20250601_01

June 1, 2025

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

[2]:
```
# Load dataset from UCI
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
 ↪winequality-white.csv"
wine = pd.read_csv(url, sep=';')
```

[3]:
```
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
```

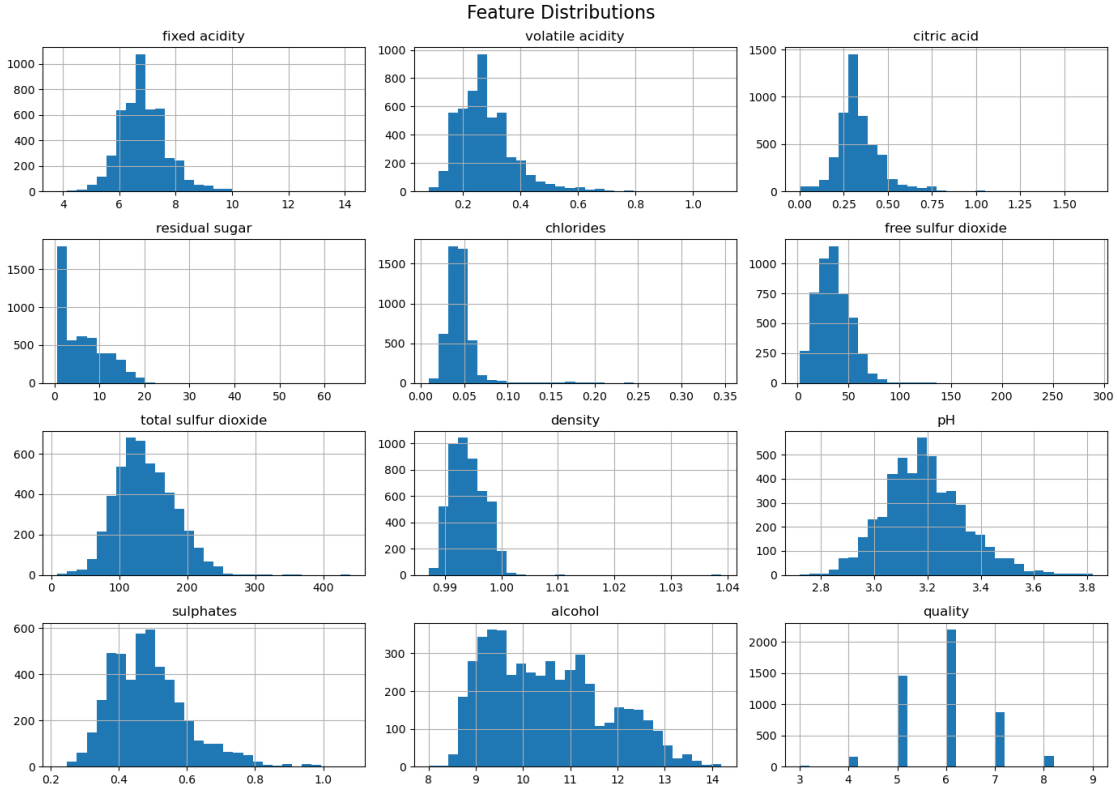
[3]:

	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	

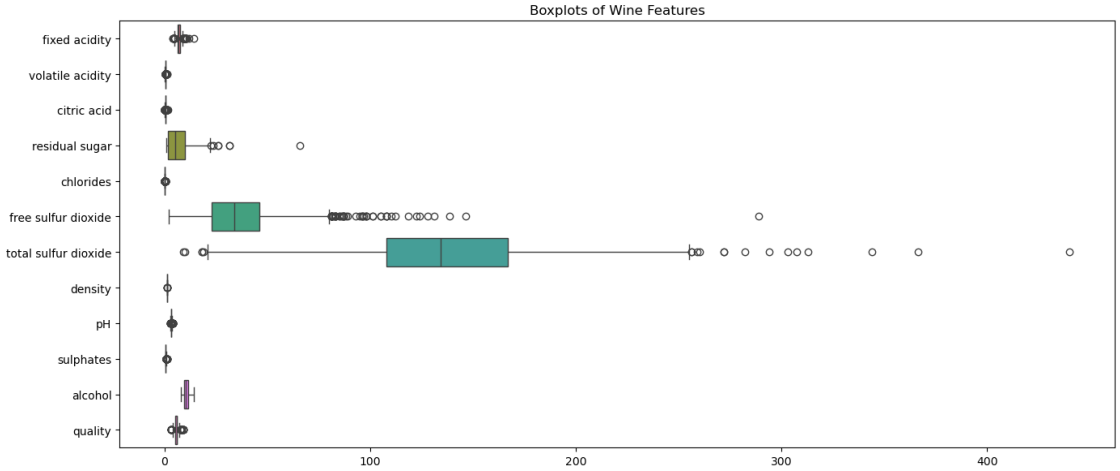
```
max        14.200000          1.100000     1.660000       65.800000

         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
```

[4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[7]:
```
# Visaulization using histograms
wine.hist(bins = 30, figsize = (14, 10))
plt.suptitle("Feature Distributions", fontsize = 16)
plt.tight_layout()
plt.show()
```



[8]:
```
# Visaulization using boxplots (horizontally for visibility)
plt.figure(figsize = (14, 6))
sns.boxplot(data = wine, orient = "h")
plt.title("Boxplots of Wine Features")
plt.tight_layout()
plt.show()
```



[11]:
```
# Create a dictionary
outlier_counts = {}

for col in wine.columns[:-1]:  # Skip 'quality' for now
    Q1 = wine[col].quantile(0.25)
    Q3 = wine[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    count = ((wine[col] < lower) | (wine[col] > upper)).sum()
    outlier_counts[col] = count

# Display as sorted series
pd.Series(outlier_counts).sort_values(ascending = False)
```

[11]:
```
citric acid             270
chlorides               208
volatile acidity        186
sulphates               124
fixed acidity           119
pH                       75
free sulfur dioxide      50
total sulfur dioxide     19
residual sugar            7
density                   5
alcohol                   0
dtype: int64
```

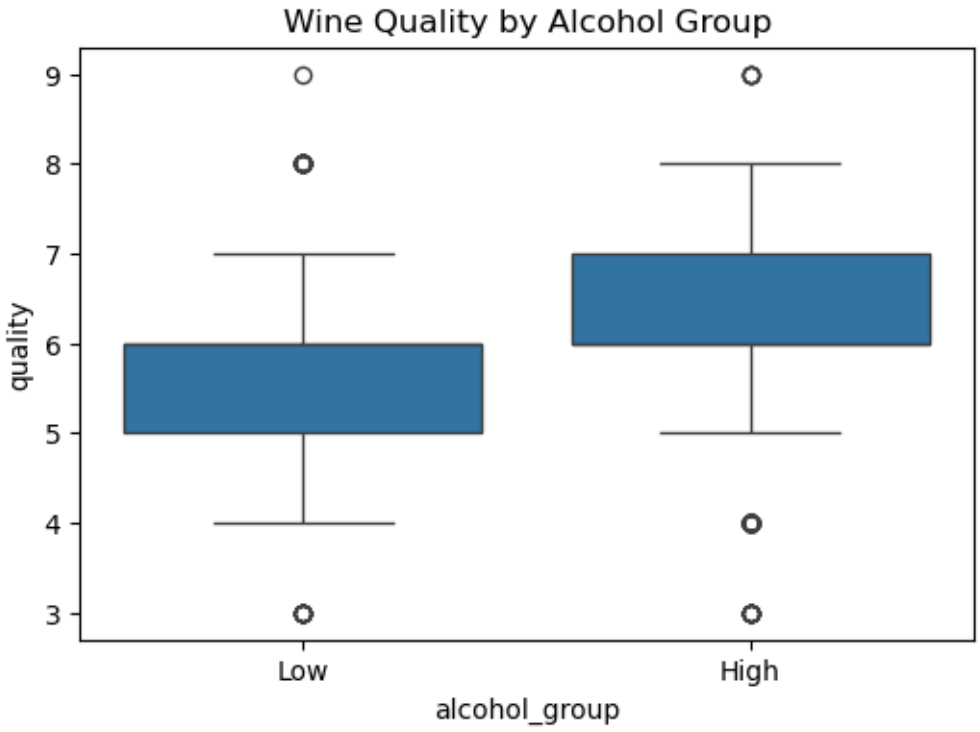
[12]:
```
# Since 'alcohol' contains no outliers, we use this factor to do hypothesis␣
 ↪test.
# H  : No difference in quality between high and low alcohol wines
# H  : There is a difference
# Using Welch's t-test since they have unequal variances
from scipy.stats import ttest_ind
```

[13]:
```
# Split by alcohol median
median_alcohol = wine['alcohol'].median()
high = wine[wine['alcohol'] > median_alcohol]['quality']
low = wine[wine['alcohol'] <= median_alcohol]['quality']

# Welch's t-test
t_stat, p_val = ttest_ind(high, low, equal_var = False)

print("t-statistic:", t_stat)
print("p-value:", p_val)
```

```
t-statistic: 28.398404160174177
p-value: 1.2734524613511852e-163
```

We **reject** the null hypothesis, which means that there is indeed a difference between the quality of high and low alcohol wine.

[14]:
```
# Add alcohol group label
wine['alcohol_group'] = wine['alcohol'].apply(lambda x: 'High' if x >␣
 ↪median_alcohol else 'Low')

# Boxplot
plt.figure(figsize = (6, 4))
sns.boxplot(data = wine, x = 'alcohol_group', y = 'quality')
plt.title("Wine Quality by Alcohol Group")
plt.show()
```



[18]:
```
# Now we do CI to conclude this lesson.
# We pretend this dataset is a random sample from a larger population of white␣
 ↪wines.
import numpy as np
```

[16]:
```
sample = wine['alcohol']
mean = sample.mean()
std = sample.std(ddof = 1)
```

```python
n = len(sample)

margin = 1.96 * (std / np.sqrt(n))
ci_low = mean - margin
ci_high = mean + margin

print(f"95% Confidence Interval for mean alcohol: ({ci_low:.3f}, {ci_high:.
  ↪3f})")
```

```
95% Confidence Interval for mean alcohol: (10.480, 10.549)
```