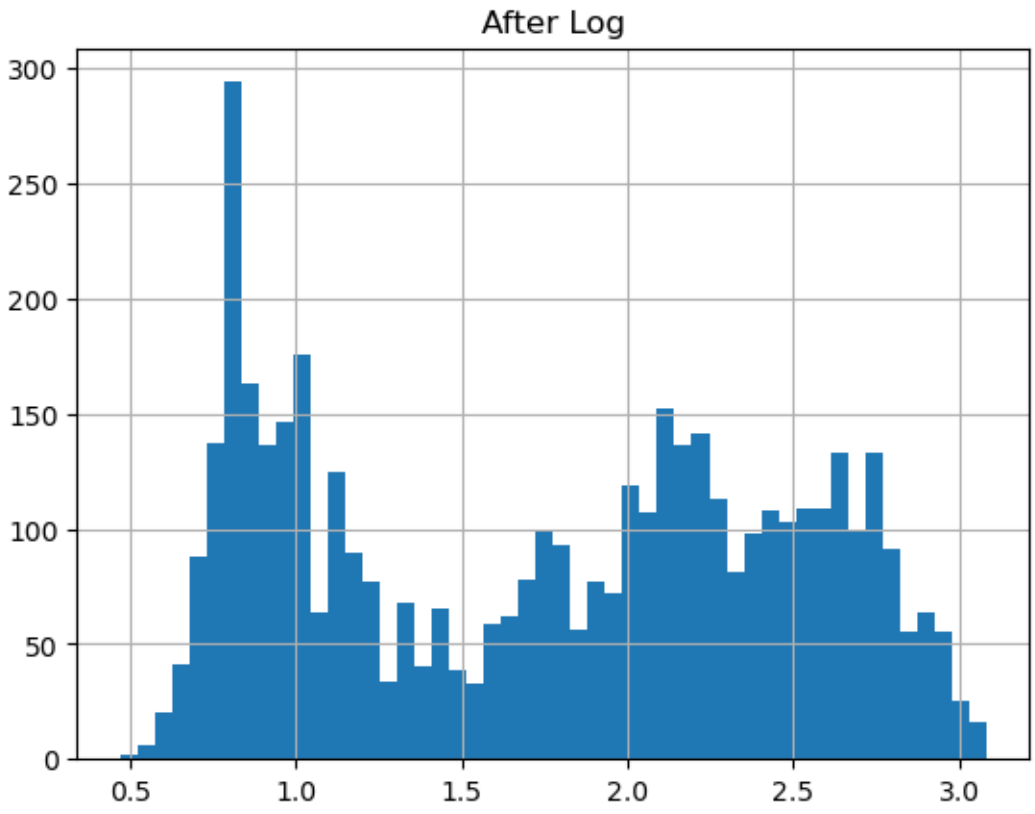
```
Original shape: (4898, 12)
Cleaned shape: (4487, 12)
```

[26]:
```
import matplotlib.pyplot as plt
```

[32]:
```
# Now we do log Transformation (for skewed distributions)
# It's also a way to get rid of outliers, but instead of dropping them, we make␣
 ↪them more concentreted.

# Before log
wine_cleaned['residual sugar'].hist(bins = 50)
plt.title("Before Log")
plt.show()
```



[31]:
```
# Apply log1p (x -> log(x + 1))
wine_cleaned.loc[:, 'residual sugar_log'] = np.log1p(wine_cleaned['residual␣
 ↪sugar'])

# After log
wine_cleaned['residual sugar_log'].hist(bins = 50)
plt.title("After Log")
plt.show()
```


After Log
300
250
200
150
100
50
0
0.5
1.0
1.5
2.0
2.5
3.0