EXPLORATORY DATA SCIENCE

Zhabiyan Surya Wiradenish

DATA SCIENCE

IMPORT & LOAD DATA

```
[8] import pandas as pd

    df = pd.read_csv('/content/winequality-red.csv')
```

```
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

This code imports the Wine Quality dataset using the Pandas library and displays 1,599 wine samples containing various physicochemical properties such as acidity, residual sugar, pH, and alcohol content. Each sample is also labeled with a quality score that reflects the wine's overall rating. This dataset is well-suited for data profiling, pattern discovery, and predictive modeling using regression or classification techniques.

MISSING VALUE

Kode ini menggunakan library Pandas untuk memeriksa struktur dari dataset Wine Quality yang disimpan dalam variabel df. Metode df.info() memberikan ringkasan singkat mengenai dataset, termasuk:

- Jumlah total entri (1599 baris, diindeks dari 0 hingga 1598).
- Jumlah total kolom (12 kolom secara keseluruhan).
- Nama kolom, jumlah nilai yang tidak kosong (non-null), serta tipe data dari setiap kolom.
- Penggunaan memori dari DataFrame (sekitar 150.0 KB).

```
[10] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

MISSING VALUE

```
df.isna().sum()
```

	0
fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
quality	0

dtype: int64

```
[12] df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

This image shows an analysis of the Wine Quality dataset using Pandas. The df.isna().sum() function confirms there are no missing values, while df.describe() provides statistical summaries such as mean, standard deviation, minimum, maximum, and percentiles to help understand the data distribution and characteristics.

MISSING VALUE

```
[13] df['alcohol'].describe()
```

	alcohol
count	1599.000000
mean	10.422983
std	1.065668
min	8.400000
25%	9.500000
50%	10.200000
75%	11.100000
max	14.900000

dtype: float64

The output provides a statistical summary of the 'alcohol' column in the Wine Quality dataset, indicating there are 1,599 entries with no missing values. The average alcohol content is approximately 10.42%, with a standard deviation of 1.07. The values range from a minimum of 8.4% to a maximum of 14.9%. The 25th percentile is 9.5%, the median (50th percentile) is 10.2%, and the 75th percentile is 11.1%, showing a relatively even distribution of alcohol content across the samples.

MISSING VALUE

```
[14] # Mengatasi missing value
     for column in df.columns:
         if df[column].dtype == 'object':
             # Jika kolom bertipe object, isi dengan mode
             df[column].fillna(df[column].mode()[0], inplace=True)
         else:
             # Jika kolom bertipe numerik, isi dengan mean
             df[column].fillna(df[column].mean(), inplace=True)
```

```
<ipython-input-14-da1a6285f769>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original


  df[column].fillna(df[column].mean(), inplace=True)
```

The code handles missing values in a DataFrame by checking each column's data type. If a column is of type object (usually for categorical/text data), it fills missing values with the most frequent value (mode). If the column is numeric, it fills missing values with the average (mean). However, a warning is shown because the code uses chained assignment with the inplace=True argument, which is discouraged in future versions of pandas (3.0 and above). Instead, it's recommended to reassign the result directly to the column using df[column] = df[column].fillna(...) for more reliable behavior.

MISSING VALUE

```
df.isna().sum()
```

	0
fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
quality	0

dtype: int64

The code df.isna().sum() is used to check for missing values in each column of the dataset. The output shows that all columns contain o missing values, indicating that the dataset is complete and does not require any imputation or cleaning for missing data. This ensures that further analysis can be performed without handling null values.

MISSING VALUE

The df.info() output provides a summary of the dataset, indicating that it contains 1,599 entries and 12 columns. Each column has 1,599 non-null values, meaning there are no missing values in the dataset. The data types are mostly float64, except for the quality column which is int64. The total memory usage of the DataFrame is approximately 150.0 KB. This confirms the dataset is complete and well-structured for analysis.

```
[16] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

MISSING VALUE

```
[17] # Mengecek apakah ada duplicate di seluruh kolom
     check_duplicate = df.duplicated().sum()

     print(f"Jumlah data yang duplikat = {check_duplicate}")
```

```
Jumlah data yang duplikat = 240
```

The code checks for duplicate rows across all columns using df.duplicated().sum(). The output indicates that there are 240 duplicate entries in the dataset. This suggests that multiple rows contain identical values across all features and may need to be reviewed or removed for accurate analysis.

MISSING VALUE

```
[18] # Handling duplicate
     df = df.drop_duplicates()
```

The code df = df.drop_duplicates() is used to remove all duplicate rows from the dataset. By assigning the result back to df, the dataset is updated to include only unique rows, which helps improve data quality and avoid redundancy in analysis.

MISSING VALUE

```
[19] # Mengecek duplicate setelah di-handle
     handle_duplicate = df.duplicated().sum()

     print(f"Jumlah data yang duplikat = {handle_duplicate}")
```

```
Jumlah data yang duplikat = 0
```

The code checks for any remaining duplicate rows in the dataset after handling them. Since the result shows Jumlah data yang duplikat = 0, it confirms that all duplicate rows have been successfully removed and the dataset now only contains unique records.

THANK YOU

DATA SCIENCE