**Mata Kuliah Aplikasi Data Scientist**

# LAPORAN TUGAS RANDOM FOREST CLASSIFIER

**Dosen Pengampu : Ledy Elsera Astrianty, S.Kom., M.Kom**


**Disusun Oleh :**

- **Lathif Ramadhan (5231811022)**
- **Andini Angel Meivita M (5231811029)**
- **Rama Panji N (5231811033)**
- **Giffari Riyanda P (5231811036)**



**PROGRAM STUDI SAINS DATA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS TEKNOLOGI YOGYAKARTA**

**TAHUN 2025**

# Daftar Isi

# Deskripsi Dataset: Kualitas Anggur Merah "Vinho Verde"

## 1. Gambaran Umum Dataset

Dataset yang digunakan dalam analisis ini adalah data kualitas anggur merah (*red wine*) dari varian "Vinho Verde" yang berasal dari Portugal. Dataset ini menyediakan informasi kuantitatif mengenai atribut fisikokimia yang didapat dari pengujian laboratorium dan penilaian sensorik berupa skor kualitas.

Penting untuk dicatat bahwa dataset ini tidak mencakup informasi mengenai jenis anggur, merek, atau harga jual, sehingga fokus analisis murni pada hubungan antara karakteristik fisikokimia dengan kualitas anggur yang dihasilkan. Permasalahan ini dapat didekati sebagai tugas regresi (memprediksi skor kualitas secara langsung) atau klasifikasi (mengelompokkan anggur ke dalam kategori kualitas tertentu), di mana pendekatan klasifikasi menjadi fokus dalam laporan ini.

## 2.Sumber dan Sitasi

Dataset ini merupakan koleksi publik yang tersedia di **UCI Machine Learning Repository** dan telah dibagikan kembali di platform **Kaggle** untuk kemudahan akses. Berikut tautan dataset dari Kaggle yang kami gunakan:
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez- et-al-2009

## 3.Deskripsi Variabel (Atribut)

Dataset ini terdiri dari 11 variabel input (fitur) dan 1 variabel output (target). Semua fitur merupakan variabel numerik dengan tipe data `float64`.

### a. Variabel Input (Fitur Fisikokimia):

1. **`fixed acidity`** (Keasaman Tetap):
- Mengukur konsentrasi asam-asam non-volatil (tidak mudah menguap) dalam anggur, seperti asam tartarat dan malat. Berkontribusi pada rasa "tajam" pada anggur. (Tipe Data: `float64`)
1. **`volatile acidity`** (Keasaman Volatil):
- Mengukur jumlah asam asetat dalam anggur. Tingkat yang terlalu tinggi dapat menyebabkan aroma dan rasa seperti cuka yang tidak diinginkan. (Tipe Data: `float64`)
1. **`citric acid`** (Asam Sitrat)
- Dalam jumlah kecil, asam sitrat dapat menambah kesegaran dan rasa "segar" pada anggur. (Tipe Data: `float64`)
1. **`residual sugar`** (Sisa Gula):
- Jumlah gula yang tersisa setelah proses fermentasi berhenti. Ini menentukan tingkat kemanisan anggur. (Tipe Data: `float64`)
1. **`chlorides`** (Klorida):
- Jumlah garam (khususnya natrium klorida) dalam anggur. (Tipe Data: `float64`)
1. **`free sulfur dioxide`** (Belerang Dioksida Bebas):

- Bagian dari $SO_2$ yang berada dalam bentuk bebas dan berfungsi sebagai antimikroba serta antioksidan untuk mencegah pembusukan. (Tipe Data: `float64`)

1. **`total sulfur dioxide`** (Total Belerang Dioksida):
- Gabungan dari belerang dioksida dalam bentuk bebas dan terikat. Berfungsi sebagai pengawet. (Tipe Data: `float64`)

1. **`density`** (Kepadatan):
- Kepadatan anggur yang umumnya mendekati kepadatan air, dipengaruhi oleh kandungan alkohol dan gula. (Tipe Data: `float64`)

1. **`pH`**:
- Mengukur tingkat keasaman atau kebasaan pada skala pH, yang memengaruhi rasa dan stabilitas anggur. Sebagian besar anggur memiliki pH antara 3 dan 4. (Tipe Data: `float64`)

1. **`sulphates`** (Sulfat):
- Garam sulfat yang dapat berkontribusi pada efektivitas $SO_2$ dan memengaruhi rasa anggur. (Tipe Data: `float64`)

1. **`alcohol`** (Alkohol):
- Persentase kandungan alkohol dalam anggur berdasarkan volume. (Tipe Data: `float64`)

### b. Variabel Output (Target Sensorik):

1. **`quality`** (Kualitas):
- Skor kualitas yang diberikan oleh ahli berdasarkan penilaian sensorik, dengan skala asli antara 0 hingga 10. Untuk tugas klasifikasi ini, variabel ini telah ditransformasikan menjadi kategori biner:
  - **1 (Baik)**: Jika skor asli >= 7
  - **0 (Buruk)**: Jika skor asli < 7

## 4. Karakteristik Statistik dan Pembersihan Data

Analisis eksplorasi data awal mengungkapkan beberapa karakteristik penting:

- **Ukuran Data Awal**: Dataset ini pada awalnya berisi **1599 baris** data.
- **Data Duplikat**: Teridentifikasi adanya **240 baris** data yang duplikat. Baris-baris ini telah dihapus untuk memastikan integritas data dalam pemodelan.
- **Ukuran Data Final**: Setelah pembersihan, dataset yang digunakan untuk analisis terdiri dari **1359 baris dan 12 kolom**.
- **Nilai Hilang**: **Tidak ditemukan** adanya nilai yang hilang (*missing values*) pada setiap kolom, sehingga tidak diperlukan proses imputasi.
- **Distribusi dan Skala**: Ringkasan statistik menunjukkan bahwa rentang nilai dan skala antar fitur sangat bervariasi. Sebagai contoh, `total sulfur dioxide` memiliki nilai maksimum 289.0, sedangkan `chlorides` memiliki nilai maksimum 0.611.
- **Keseimbangan Kelas**: Setelah variabel `quality` ditransformasikan menjadi kategori biner, teridentifikasi adanya **ketidakseimbangan kelas (*class imbalance*)**. Terdapat **1175 sampel** untuk anggur berkualitas "Buruk" (kelas 0) dan hanya **184 sampel** untuk anggur berkualitas "Baik" (kelas 1). Kondisi ini menjadi pertimbangan krusial dalam proses pemodelan, di mana strategi seperti *stratified sampling* dan penyesuaian bobot kelas (*class weighting*) diterapkan untuk mencegah model menjadi bias terhadap kelas mayoritas.

**Link Notebook Pengerjaan:**
https://github.com/LatiefDataVisionary/data-science-application-college-task/blob/main/src/RandomForestClassifier_Week12.ipynb

# 1. Import Library

```
# Manipulasi dan Analisis Data
import pandas as pd
import numpy as np

# Visualisasi Data
import seaborn as sns
import matplotlib.pyplot as plt

# Pemuatan Dataset dari Kaggle
import kagglehub
from kagglehub import KaggleDatasetAdapter

# Pemodelan dan Evaluasi
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (accuracy_score, precision_score,
recall_score,
                                    classification_report,
ConfusionMatrixDisplay,
                                    confusion_matrix, roc_auc_score,

                                    roc_curve)

# Visualisasi Decision Tree
from sklearn.tree import export_graphviz
import graphviz
```

**Penjelasan Library yang digunaknan**

- **pandas (pd)**
  - Digunakan untuk manipulasi dan analisis data.
  - Fungsi dalam proyek:
    - Membaca dataset
    - Mengolah data dalam bentuk `DataFrame`
    - Membersihkan data (menghapus duplikat, memeriksa missing values)
    - Memanipulasi kolom data
- **numpy (np)**
  - Digunakan untuk operasi numerik.
  - Meskipun tidak eksplisit, menjadi dasar operasi di library lain seperti pandas dan scikit- learn.
- **seaborn (sns)**

- Library visualisasi data tingkat tinggi.
- Fungsi: Membuat `histplot`, `boxplot`, dan `heatmap` untuk eksplorasi data. [1]

- **matplotlib.pyplot (plt)**
  - Library visualisasi data dasar.
  - Fungsi:
    - Membuat dan mengatur plot
    - Menambahkan judul, label, grid
    - Menampilkan plot [1]

- **kagglehub**
  - Library untuk berinteraksi dengan Kaggle Hub.
  - Digunakan khusus untuk memuat dataset langsung dari Kaggle.

- **KaggleDatasetAdapter**
  - Adapter dari `kagglehub` untuk memuat dataset sebagai `DataFrame` pandas.

- **sklearn.model_selection.train_test_split**
  - Membagi dataset menjadi set **pelatihan** dan **pengujian**.

- **sklearn.model_selection.GridSearchCV**
  - Melakukan **hyperparameter tuning** dengan cross-validation.

- **sklearn.ensemble.RandomForestClassifier**
  - Model ML berbasis ensemble (gabungan decision tree).
  - Digunakan untuk klasifikasi kualitas anggur.

- **sklearn.metrics**
  - Modul metrik evaluasi performa model klasifikasi:
    - `accuracy_score`, `precision_score`, `recall_score`
    - `classification_report` (laporan presisi, recall, f1-score per kelas)
    - `confusion_matrix` + `ConfusionMatrixDisplay` (visualisasi)
    - `roc_auc_score` (AUC-ROC)
    - `roc_curve` (menghitung FPR/TPR untuk plot ROC)

- **sklearn.tree.export_graphviz**
  - Menghasilkan representasi DOT dari decision tree.

- **graphviz**
  - Merender output `export_graphviz` menjadi gambar decision tree.

# 2. Load data dan Exploratory Data Analysis (EDA)

Pada tahap ini, kita akan memuat dataset dan melakukan beberapa langkah eksplorasi data awal untuk memahami karakteristik data yang kita miliki.

## 2.1. Memuat Dataset

Dataset diunduh langsung dari Kaggle Hub menggunakan pustaka kagglehub.

```
df = kagglehub.load_dataset(
  KaggleDatasetAdapter.PANDAS,
  "uciml/red-wine-quality-cortez-et-al-2009"
  , "winequality-red.csv"
)

display(df.head())
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | su |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | |

## 2.2. Pemeriksaan Awal Data

Kita akan melihat informasi dasar dari dataset seperti tipe data setiap kolom, jumlah entri, dan ringkasan statistiknya.

```
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
```

Metode `df.info()` memberikan kita ringkasan singkat dan penting mengenai DataFrame (tabel data) yang baru saja kita muat. Ini adalah langkah pertama yang krusial dalam setiap proyek analisis data.

Dari output di atas, kita dapat menyimpulkan beberapa hal:

- **Struktur Data**:
  - Data kita memiliki **1599 baris** (*entries*) dan **12 kolom**. Ini memberikan gambaran awal tentang ukuran dataset.
- **Kolom dan Tipe Data**:
  - **`Non-Null Count`**: Semua kolom menunjukkan `1599 non-null`. Ini adalah kabar baik! Artinya, **tidak ada data yang hilang (*missing values*)** dalam dataset
    kita. Dengan demikian, kita tidak perlu melakukan penanganan data kosong seperti imputasi.
  - **`Dtype`**: Menunjukkan tipe data setiap kolom.
    - Terdapat **11 kolom** dengan tipe `float64` (angka desimal), yang cocok untuk atribut fisiko-kimia seperti `fixed acidity`, `alcohol`, dll.
    - Ada **1 kolom** dengan tipe `int64` (bilangan bulat), yaitu kolom target kita, `quality`.
- **Penggunaan Memori**:
  - DataFrame ini menggunakan sekitar **150.0 KB** memori.

Secara keseluruhan, data ini terlihat "bersih" dari nilai yang hilang dan siap untuk tahap eksplorasi lebih lanjut.

```
display(df.describe())
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free su diox |
|---|---|---|---|---|---|---|
| **count** | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000 |
| **mean** | 8.319637 | 0.527821 | 0.270976 | 2.538806 | 0.087467 | 15.874 |
| **std** | 1.741096 | 0.179060 | 0.194801 | 1.409928 | 0.047065 | 10.460 |
| **min** | 4.600000 | 0.120000 | 0.000000 | 0.900000 | 0.012000 | 1.000 |
| **25%** | 7.100000 | 0.390000 | 0.090000 | 1.900000 | 0.070000 | 7.000 |
| **50%** | 7.900000 | 0.520000 | 0.260000 | 2.200000 | 0.079000 | 14.000 |
| **75%** | 9.200000 | 0.640000 | 0.420000 | 2.600000 | 0.090000 | 21.000 |
| **max** | 15.900000 | 1.580000 | 1.000000 | 15.500000 | 0.611000 | 72.000 |

Metode `df.describe()` memberikan ringkasan statistik yang sangat berguna untuk kolom-kolom numerik. Dari sini, kita bisa mendapatkan pemahaman awal tentang pusat data, sebaran, dan potensi adanya nilai-nilai aneh (*outlier*).

Berikut adalah penjabaran dari setiap metrik yang ditampilkan:

- **`count`**: Jumlah baris data yang tidak kosong. Angka `1599` di semua kolom mengonfirmasi bahwa tidak ada nilai yang hilang.
- **`mean`**: Nilai rata-rata dari setiap kolom.
- **`std`** (Standar Deviasi): Mengukur seberapa tersebar data dari nilai rata-ratanya. Nilai `std` yang besar, seperti pada `total sulfur dioxide` (32.89), menunjukkan sebaran data yang luas.
- **`min`**: Nilai terkecil dalam kolom.
- **`25%`**: Kuartil pertama (Q1). 25% dari data memiliki nilai di bawah angka ini.
- **`50%`**: Median atau kuartil kedua (Q2). Ini adalah nilai tengah dari data, yang seringkali lebih representatif daripada `mean` jika ada outlier.
- **`75%`**: Kuartil ketiga (Q3). 75% dari data memiliki nilai di bawah angka ini.
- **`max`**: Nilai terbesar dalam kolom.

Pengamatan Kunci dari Statistik:

1. **Potensi Outlier**
   - Terlihat perbedaan yang sangat signifikan antara nilai `75%` dan `max` pada beberapa fitur. Contohnya, pada **`total sulfur dioxide`**, nilai kuartil ke-3 adalah 62, namun nilai maksimumnya mencapai 289. Hal yang sama terlihat pada `residual sugar` dan `chlorides`. Ini adalah indikasi kuat adanya *outlier* yang perlu kita perhatikan.
2. **Distribusi Data**
   - Untuk beberapa kolom seperti `residual sugar`, nilai `mean` (2.53) lebih tinggi dari `median/50%` (2.2). Ini mengindikasikan distribusi data yang miring ke kanan (*right- skewed*), yang berarti ada beberapa nilai yang sangat tinggi yang "menarik" rata-ratanya.
3. **Variabel Target `quality`**
   - Kolom `quality` memiliki nilai rata-rata (`mean`) 5.63. Skornya berkisar dari 3 (`min`) hingga 8 (`max`), menunjukkan tidak ada anggur dengan kualitas sangat buruk atau sangat baik dalam dataset ini. Sebagian besar data terkumpul di tengah.
4. **Skala Fitur**
   - Nilai pada kolom `total sulfur dioxide` jauh lebih besar dibandingkan `chlorides`. Ini menunjukkan bahwa penskalaan fitur (*feature scaling*) mungkin akan diperlukan sebelum melatih beberapa jenis model machine learning untuk memastikan setiap fitur memberikan kontribusi yang seimbang.

## 2.3. Data Cleaning: Duplikat dan Missing Value

```
# Memeriksa data duplikat
print(f'Jumlah data duplikat : {df.duplicated().sum()}')
print(f'Jumlah baris sebelum menghapus duplikat: {len(df)}')

# Menghapus data dupliksa
df.drop_duplicates(inplace=True)

# Memverifikasi setelah penanganan
print(f'Jumlah baris setelah menghapus duplikat : {len(df)}')
```

```
Jumlah data duplikat : 240
Jumlah baris sebelum menghapus duplikat: 1599
Jumlah baris setelah menghapus duplikat : 1359
```

```
# Memeriksa nilai yang hilang (missing values)
df.isna().sum()
```

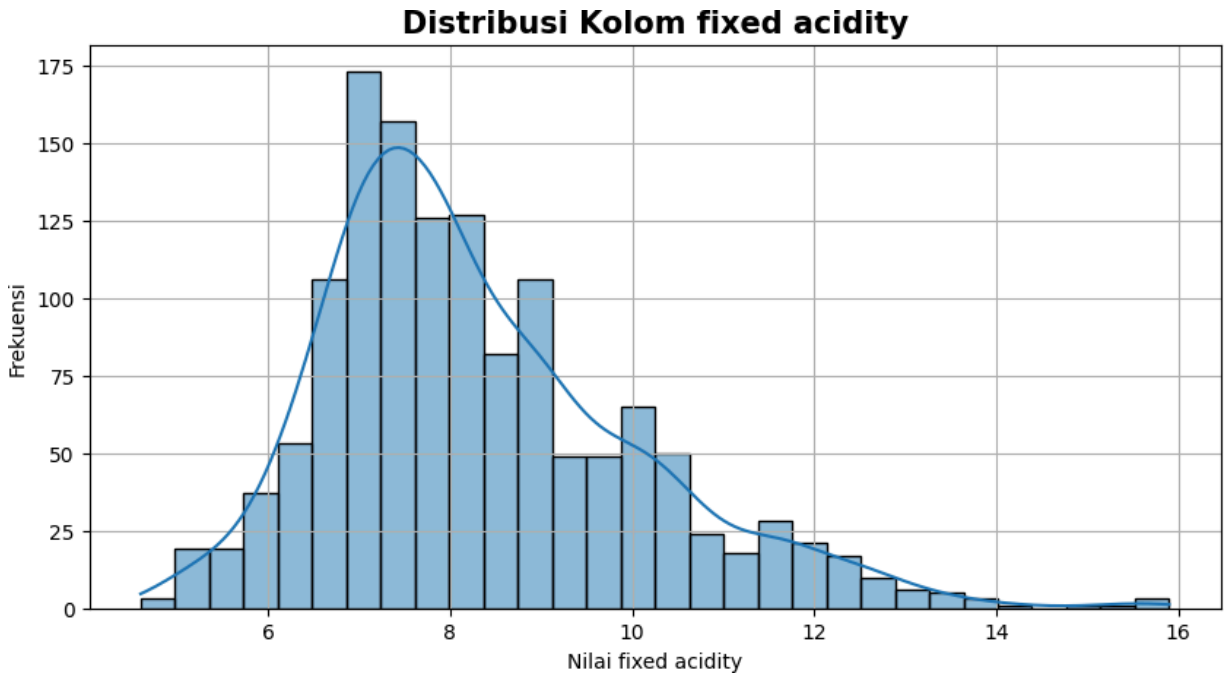
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur            0
dioxide density         0
```

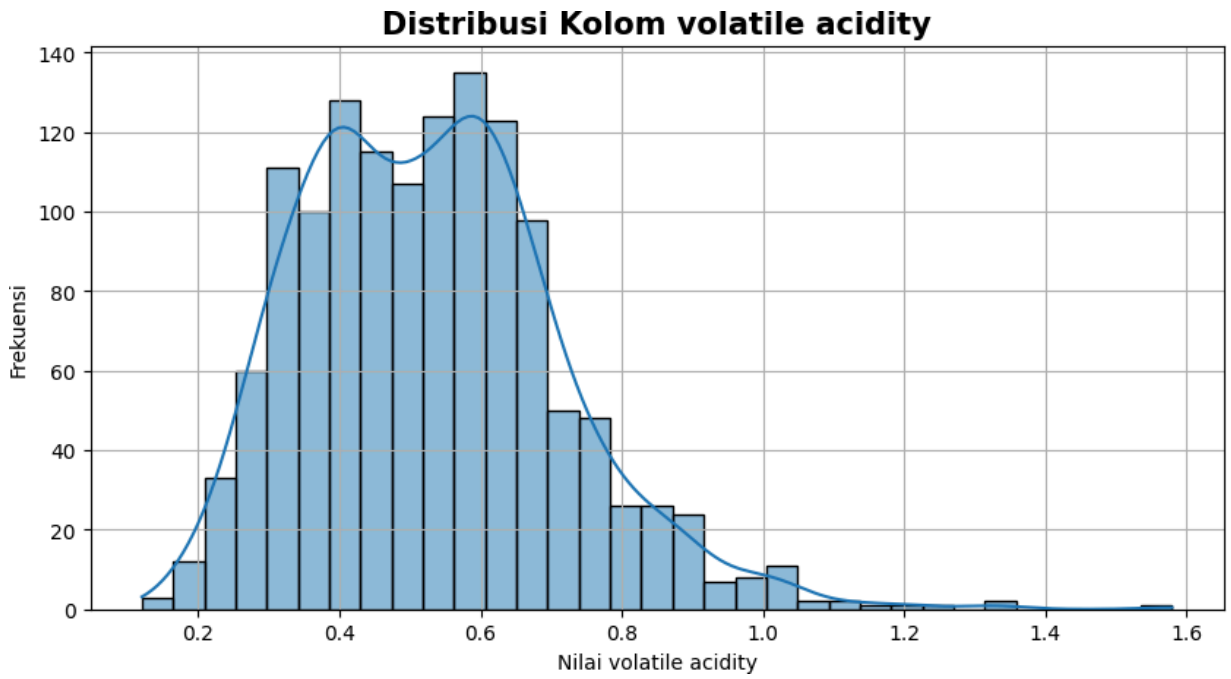
```
pH                    0
sulphate              0
s                     0
                      0
dtype: int64
quality
```

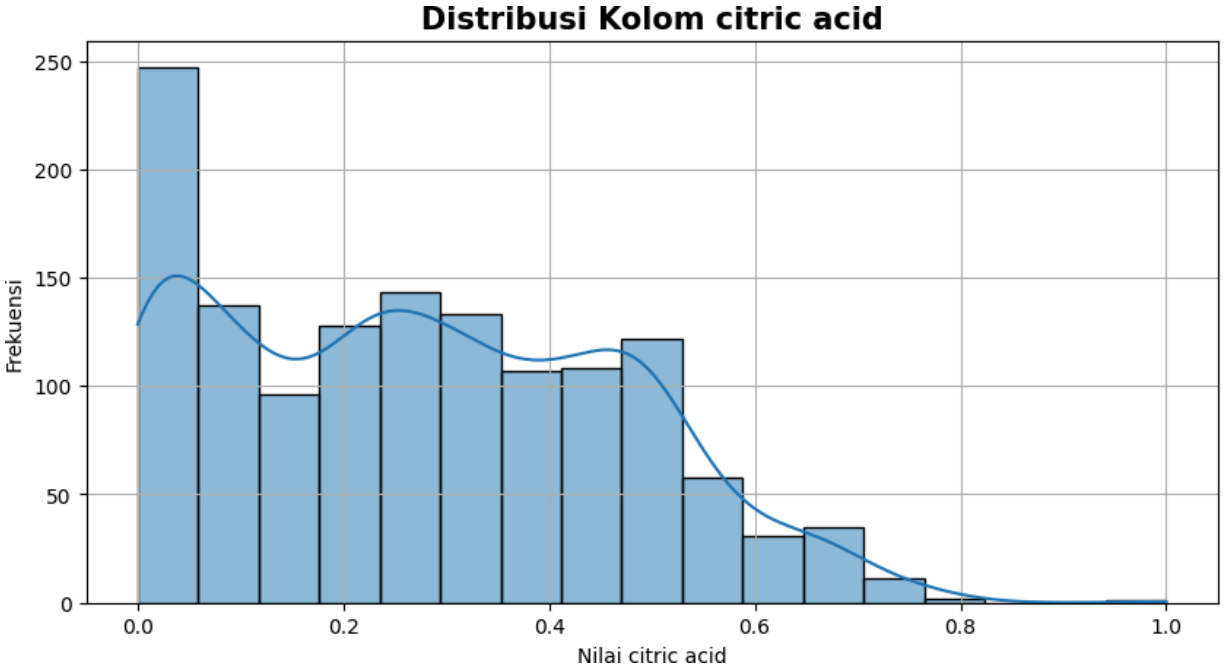
Hasil di atas menunjukkan bahwa data sudah bersih dari nilai yang hilang.

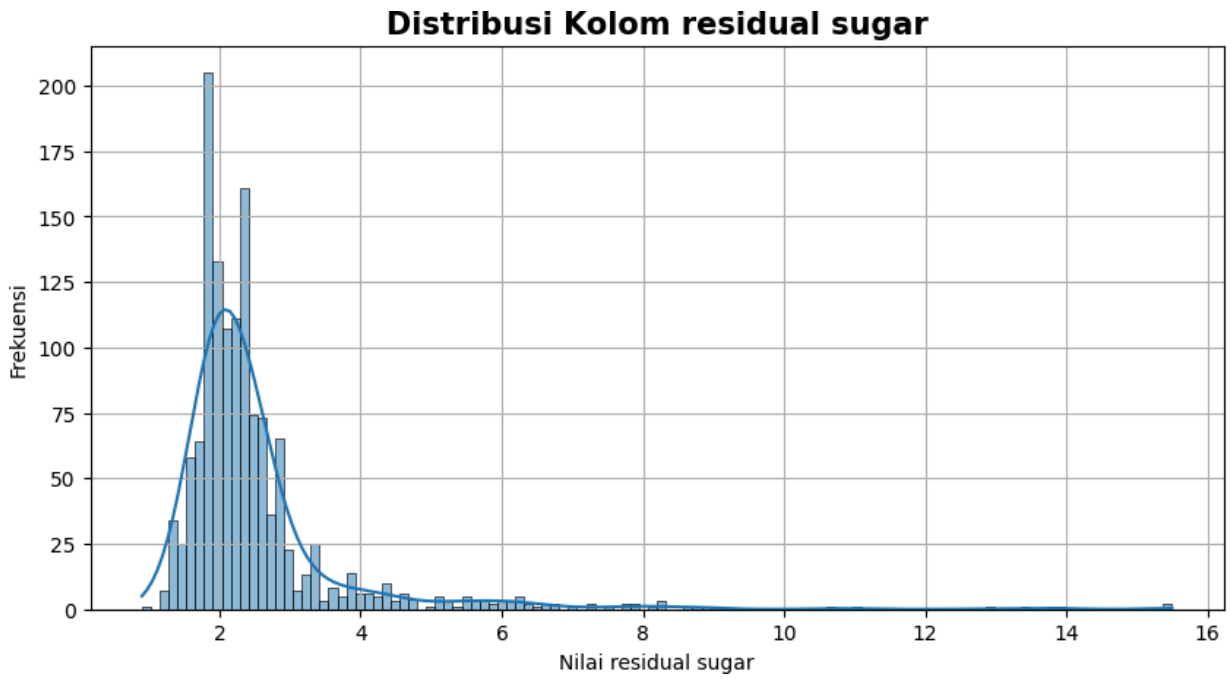
## 2.4. Visualisasi Distribusi Fitur

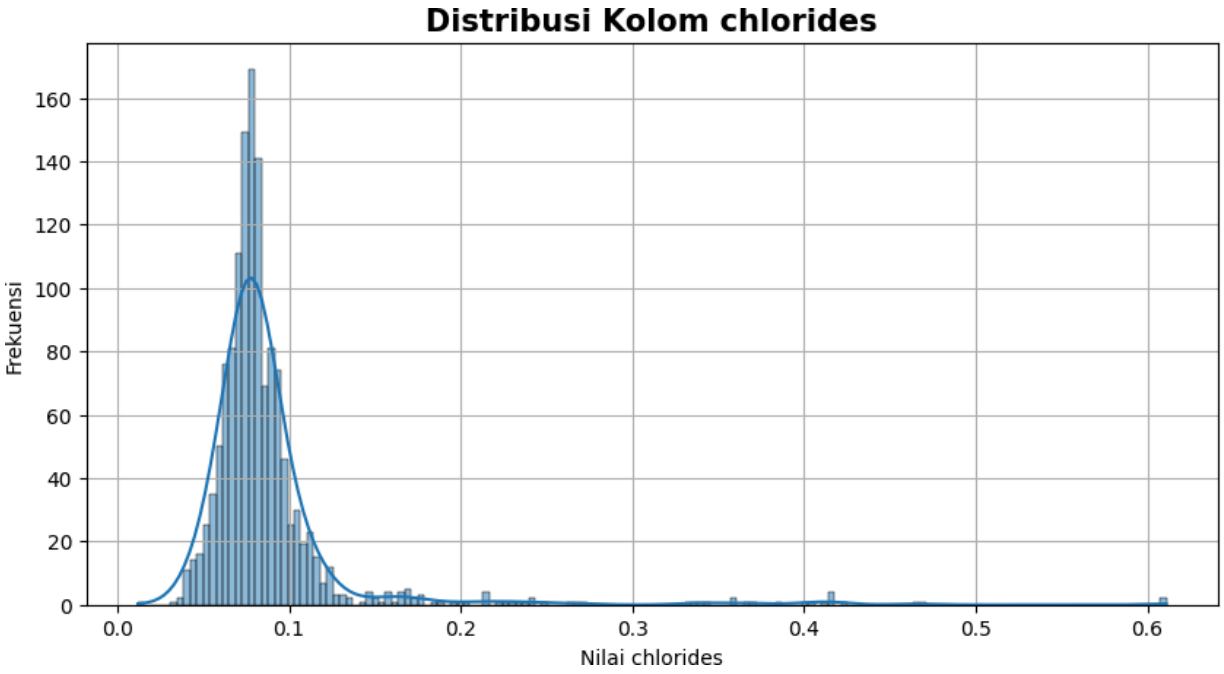
```
print("\nVisualisasi Distribusi Setiap Fitur:")
 for column in df.columns:
plt.figure(figsize=(10, 5))
   sns.histplot(data=df, x=column, kde=True)
   plt.title(f'Distribusi Kolom {column}', fontweight='bold',
 fontsize=15)
plt.xlabel(f'Nilai {column}')
   plt.ylabel(f'Frekuensi')
   plt.grid(True)


Visualisasi Distribusi Setiap Fitur:
```

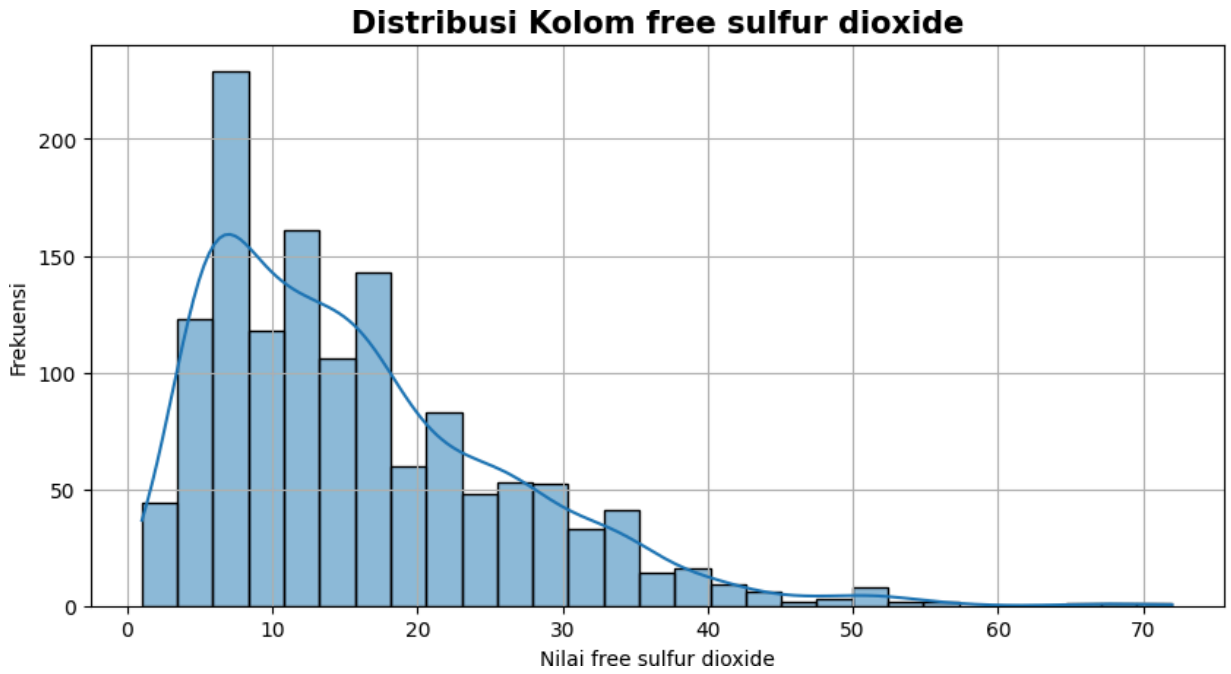


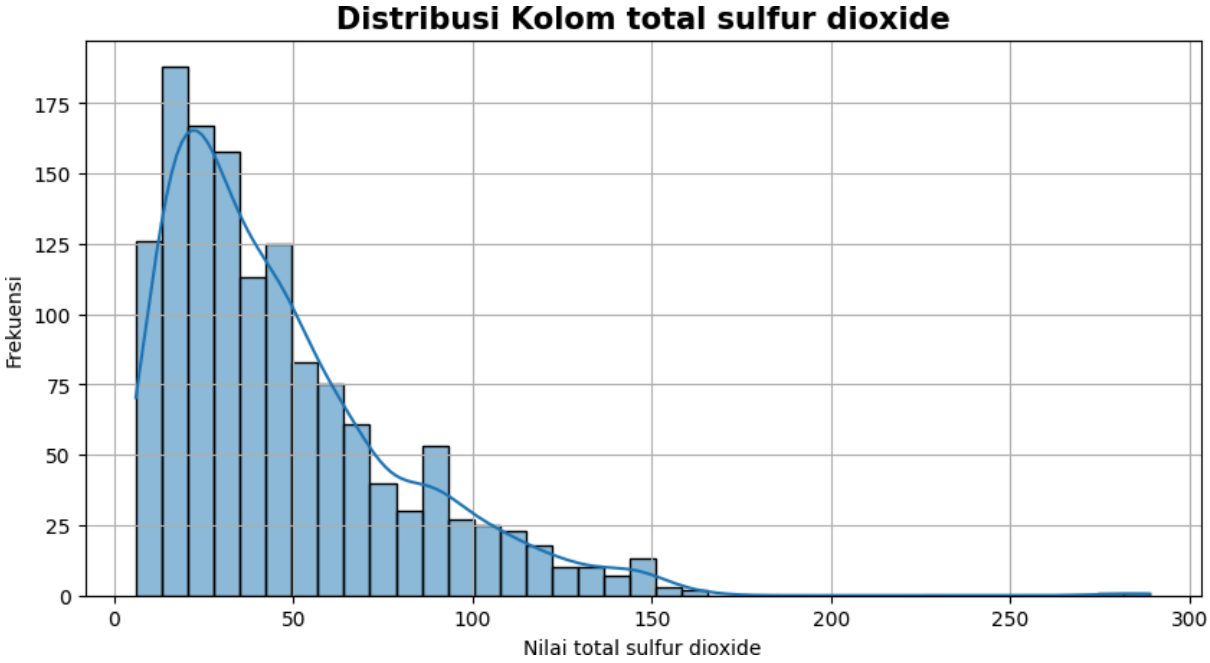

Distribusi Kolom volatile acidity
Frekuensi
Nilai volatile acidity
0
20
40
60
80
100
120
140
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

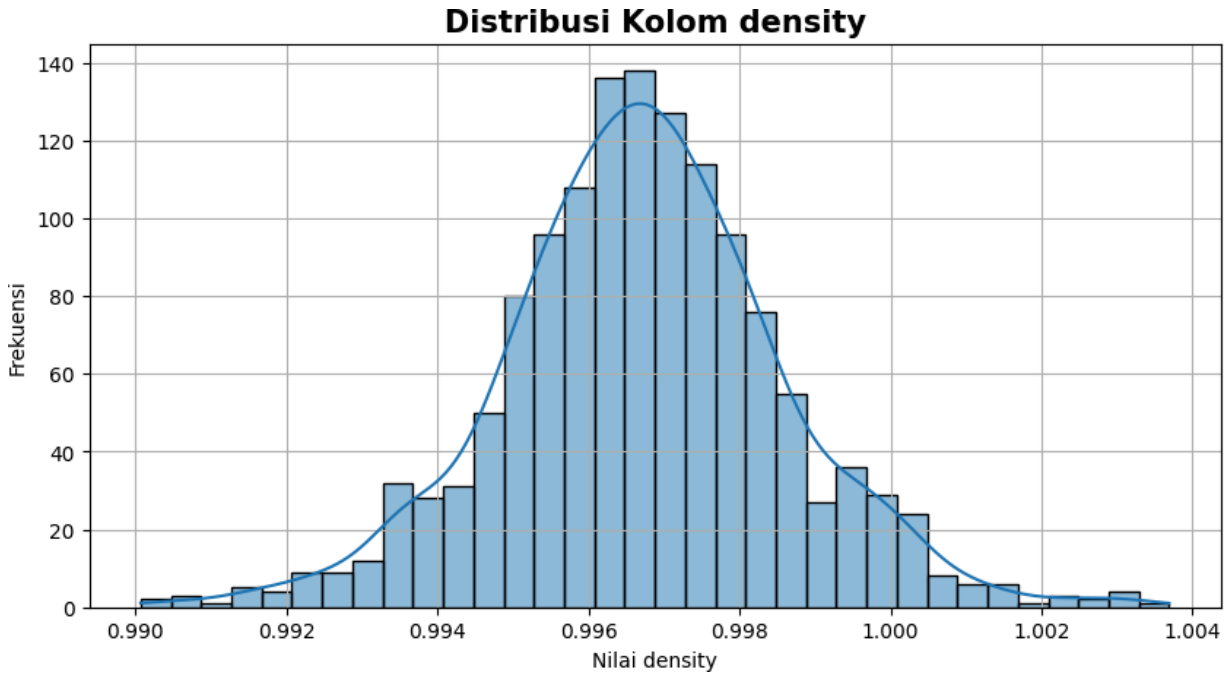

Distribusi Kolom citric acid
Frekuensi
Nilai citric acid
0
50
100
150
200
250
0.0
0.2
0.4
0.6
0.8
1.0

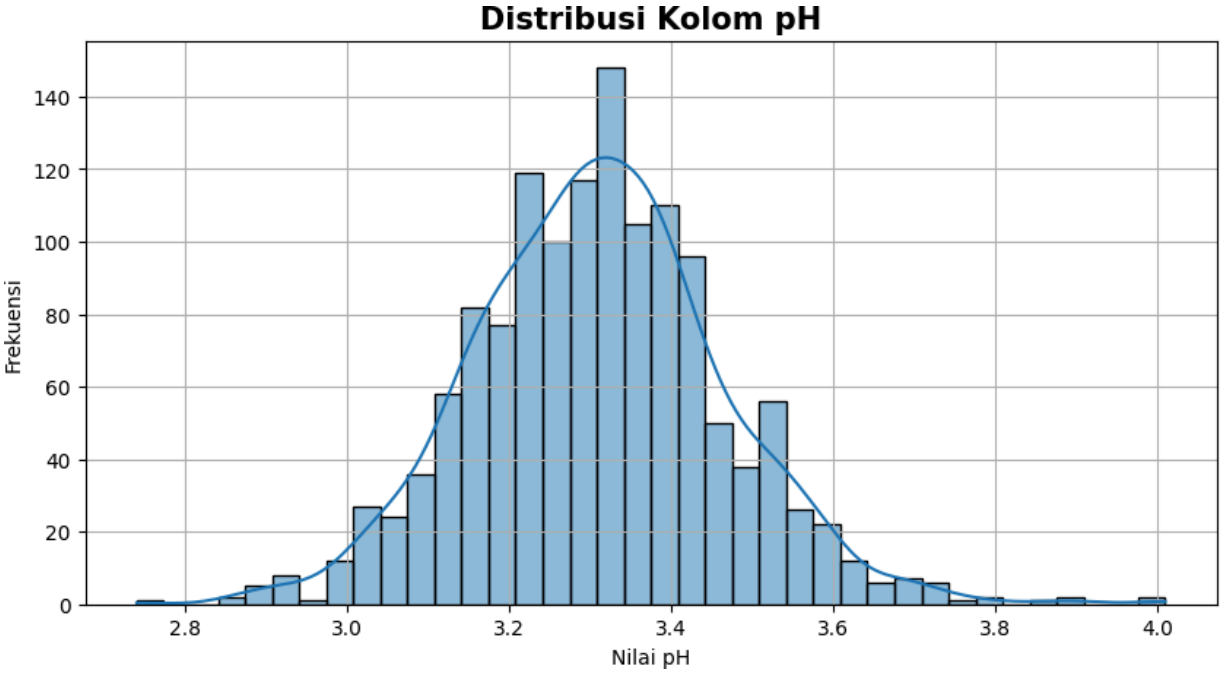

Distribusi Kolom residual sugar
Frekuensi
Nilai residual sugar
0
25
50
75
100
125
150
175
200
2
4
6
8
10
12
14
16

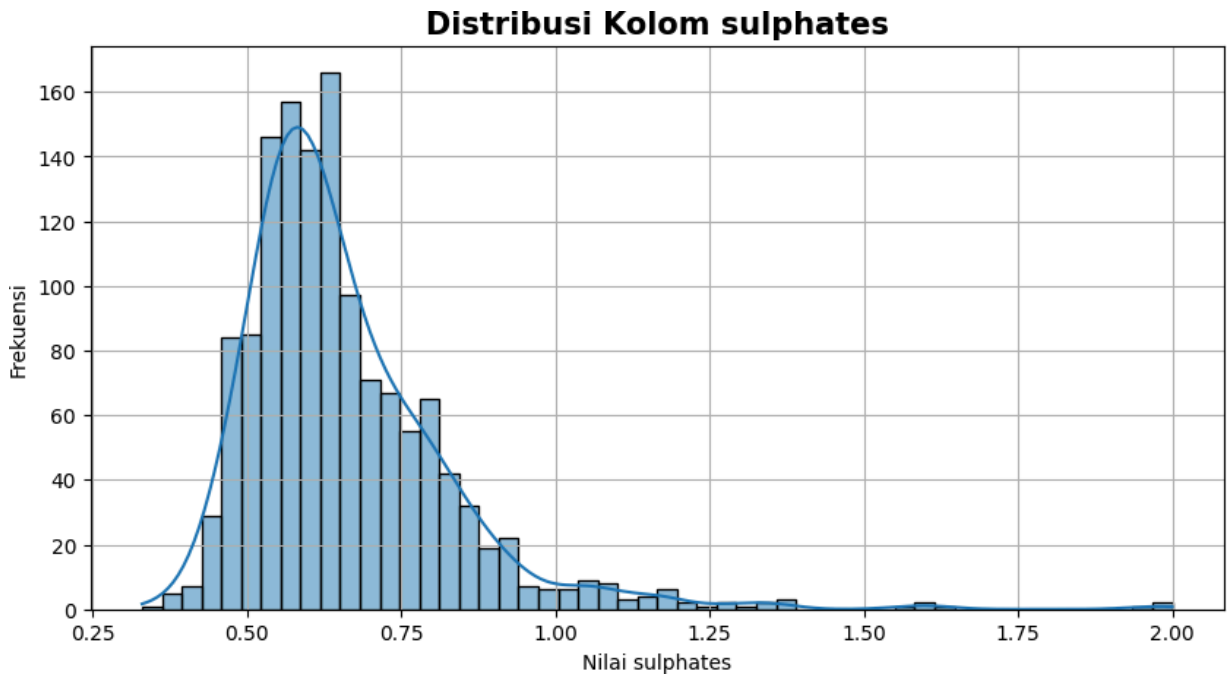

Distribusi Kolom chlorides
Frekuensi
Nilai chlorides
0
20
40
60
80
100
120
140
160
0.0
0.1
0.2
0.3
0.4
0.5
0.6

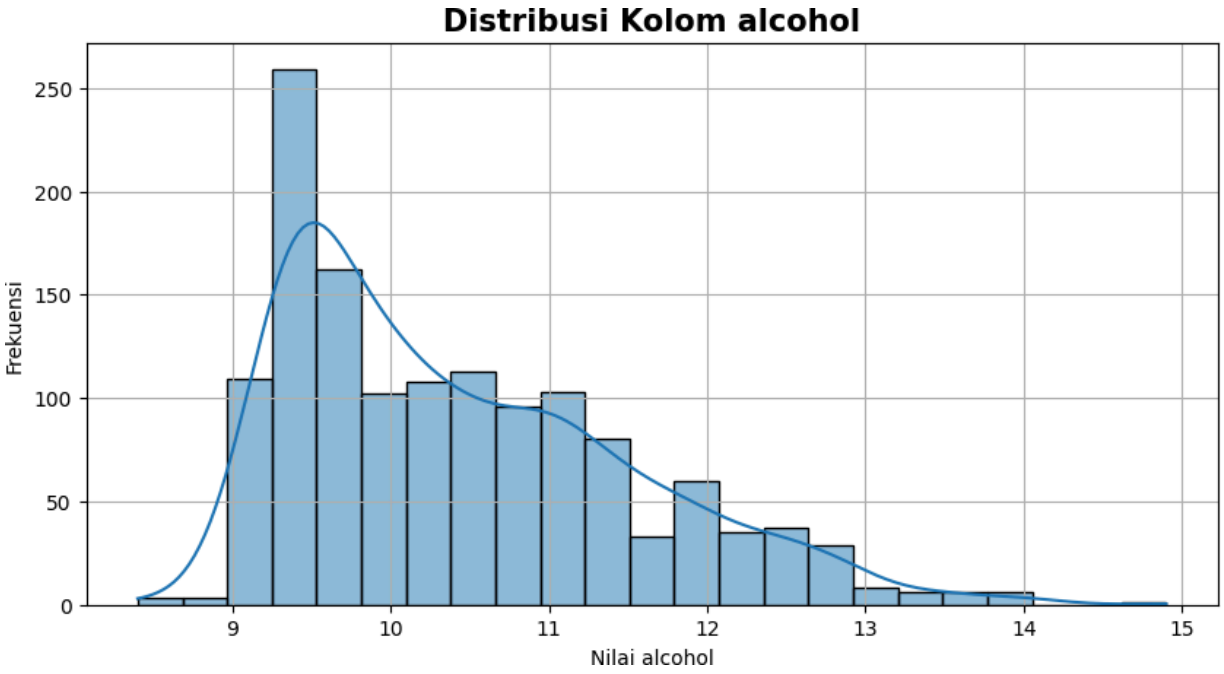

Distribusi Kolom free sulfur dioxide
Frekuensi
Nilai free sulfur dioxide
0
50
100
150
200
0
10
20
30
40
50
60
70

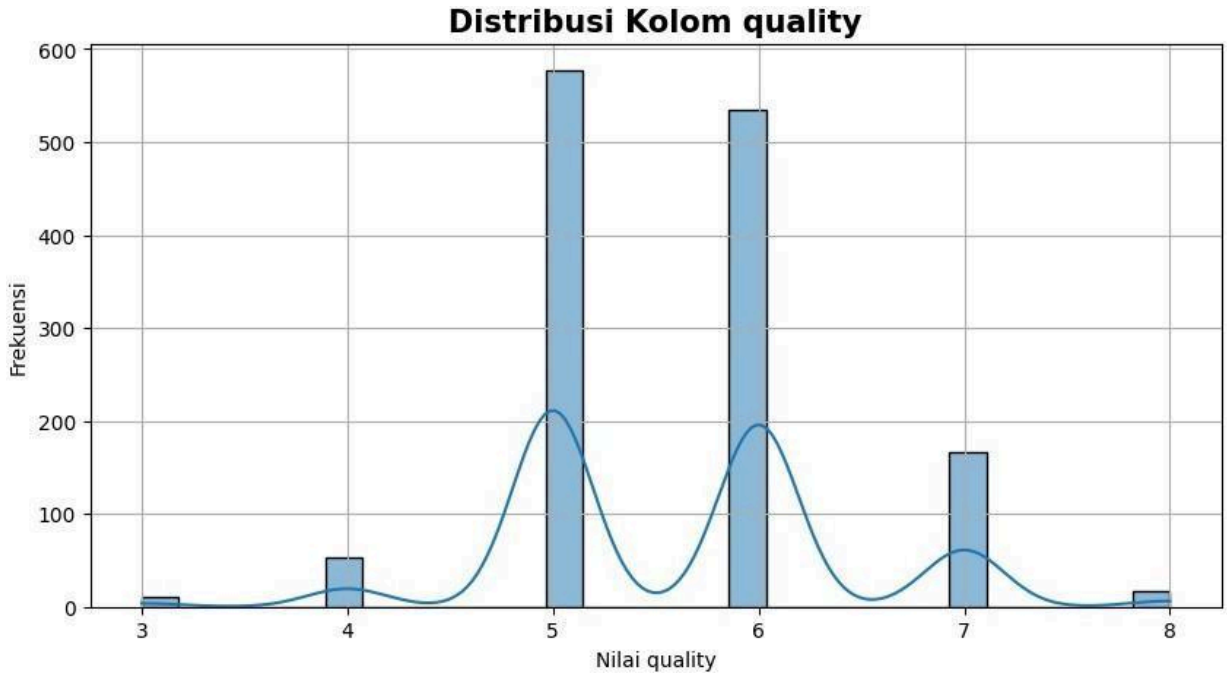

Distribusi Kolom total sulfur dioxide
Frekuensi
Nilai total sulfur dioxide
0
25
50
75
100
125
150
175
0
50
100
150
200
250
300


Distribusi Kolom density
Frekuensi
Nilai density
0
20
40
60
80
100
120
140
0.990
0.992
0.994
0.996
0.998
1.000
1.002
1.004


Distribusi Kolom pH
Frekuensi
Nilai pH
0
20
40
60
80
100
120
140
2.8
3.0
3.2
3.4
3.6
3.8
4.0


Distribusi Kolom sulphates
Frekuensi
Nilai sulphates
0
20
40
60
80
100
120
140
160
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Distribusi Kolom alcohol
Frekuensi
Nilai alcohol
0
50
100
150
200
250
9
10
11
12
13
14
15




Langkah ini penting untuk memahami karakteristik sebaran data pada setiap kolom secara visual. Dengan menggunakan **histogram** yang dilengkapi dengan kurva **Kernel Density Estimate (KDE)**, kita dapat melihat bentuk distribusi dan mengidentifikasi potensi adanya *outlier* atau kemiringan data (*skewness*).

Hasil Visualisasi & Pengamatan Kunci:

- **Distribusi Normal**:
  - Fitur `density`, `pH`, dan `fixed acidity` menunjukkan distribusi yang mendekati kurva normal (lonceng), meskipun tidak sempurna. Ini berarti sebagian besar nilai terpusat di sekitar rata-rata.
- **Distribusi Miring ke Kanan (*Right-Skewed*)**:
  - Banyak fitur yang distribusinya miring ke kanan. Ini terlihat dari "ekor" panjang di sisi kanan grafik.
  - Fitur-fitur seperti `residual sugar`, `chlorides`, `free sulfur dioxide`, dan `total sulfur dioxide` menunjukkan kemiringan yang sangat jelas. Ini mengindikasikan bahwa sebagian besar anggur memiliki nilai rendah untuk atribut ini, namun ada beberapa sampel dengan nilai yang sangat tinggi (outlier).
  - `alcohol` dan `sulphates` juga menunjukkan kemiringan ke kanan, menandakan mayoritas anggur memiliki kandungan alkohol dan sulfat yang moderat, dengan beberapa pengecualian yang nilainya lebih tinggi.
- **Distribusi Bimodal**:
  - `volatile acidity` menunjukkan dua puncak (bimodal), yang bisa berarti ada dua kelompok anggur yang berbeda dalam dataset berdasarkan keasaman volatilnya.
- **Distribusi Target `quality`**:
  - Grafik untuk kolom `quality` sangat jelas menunjukkan bahwa sebagian besar anggur dinilai dengan skor **5 dan 6**.

- Sangat sedikit anggur yang memiliki skor kualitas rendah (3) atau sangat tinggi (8). Ini adalah konfirmasi visual dari **ketidakseimbangan data**, di mana model mungkin akan lebih mudah mempelajari karakteristik anggur berkualitas rata- rata dibandingkan anggur yang sangat baik atau sangat buruk.

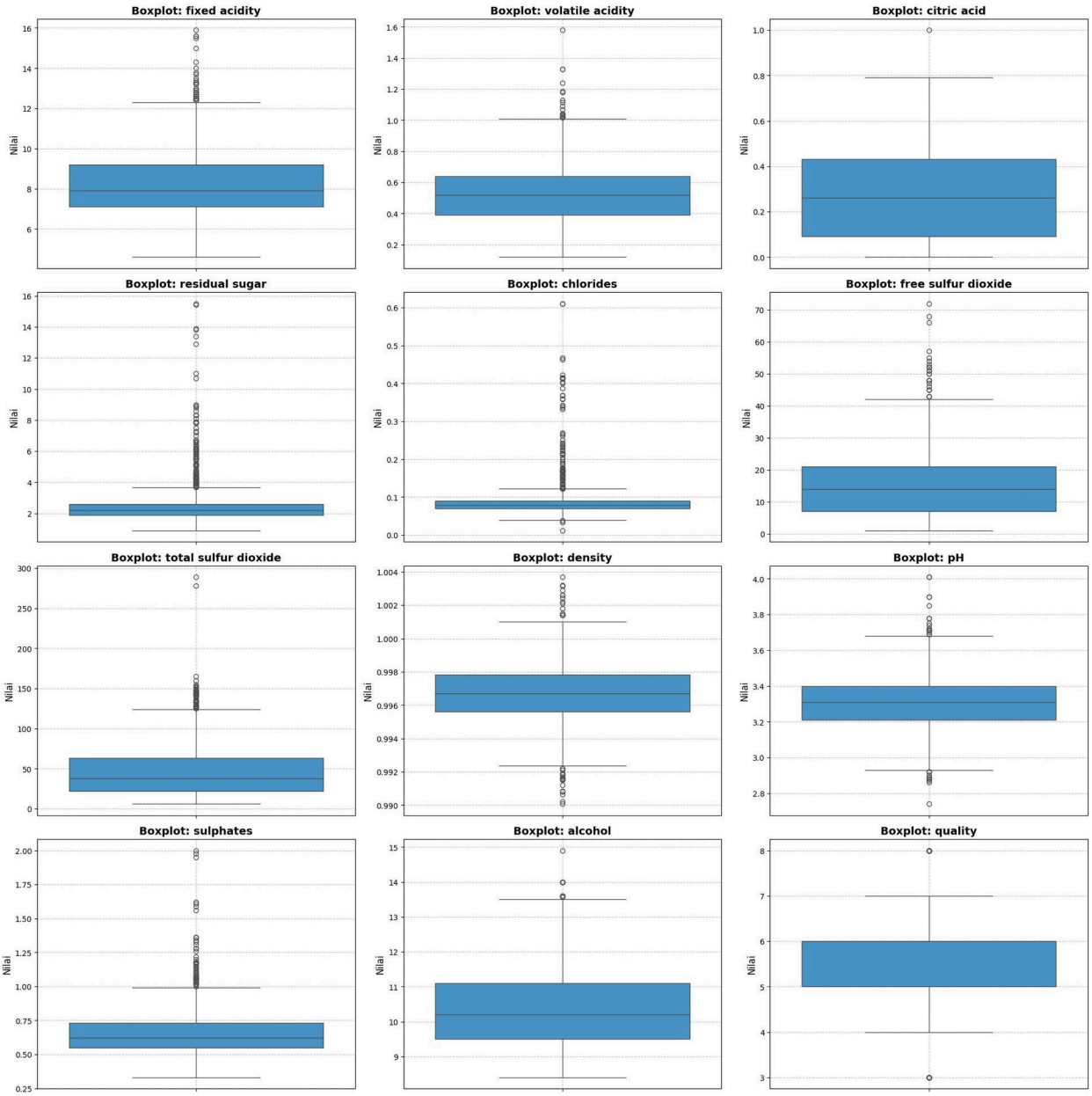
Secara umum, visualisasi ini memperkuat temuan dari analisis statistik sebelumnya dan memberikan wawasan penting tentang perlunya penskalaan fitur (karena rentang nilai yang berbeda) dan penanganan outlier/data miring untuk beberapa model machine learning.

## 2.5. Identifikasi Outliers

Boxplot digunakan untuk melihat sebaran data dan mengidentifikasi adanya outliers (pencilan) pada setiap fitur.

```
# Visualisasi Boxplot untuk setiap fitur secara individual
print("\nBoxplot Individual untuk Setiap Fitur:")
columns = df.columns
num_columns = len(columns)
num_rows = (num_columns + 2) // 3 # Atur 3 kolom per baris

plt.figure(figsize=(20, 5 * num_rows)) # Sesuaikan ukuran figure
berdasarkan jumlah baris

for i, column in enumerate(columns):
    plt.subplot(num_rows, 3, i + 1)
    sns.boxplot(y=df[column], color='#3498db') # Menggunakan seaborn
untuk tampilan yang lebih baik dan konsisten
plt.title(f'Boxplot: {column}', fontsize=14, fontweight='bold')
    plt.ylabel('Nilai', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout() # Menyesuaikan layout agar tidak tumpang tindih
plt.show()
```

```
Boxplot Individual untuk Setiap Fitur:
```




**Boxplot** adalah alat visual yang sangat efektif untuk memahami sebaran data dan mengidentifikasi keberadaan **outliers** (pencilan) pada setiap fitur.

Setiap "kotak" dalam grafik ini merepresentasikan *Interquartile Range (IQR)*, yaitu rentang antara kuartil pertama (Q1 atau 25%) dan kuartil ketiga (Q3 atau 75%). Garis di tengah kotak adalah median (50%). Garis vertikal (disebut *whiskers*) memanjang dari kotak untuk menunjukkan rentang data, biasanya hingga 1.5 kali IQR. **Titik-titik di luar *whiskers*** adalah outlier.

Hasil Visualisasi & Pengamatan Kunci:

- **Banyaknya Outlier**: Hampir semua fitur fisikokimia, kecuali `citric acid` dan `pH`, menunjukkan adanya outlier. Outlier paling ekstrem terlihat pada fitur `total sulfur dioxide`, `free sulfur dioxide`, dan `residual sugar`.

- **Konfirmasi Kemiringan Data**: Adanya banyak outlier di sisi atas (*upper outliers*) pada fitur seperti `residual sugar`, `chlorides`, dan `sulphates` mengonfirmasi kembali bahwa distribusi data pada fitur-fitur ini sangat miring ke kanan (*right- skewed*).
- **Fitur dengan Sebaran Normal**: Fitur `pH` dan `density` menunjukkan distribusi yang paling simetris (mendekati normal) dengan sedikit atau tanpa outlier yang signifikan. Ini menandakan bahwa sebagian besar nilai untuk fitur ini terpusat di tengah dengan sebaran yang wajar.
- **Variabel Target `quality`**: Kolom `quality` juga memiliki outlier, yaitu pada nilai 3 dan 8, yang menegaskan bahwa anggur dengan kualitas sangat rendah atau sangat tinggi adalah kasus yang jarang terjadi dalam dataset ini.

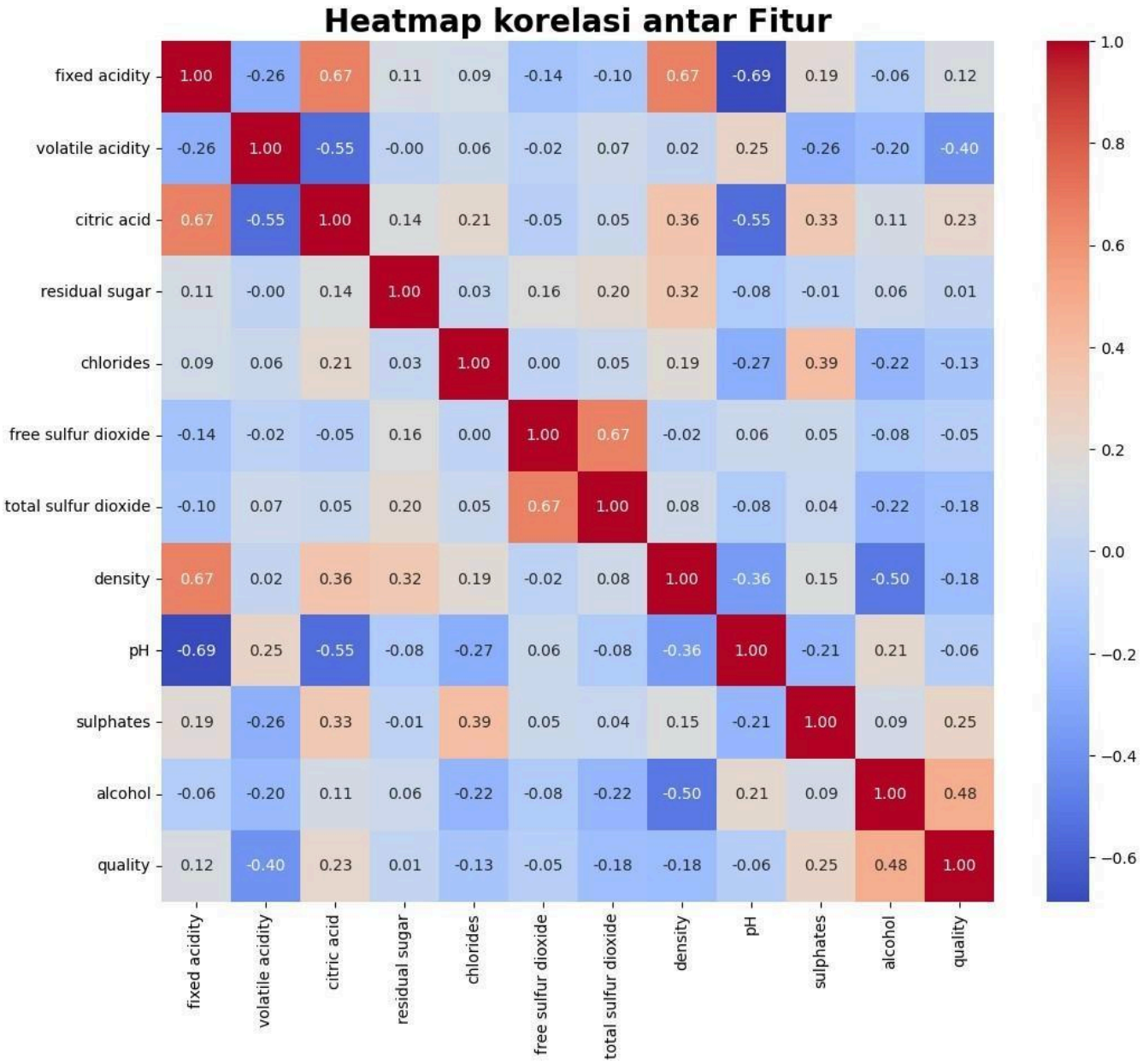
Tindak Lanjut:

Kehadiran outlier ini penting untuk diketahui. Meskipun Random Forest secara umum cukup tahan (*robust*) terhadap outlier, keberadaannya bisa memengaruhi metrik statistik seperti rata-rata dan standar deviasi. Untuk model lain yang lebih sensitif, penanganan outlier mungkin diperlukan. Namun, untuk kasus ini, kita akan melanjutkan tanpa menghapus outlier terlebih dahulu, mengingat ketahanan alami dari model Random Forest.

## 2.6. Analisis Korelasi

Heatmap korelasi menunjukkan hubungan linear antar variabel. Korelasi yang tinggi antara fitur-fitur independen (multikolinearitas) dapat memengaruhi interpretasi model.

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm',
 fmt='.2f')
plt.title('Heatmap korelasi antar Fitur', fontsize=20,
 fontweight='bold')
plt.show()
```




**Heatmap korelasi** adalah visualisasi yang sangat berguna untuk melihat **hubungan linear antar variabel** dalam dataset. Setiap sel dalam heatmap menunjukkan koefisien korelasi antara dua variabel.

- **Skala Warna**:
  - Warna **merah tua** menunjukkan korelasi positif yang kuat (mendekati +1). Artinya, jika satu variabel meningkat, variabel lainnya cenderung meningkat juga.
  - Warna **biru tua** menunjukkan korelasi negatif yang kuat (mendekati -1). Artinya, jika satu variabel meningkat, variabel lainnya cenderung menurun.
  - Warna **netral** (sekitar putih atau abu-abu muda) menunjukkan korelasi yang lemah atau tidak ada korelasi linear (mendekati 0).
- **Nilai dalam Sel**: Angka di setiap sel adalah nilai koefisien korelasi Pearson, yang berkisar antara -1 hingga +1. Format `fmt='.2f'` memastikan nilai ditampilkan dengan

dua angka desimal.

Hasil Visualisasi & Pengamatan Kunci:

- **Korelasi dengan Target (`quality`)**:
  - **`alcohol`** memiliki korelasi positif tertinggi dengan `quality` **(0.48)**. Ini mengindikasikan bahwa anggur dengan kandungan alkohol yang lebih tinggi cenderung memiliki skor kualitas yang lebih baik.
  - **`volatile acidity`** menunjukkan korelasi negatif tertinggi dengan `quality` **(- 0.40)**. Ini berarti semakin tinggi keasaman volatil, kualitas anggur cenderung semakin rendah.
  - **`sulphates`** juga menunjukkan korelasi positif yang cukup baik dengan `quality` **(0.25)**.
  - **`citric acid`** memiliki korelasi positif moderat dengan `quality` **(0.23)**.
  - Fitur lain seperti `residual sugar`, `free sulfur dioxide`, dan `pH` menunjukkan korelasi yang sangat lemah dengan `quality`.
- **Korelasi Antar Fitur (Potensi Multikolinearitas)**:
  - Terdapat korelasi positif yang cukup kuat antara **`fixed acidity`** dan **`citric acid`** **(0.67)**. Ini masuk akal karena asam sitrat adalah salah satu komponen dari keasaman tetap.
  - **`fixed acidity`** juga berkorelasi positif dengan **`density`** **(0.67)** dan negatif dengan **`pH`** **(-0.69)**. Ini juga logis karena peningkatan asam akan meningkatkan densitas dan menurunkan pH.
  - **`free sulfur dioxide`** dan **`total sulfur dioxide`** memiliki korelasi positif yang cukup tinggi **(0.67)**, yang memang diharapkan.

Implikasi untuk Pemodelan:

- Fitur-fitur dengan korelasi yang lebih tinggi terhadap `quality` (baik positif maupun negatif) kemungkinan besar akan menjadi prediktor yang lebih penting dalam model kita.
- Adanya korelasi yang cukup tinggi antar beberapa fitur input (seperti `fixed acidity` dan `citric acid`) menunjukkan potensi adanya **multikolinearitas**. Meskipun Random Forest relatif tahan terhadap multikolinearitas dibandingkan model regresi linear, ini adalah sesuatu yang perlu diingat, terutama jika kita ingin menginterpretasikan kontribusi masing-masing fitur secara individual.

Analisis korelasi ini memberikan panduan awal yang baik tentang fitur mana yang mungkin paling berpengaruh dan bagaimana fitur-fitur tersebut saling terkait.

# 3. Feature Engineering dan Data Splitting

Pada bagian ini, kita akan mempersiapkan data untuk pemodelan.

## 3.1. Transformasi Variabel Target

Sesuai dengan deskripsi dataset, kita akan mengubah masalah ini dari regresi menjadi klasifikasi. Kolom target quality akan diubah menjadi variabel biner:

- `1` **(Baik)**: Jika quality >= 7
- `0` **(Buruk)**: Jika quality < 7

```
 # Mengubah kolom 'quality' menjadi kategori biner (0 = bad, 1 = good)
df['quality'] = df['quality'].apply(lambda x: 1 if x >= 7 else 0)

print("Hasil transformasi kolom 'quality':")
 df['quality'].head(10)
```

```
Hasil transformasi kolom 'quality':

0     0
1     0
2     0
3     0
5     0
6     0
7     1
8     1
9     0
10    0
Name: quality, dtype: int64
```

## 3.2. Pemeriksaan Keseimbangan Kelas (Class Balance)

Setelah kolom target `quality` ditransformasi menjadi kategori biner (0 untuk "Buruk" dan 1 untuk "Baik"), langkah krusial selanjutnya adalah memeriksa **distribusi atau keseimbangan kelas**. Ketidakseimbangan kelas terjadi ketika jumlah sampel pada satu kelas jauh lebih banyak
daripada kelas lainnya. Hal ini dapat menyebabkan model machine learning menjadi bias dan lebih cenderung memprediksi kelas mayoritas, sehingga performanya pada kelas minoritas menjadi buruk.

## 3.3. Memeriksa distribusi kelas setelah transformasi

Kode di bawah ini menghitung jumlah kemunculan dan persentase dari setiap kelas pada variabel target `quality`.

```
 # Menghitung jumlah kemunculan setiap nilai di kolom 'quality'
jumlah_kualitas = df['quality'].value_counts()

 # Menghitung persentase kemunculan setiap nilai
persentase_kualitas = df['quality'].value_counts(normalize=True) * 100

 # Menggabungkan jumlah dan persentase ke dalam satu DataFrame
 # menggunakan pandas.concat
df_kualitas_distribusi = pd.concat([jumlah_kualitas,
 persentase_kualitas], axis=1)

 # Memberi nama kolom
df_kualitas_distribusi.columns = ['Jumlah', 'Persentase (%)']

 # Memformat kolom 'Persentase (%)' menjadi 2 angka di belakang koma
df_kualitas_distribusi['Persentase (%)'] =
```

```
df_kualitas_distribusi['Persentase (%)'].round(2)

# Menampilkan hasilnya
print("Distribusi Kualitas Anggur (0: Buruk, 1: Baik) dengan
Persentase:")
display(df_kualitas_distribusi)
```

```
Distribusi Kualitas Anggur (0: Buruk, 1: Baik) dengan Persentase:
```

| quality | Jumlah | Persentase (%) |
|---|---|---|
| **0** | 1175 | 86.46 |
| **1** | 184 | 13.54 |

Dari tabel di atas, kita dapat melihat:

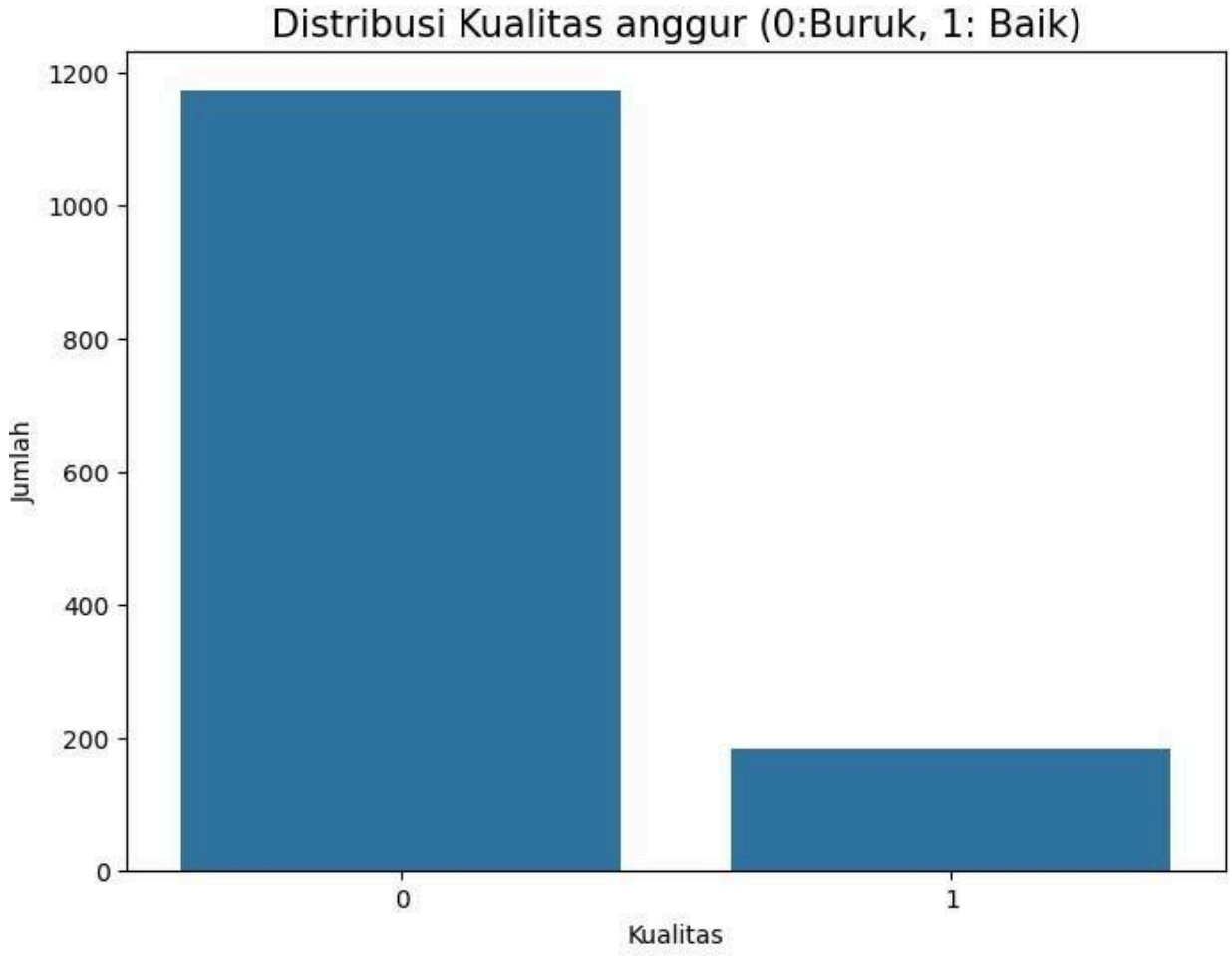
- **Kelas 0 (Buruk)**: Terdapat **1175 sampel**, yang merupakan **86.46%** dari total dataset.
- **Kelas 1 (Baik)**: Terdapat **184 sampel**, yang hanya merupakan **13.54%** dari total dataset.

Ini menunjukkan adanya **ketidakseimbangan kelas yang signifikan**. Kelas "Buruk" (0) adalah kelas mayoritas, sedangkan kelas "Baik" (1) adalah kelas minoritas.

### 3.3.1. Visualisasi Distribusi Kelas

Untuk memperjelas ketidakseimbangan ini, kita akan memvisualisasikannya menggunakan *countplot* dari `seaborn`.

```
plt.figure(figsize=(8, 6))
 sns.countplot(data=df, x="quality")
plt.title('Distribusi Kualitas anggur (0:Buruk, 1: Baik)',
 fontsize=15)
plt.xlabel('Kualitas')
 plt.ylabel('Jumlah')
 plt.show()
```




Grafik batang di atas secara visual mengonfirmasi temuan dari tabel. Batang untuk kualitas "Buruk" (0) jauh lebih tinggi dibandingkan batang untuk kualitas "Baik" (1).

Implikasi untuk Pemodelan:

Ketidakseimbangan kelas ini adalah masalah umum dalam tugas klasifikasi. Jika tidak ditangani, model akan cenderung "malas" dan lebih sering memprediksi kelas mayoritas karena itu akan memberikan akurasi yang tinggi secara keseluruhan, meskipun performanya buruk dalam mengidentifikasi kelas minoritas (yang seringkali justru lebih penting).

Oleh karena itu, dalam tahap pemodelan selanjutnya, kita perlu menerapkan strategi untuk mengatasi ketidakseimbangan ini, seperti:

1. **Penggunaan Metrik Evaluasi yang Tepat**:
- Selain akurasi, kita akan fokus pada *precision*, *recall*, dan *F1-score*, terutama untuk kelas minoritas.
1. **Teknik Resampling**:
- Seperti *oversampling* pada kelas minoritas (misalnya, SMOTE) atau *undersampling* pada kelas mayoritas.
1. **Penyesuaian Bobot Kelas**:
- Memberikan bobot yang lebih tinggi pada kelas minoritas saat melatih model (seperti `class_weight='balanced'` pada Random Forest).

Kesadaran akan ketidakseimbangan ini sejak awal adalah kunci untuk membangun model klasifikasi yang robust dan dapat diandalkan.

## 3.4. Pemisahan Data (Train-Test Split)

Sebelum kita melatih model, dataset perlu dibagi menjadi dua bagian utama:

1. **Set Pelatihan (Train Set)**: Bagian data yang akan digunakan untuk "mengajari" model kita pola-pola yang ada.
2. **Set Pengujian (Test Set)**: Bagian data yang akan digunakan untuk mengevaluasi seberapa baik model yang telah dilatih dapat melakukan prediksi pada data baru yang belum pernah dilihat sebelumnya.

Kita akan memisahkan dataset menjadi data fitur (X) dan data target (y), kemudian membaginya menjadi set pelatihan dan set pengujian. Penggunaan `stratify=y` sangat penting untuk memastikan proporsi kelas pada data latih dan data uji sama dengan proporsi pada data asli, terutama pada kasus dataset tidak seimbang.

Pertama, kita memisahkan kolom-kolom fitur (variabel input) dan kolom target (variabel output).

- `X`: Berisi semua kolom kecuali kolom `quality`. Ini adalah fitur-fitur yang akan digunakan model untuk belajar.
- `y`: Hanya berisi kolom `quality` (yang sudah biner 0 atau 1). Ini adalah apa yang ingin diprediksi oleh model.

Selanjutnya, kita menggunakan fungsi `train_test_split` dari `sklearn` untuk membagi data.

- `test_size=0.2`: Menentukan bahwa **20%** dari data akan digunakan sebagai set pengujian, dan sisanya **80%** sebagai set pelatihan.
- `random_state=42`: Mengatur *seed* untuk generator angka acak. Ini memastikan bahwa jika kita menjalankan kode ini lagi, pembagian datanya akan selalu sama. Ini penting untuk reproduktifitas hasil.
- **`stratify=y`**: Ini adalah parameter yang **sangat penting**, terutama karena kita tahu dataset kita tidak seimbang. Dengan menggunakan `stratify=y`, kita memastikan bahwa proporsi kelas (0 dan 1) pada variabel target `y` akan **dijaga sama** baik di set pelatihan maupun di set pengujian. Artinya, jika kelas "Baik" (1) adalah 13.54% di dataset asli, maka di set pelatihan dan set pengujian juga akan sekitar 13.54%. Ini mencegah situasi di mana salah satu set (misalnya, set uji) secara kebetulan memiliki sangat sedikit atau tidak ada sampel dari kelas minoritas.

```
# Memisahkan fitur (X) dan target (y)
X = df.drop('quality', axis=1)
y = df['quality']

# Membagi data menjadi 80% data latih dan 20% data uji
# Menggunakan stratify=y untuk menjaga proporsi kelas
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.2, random_state=42, stratify=y)

print(f'Ukuran X_train: {X_train.shape}')
print(f'Ukuran X_test: {X_test.shape}')
print(f'Ukuran y_train: {y_train.shape}')
print(f'Ukuran y_test: {y_test.shape}')
```

```
Ukuran X_train: (1087, 11)
Ukuran X_test: (272, 11)
Ukuran y_train: (1087,)
Ukuran y_test: (272,)
```

Verifikasi proporsi kelas setelah stratifikasi

```
print("\\nProporsi kelas pada y_train:")
print(y_train.value_counts(normalize=True) * 100)
print("\\nProporsi kelas pada y_test:")
print(y_test.value_counts(normalize=True) * 100)
```

```
\nProporsi kelas pada y_train:
quality
0    86.476541
1    13.523459
Name: proportion, dtype: float64
\nProporsi kelas pada y_test:
quality
0    86.397059
1    13.602941
Name: proportion, dtype: float64
```

Terlihat bahwa proporsi kelas "Baik" dan "Buruk" pada `y_train` dan `y_test` sangat mirip dengan proporsi pada dataset keseluruhan (86.46% untuk Buruk dan 13.54% untuk Baik). Ini menunjukkan bahwa parameter `stratify=y` bekerja dengan baik.

Dengan pembagian data yang tepat ini, kita siap untuk melanjutkan ke tahap pra-pemrosesan lebih lanjut dan pelatihan model.

Melihat nilai `X` dan `y`

```
print("Nilai X:")
display(X)
```

```
Nilai X:
```

```
{"summary":"{\n  \"name\": \"X\",\n  \"rows\": 1359,\n  \"fields\": [\
n    {\n      \"column\": \"fixed acidity\",\n      \"properties\": {\ n
\"dtype\": \"number\",\n                    \"std\":
\"min\": 4.6,\n  \"max\": 15.9,\n          1.736989807532466,\n
96,\n          \"samples\":                5.3,\n  \"num_unique_values\":
12.6\n         [\n\n        \"semantic_type\":
                    \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"volatile acidity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.18303131761907185,\n
\"min\": 0.12,\n      \"max\": 1.58,\n  \"num_unique_values\":
143,\n
\"samples\": [\n          1.025,\n          0.4,\n          0.87\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"citric acid\",\n
\"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.1955365445504639,\n
\"min\": 0.0,\n            \"max\": 1.0,\n
\"num_unique_values\": 80,\n        \"samples\": [\n
0.37,\n 0.0,\n  0.09\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"residual sugar\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.3523137577104198,\n
\"min\": 0.9,\n      \"max\": 15.5,\n \"num_unique_values\": 91,\n
\"samples\": [\n          11.0,\n          3.0,\n          15.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"chlorides\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.04937686244348626,\n        \"min\": 0.012,\n        \"max\":
0.611,\n        \"num_unique_values\": 153,\n        \"samples\":
[\n 0.096,\n      0.3429999999999999,\n          0.159\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n
}\ n },\n    {\n      \"column\": \"free sulfur dioxide\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
10.447270259048695,\n        \"min\": 1.0,\n        \"max\":
72.0,\n
\"num_unique_values\": 60,\n        \"samples\": [\n
11.0,\n 9.0,\n  32.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total sulfur dioxide\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 33.40894570661654,\n        \"min\":
6.0,\n        \"max\": 289.0,\n        \"num_unique_values\":
144,\n
\"samples\": [\n          68.0,\n          35.0,\n          101.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"density\",\n
\"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.0018689171325591398,\n
\"min\": 0.99007,\n        \"max\":
1.00369,\n        \"num_unique_values\": 436,\n        \"samples\":
[\ n       0.99974,\n          1.0001,\n          0.99471\n
],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"pH\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.15503631128729595,\n
\"min\": 2.74,\n        \"max\": 4.01,\n
\"num_unique_values\": 89,\n          \"samples\": [\n 3.07,\n
            3.0,\n
3.15\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sulphates\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.17066689057420695,\n
\"min\": 0.33,\n      \"max\": 2.0,\n  \"num_unique_values\": 96,\n

\"samples\": [\n 1.07,\n 1.04,\n 1.18\n ],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"alcohol\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 1.0820654499402564,\n \"min\": 8.4,\n \"max\": 14.9,\n \"num_unique_values\": 65,\n \"samples\": [\n 8.5,\n 9.95,\n 9.4\n ],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n ]\n}","type":"dataframe","variable_name":"X"
}

Nilai X:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.700 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.99780 | 3.51 |
| **1** | 7.8 | 0.880 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.99680 | 3.20 |
| **2** | 7.8 | 0.760 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.99700 | 3.26 |
| **3** | 11.2 | 0.280 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.99800 | 3.16 |
| **5** | 7.4 | 0.660 | 0.00 | 1.8 | 0.075 | 13.0 | 40.0 | 0.99780 | 3.51 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1593** | 6.8 | 0.620 | 0.08 | 1.9 | 0.068 | 28.0 | 38.0 | 0.99651 | 3.42 |
| **1594** | 6.2 | 0.600 | 0.08 | 2.0 | 0.090 | 32.0 | 44.0 | 0.99490 | 3.45 |
| **1595** | 5.9 | 0.550 | 0.10 | 2.2 | 0.062 | 39.0 | 51.0 | 0.99512 | 3.52 |
| **1597** | 5.9 | 0.645 | 0.12 | 2.0 | 0.075 | 32.0 | 44.0 | 0.99547 | 3.57 |
| **1598** | 6.0 | 0.310 | 0.47 | 3.6 | 0.067 | 18.0 | 42.0 | 0.99549 | 3.39 |

```
print("Nilai y:")
display(y)
```

```
Nilai y:
```

```
0       0
1       0
2       0
3       0
5       0
       ..
1593    0
1594    0
1595    0
1597    0
```

```
1598    0
Name: quality, Length: 1359, dtype: int64
```

# 4. Pelatihan Model dan Tuning Hyperparameter

Setelah data dipersiapkan, langkah selanjutnya adalah melatih model klasifikasi kita, yaitu **Random Forest Classifier**. Untuk mendapatkan performa model yang optimal, kita tidak hanya melatih model dengan parameter default, tetapi juga melakukan **tuning hyperparameter**.

**Hyperparameter** adalah parameter yang nilainya kita tentukan *sebelum* proses pelatihan model dimulai (berbeda dengan parameter model yang dipelajari selama pelatihan). Contoh hyperparameter pada Random Forest adalah jumlah pohon (`n_estimators`) atau kedalaman maksimum setiap pohon (`max_depth`).

Kita akan menggunakan **`GridSearchCV`** dari `sklearn` untuk mencari kombinasi hyperparameter terbaik secara sistematis. `GridSearchCV` akan mencoba semua kemungkinan kombinasi hyperparameter yang kita definisikan dan mengevaluasinya menggunakan *cross- validation*.

Penjelasan Kode:

1. **`param_grid`**: Kita mendefinisikan sebuah *dictionary* yang berisi daftar hyperparameter dan nilai-nilai yang ingin diuji untuk masing-masing hyperparameter tersebut.
   - `n_estimators`: Jumlah pohon yang akan dibangun.
   - `max_depth`: Seberapa dalam setiap pohon dapat tumbuh. `None` berarti pohon akan tumbuh sampai semua daun murni atau sampai mencapai `min_samples_split`.
   - `min_samples_split`: Jumlah sampel minimum yang harus ada di sebuah node agar node tersebut bisa dipecah lagi.
   - `min_samples_leaf`: Jumlah sampel minimum yang harus ada di setiap *leaf node* (ujung pohon)

2. **`RandomForestClassifier(...)`**: Kita membuat instance dari model Random Forest.
   - **`class_weight='balanced'`**: Ini adalah poin penting. Karena kita memiliki masalah ketidakseimbangan kelas, parameter ini akan secara otomatis menyesuaikan bobot kelas sedemikian rupa sehingga kelas minoritas (anggur "Baik") mendapatkan perhatian lebih selama pelatihan.
   - **`random_state=42`**: Untuk konsistensi hasil.
3. GridSearchCV(...):
   - `estimator=model`: Model yang akan di-tuning.
   - `param_grid=param_grid`: Kisi-kisi hyperparameter yang akan dicoba.
   - `cv=5`: Melakukan 5-fold cross-validation.
   - `n_jobs=-1`: Menggunakan semua prosesor yang tersedia untuk mempercepat pencarian.
   - `verbose=1`: Akan menampilkan pesan selama proses fitting, menunjukkan kombinasi mana yang sedang diuji.
   - **`scoring='recall_weighted'`**: Karena data kita tidak seimbang, akurasi saja tidak cukup. Kita memilih `recall_weighted` sebagai metrik utama untuk dioptimalkan oleh `GridSearchCV`. *Recall weighted* menghitung rata-rata recall dari setiap kelas, dengan bobot berdasarkan jumlah sampel sebenarnya di setiap kelas. Ini memberikan gambaran yang lebih baik tentang kemampuan model mengenali semua kelas, termasuk kelas minoritas.
4. **`grid_search.fit(X_train, y_train)`**: Memulai proses pencarian hyperparameter terbaik pada data latih.
5. **`best_rf = grid_search.best_estimator_`**: Menyimpan model terbaik yang ditemukan oleh `GridSearchCV`.
6. **Output**: Menampilkan kombinasi hyperparameter terbaik dan skor `recall_weighted` rata-rata tertinggi yang dicapai selama cross-validation.

```
 # Menentukan hyperparameter yang akan diuji
param_grid = {
'n_estimators': [50, 100, 200, 500],
'max_depth': [None, 10, 15, 20],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4]
    }

 # Membuat model RandomForest. `class_weight='balanced'` digunakan
 untuk mengatasi masalah class imbalance.
model = RandomForestClassifier(random_state=42,
 class_weight='balanced')
```

```
# Menggunakan Grid Search dengan 5-fold cross-validation
grid_search = GridSearchCV(estimator=model,
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1, # Menggunakan semua core CPU
yang tersedia                                  verbose=1
                                               )

# Melatih model dengan Grid Search
grid_search.fit(X_train, y_train)

# Simpan model terbaik
best_rf = grid_search.best_estimator_

# Menampilka hyperparameter dan skor terbaik
print('\nHyperparameter terbaik:', grid_search.best_params_)
print('akurasi terbaik dari cross-validation:',
grid_search.best_score_)
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits

Hyperparameter terbaik: {'max_depth': 15, 'min_samples_leaf': 2,
'min_samples_split': 2, 'n_estimators': 500}
akurasi terbaik dari cross-validation: 0.8859299031835285
```

Hasil & Interpretasi:

- **Output** `Fitting 5 folds for each of 144 candidates, totalling 720 fits`:
  - Ini menunjukkan bahwa `GridSearchCV` telah menguji 144 kombinasi hyperparameter yang berbeda (`4 nilai n_estimators * 4 nilai max_depth * 3 nilai min_samples_split * 3 nilai min_samples_leaf = 144`). Karena kita menggunakan 5-fold cross-validation, maka total ada `144 * 5 = 720` model yang dilatih dan dievaluasi.
  - **Hyperparameter Terbaik**:
  - Output menunjukkan kombinasi hyperparameter yang menghasilkan skor `recall_weighted` tertinggi selama cross-validation. Dalam kasus ini:
    - `'max_depth': 15`
    - `'min_samples_leaf': 2`
    - `'min_samples_split': 2`
    - `'n_estimators': 500`
  - **Skor Recall Weighted Terbaik**:
  - **0.8859**. Ini adalah skor `recall_weighted` rata-rata dari 5-fold cross-validation menggunakan hyperparameter terbaik tersebut. Angka ini memberikan estimasi yang cukup baik tentang seberapa baik model kita akan bekerja pada data yang belum pernah dilihat.

Dengan model terbaik yang telah ditemukan melalui `GridSearchCV` ini (`best_rf`), kita sekarang siap untuk melakukan prediksi pada data uji dan mengevaluasi kinerjanya secara lebih komprehensif.

# 5. Evaluasi Model

Setelah mendapatkan model terbaik dari proses *tuning hyperparameter* (`best_rf`), langkah selanjutnya adalah mengevaluasi kinerjanya pada **data uji (`X_test`, `y_test`)**, yaitu data yang belum pernah dilihat oleh model selama proses pelatihan. Ini memberikan estimasi yang lebih realistis tentang seberapa baik model akan berkinerja di dunia nyata.

## 5.1. Laporan Klasifikasi & Confusion Matrix

Kita akan menggunakan dua alat utama untuk evaluasi:

1. **Laporan Klasifikasi (`classification_report`)**: Memberikan rincian metrik seperti presisi, recall, dan F1-score untuk setiap kelas.
2. **Confusion Matrix**: Memvisualisasikan performa klasifikasi model dalam bentuk matriks yang menunjukkan prediksi yang benar dan salah.

```
# Melakukan prediksi pada data uji
y_pred = best_rf.predict(X_test)

# Menampilkan laporan klasifikasi
print("Laporan Klasifikasi:")
print(classification_report(y_test,
y_pred))
```

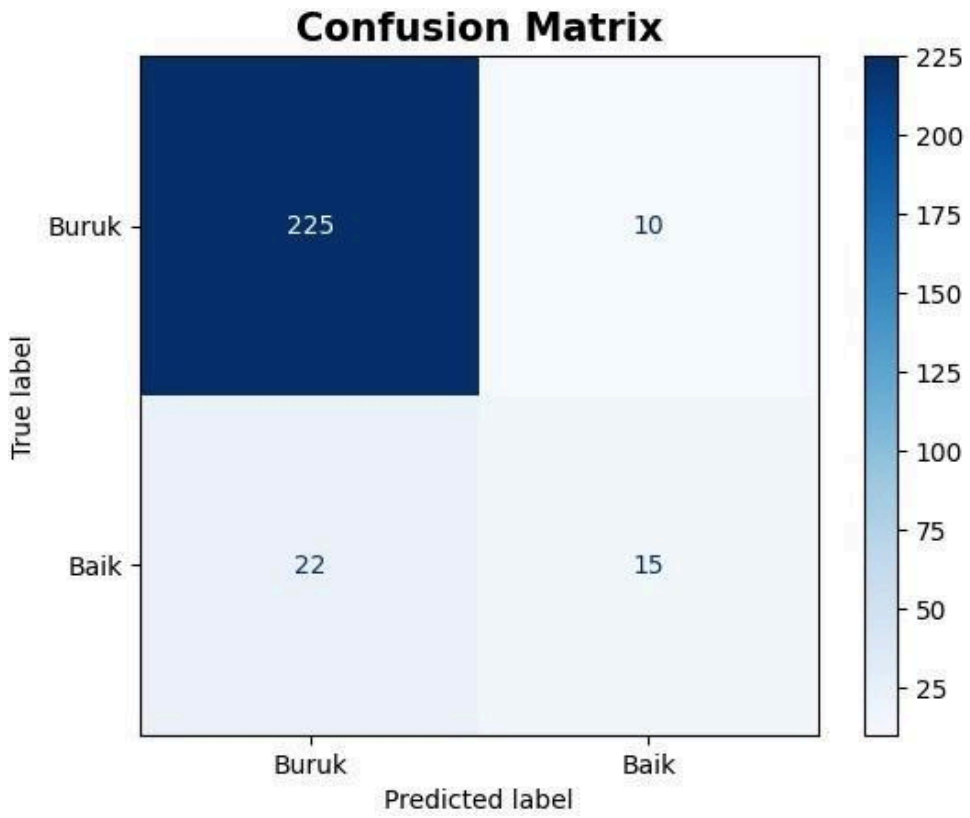
```
Laporan Klasifikasi:
              precision    recall  f1-score   support

           0       0.91      0.96      0.93       235
           1       0.60      0.41      0.48        37

    accuracy                           0.88       272
   macro avg       0.76      0.68      0.71       272
weighted avg       0.87      0.88      0.87       272
```

Laporan Klasifikasi memberikan metrik performa untuk masing-masing kelas ("Buruk (0)" dan "Baik (1)") serta rata-rata:

- `precision`:
  - **Untuk Kelas "Buruk (0)" (0.91)**: Dari semua anggur yang diprediksi sebagai "Buruk", 91% di antaranya memang benar-benar "Buruk".
  - **Untuk Kelas "Baik (1)" (0.60)**: Dari semua anggur yang diprediksi sebagai "Baik", 60% di antaranya memang benar-benar "Baik". Ini adalah peningkatan yang cukup baik dibandingkan model awal (sebelumnya 0.42).
- `recall`:
  - **Untuk Kelas "Buruk (0)" (0.96)**: Model berhasil mengidentifikasi 96% dari semua anggur yang sebenarnya "Buruk".

  - **Untuk Kelas "Baik (1)" (0.41)**: Model berhasil mengidentifikasi 41% dari semua anggur yang sebenarnya "Baik". Meskipun masih belum sempurna, ini adalah peningkatan signifikan dari model awal (sebelumnya 0.26), menunjukkan bahwa penggunaan `class_weight='balanced'` dan tuning hyperparameter (terutama dengan *scoring* `recall_weighted`) memberikan dampak positif.
- `f1-score`: Rata-rata harmonik dari precision dan recall.
  - **Untuk Kelas "Buruk (0)" (0.93)**: Sangat baik.
  - **Untuk Kelas "Baik (1)" (0.48)**: Cukup baik, dan merupakan peningkatan dari model awal (0.32). Ini menunjukkan keseimbangan yang lebih baik antara precision dan recall untuk kelas minoritas.
- `support`: Jumlah sampel sebenarnya untuk setiap kelas di data uji.
- `accuracy` **(0.88)**: Akurasi keseluruhan model pada data uji adalah 88%.
- `macro avg`: Rata-rata metrik tanpa memperhitungkan proporsi kelas. F1-score 0.71 menunjukkan performa rata-rata yang lebih baik antar kelas.
- `weighted avg`: Rata-rata metrik dengan memperhitungkan proporsi kelas. F1-score 0.87 menunjukkan performa keseluruhan yang baik, tetapi kita tahu ini lebih dipengaruhi oleh kelas mayoritas.

```
 # Membuat dan menampilkan confusion matrix
cm = confusion_matrix(y_test, y_pred)
 ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Buruk',
 'Baik']).plot(cmap='Blues')
plt.title('Confusion Matrix', fontsize=15, fontweight='bold')
 plt.show()
```




Confusion matrix memberikan visualisasi yang lebih detail:

- **True Positives (TP)** untuk kelas "Baik" (pojok kanan bawah): **15**. Model memprediksi 15 anggur sebagai "Baik" dan prediksi tersebut benar.
- **True Negatives (TN)** untuk kelas "Buruk" (pojok kiri atas): **225**. Model memprediksi 225 anggur sebagai "Buruk" dan prediksi tersebut benar.
- **False Positives (FP)** untuk kelas "Baik" (Type I Error, pojok kanan atas): **10**. Model memprediksi 10 anggur sebagai "Baik", padahal sebenarnya "Buruk".
- **False Negatives (FN)** untuk kelas "Baik" (Type II Error, pojok kiri bawah): **22**. Model memprediksi 22 anggur sebagai "Buruk", padahal sebenarnya "Baik".

Dari confusion matrix:

- Precision untuk kelas "Baik" = TP / (TP + FP) = 15 / (15 + 10) = 15 / 25 = 0.60.
- Recall untuk kelas "Baik" = TP / (TP + FN) = 15 / (15 + 22) = 15 / 37 = 0.405 (dibulatkan menjadi 0.41 di laporan).

Model yang telah di-tuning dengan `GridSearchCV` dan menggunakan `class_weight='balanced'` menunjukkan **peningkatan yang signifikan** dalam kemampuannya mengidentifikasi kelas minoritas ("Baik"), terutama pada metrik *recall* dan *F1-score* untuk kelas 1, dibandingkan dengan model awal. Meskipun *recall* untuk kelas "Baik" masih 0.41, ini sudah jauh lebih baik. Masih ada ruang untuk perbaikan lebih lanjut, mungkin dengan

teknik *oversampling* seperti SMOTE jika *recall* yang lebih tinggi untuk kelas "Baik" sangat diinginkan, namun dengan risiko peningkatan *false positives*.

## 5.2. Kurva ROC (Receiver Operating Characteristic) dan AUC

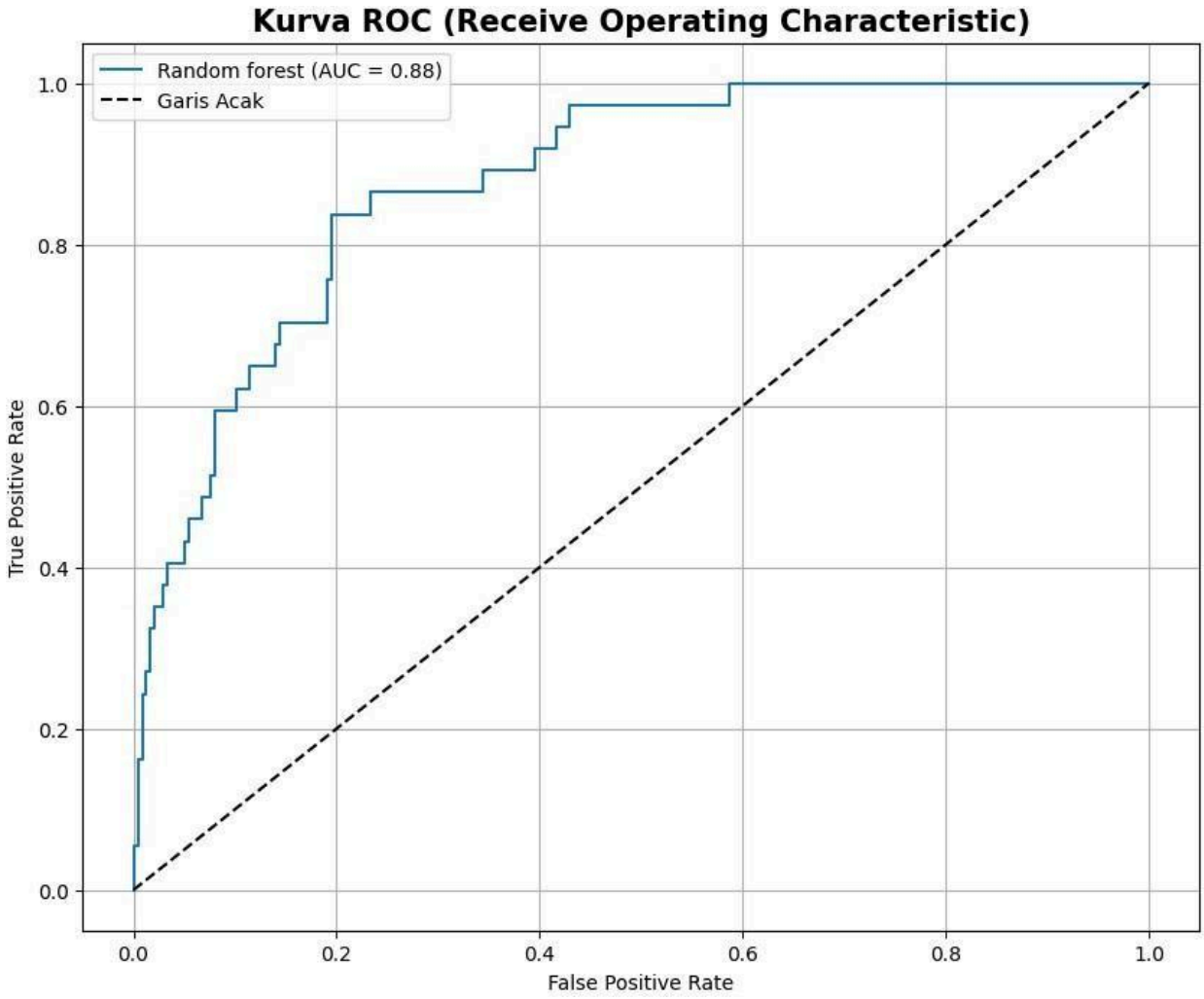
**Kurva ROC** adalah alat evaluasi grafis yang sangat baik untuk masalah klasifikasi biner. Kurva ini memvisualisasikan kemampuan model dalam membedakan antara dua kelas (dalam kasus ini, "Baik" dan "Buruk") pada berbagai ambang batas (*threshold*) klasifikasi.

- **Sumbu X (False Positive Rate - FPR)**: Proporsi dari sampel negatif yang salah diklasifikasikan sebagai positif. (Anggur "Buruk" yang salah diprediksi sebagai "Baik").
  - `FPR = FP / (FP + TN)`
- **Sumbu Y (True Positive Rate - TPR atau Recall/Sensitivity)**: Proporsi dari sampel positif yang benar diklasifikasikan sebagai positif. (Anggur "Baik" yang benar diprediksi sebagai "Baik").
  - `TPR = TP / (TP + FN)`

**Nilai AUC (Area Under the Curve)** adalah ukuran numerik dari performa keseluruhan model.

- Nilai AUC berkisar antara 0 hingga 1.
- AUC **0.5** menunjukkan model yang tidak lebih baik dari tebakan acak (diwakili oleh garis diagonal putus-putus pada grafik, disebut "Garis Acak").
- AUC **1.0** menunjukkan model yang sempurna dalam membedakan kelas.
- Semakin tinggi nilai AUC (mendekati 1), semakin baik kemampuan model dalam membedakan antara kelas positif dan negatif.

```
# Menghitung probabilitas prediksi
y_pred_proba = best_rf.predict_proba(X_test)[:, 1]

# Menghitung AUC Score
auc = roc_auc_score(y_test, y_pred_proba)
print(f"AUC score: {auc:.4f}")

AUC score: 0.8771

# Membuat plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(10, 8))
plt.plot(fpr, tpr, label=f'Random forest (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--', label='Garis Acak')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Kurva ROC (Receive Operating Characteristic)', fontsize=15,
fontweight='bold')
plt.legend(loc='best')
plt.grid()
plt.show()
```




- **AUC Score: 0.8771** (atau 0.88 jika dibulatkan) Nilai AUC sebesar **0.88** menunjukkan bahwa model kita memiliki kemampuan yang **sangat baik** dalam membedakan antara anggur berkualitas "Baik" dan "Buruk". Ini jauh lebih baik daripada tebakan acak (AUC = 0.5).

- **Kurva ROC**:
  - Kurva ROC untuk model Random Forest kita (garis biru) berada **jauh di atas** garis diagonal acak (garis hitam putus-putus).
  - Semakin kurva mendekati pojok kiri atas (di mana TPR = 1 dan FPR = 0), semakin baik performa model. Kurva kita menunjukkan tren yang baik ke arah tersebut.
  - Bentuk kurva yang naik tajam di awal dan kemudian mendatar menunjukkan bahwa model dapat mencapai *true positive rate* yang tinggi dengan *false positive rate* yang relatif rendah pada beberapa *threshold*.

Meskipun laporan klasifikasi sebelumnya menunjukkan tantangan dalam *recall* untuk kelas "Baik", nilai AUC yang tinggi (0.88) mengindikasikan bahwa model secara keseluruhan memiliki daya diskriminatif yang baik. Artinya, jika kita menyesuaikan *threshold* probabilitas untuk

klasifikasi, kita berpotensi menemukan titik operasi di mana model lebih sensitif terhadap kelas "Baik", meskipun mungkin dengan mengorbankan sedikit presisi (meningkatkan *false positives*).

AUC adalah metrik yang berguna, terutama untuk dataset yang tidak seimbang, karena ia mengevaluasi kinerja model di semua *threshold* klasifikasi yang mungkin.

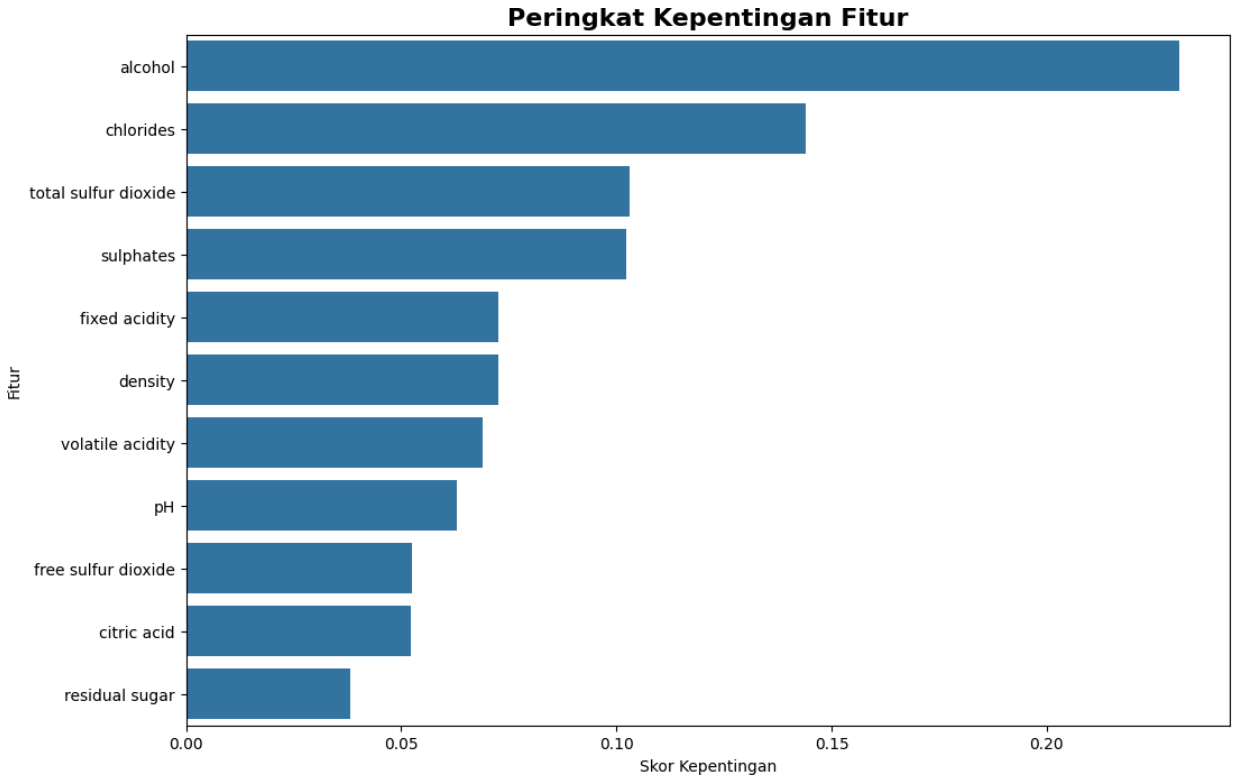
# 6. Analisis Kepentingan Fitur (Feature Importance)

Salah satu keunggulan model berbasis pohon seperti Random Forest adalah kemampuannya untuk memberikan ukuran **kepentingan fitur** (*feature importance*). Ini memberi tahu kita seberapa besar kontribusi masing-masing fitur input dalam membuat prediksi.

Fitur yang lebih penting memiliki dampak yang lebih besar pada keputusan yang dibuat oleh pohon-pohon di dalam *forest*. Informasi ini sangat berguna untuk:

- Memahami faktor-faktor kunci yang memengaruhi kualitas anggur.
- Potensi untuk seleksi fitur (memilih fitur yang paling relevan untuk model yang lebih sederhana).

```
# Membuat series yang menyimpan feature importance dari model dan
feature names dari training data
feature_importances = pd.Series(best_rf.feature_importances_,
index=X.columns.sort_values(ascending=False))

# Membuat plot bar chart
plt.figure(figsize=(12, 8))
feature_importances_sorted =
feature_importances.sort_values(ascending=False)
sns.barplot(x=feature_importances_sorted,
y=feature_importances_sorted.index)
plt.title('Peringkat Kepentingan Fitur', fontsize=16,
fontweight='bold')
plt.xlabel('Skor Kepentingan')
plt.ylabel('Fitur')
plt.show()
```




```
print('Fitur Paling Berpengaruh')
feature_importances.sort_values(ascending=False)
```

```
Fitur Paling Berpengaruh
alcohol                 0.230887
chlorides               0.144058
total sulfur dioxide    0.103051
sulphates               0.102256
fixed acidity           0.072671
density                 0.072492
volatile acidity        0.069007
pH                      0.062800
free sulfur dioxide     0.052376
citric acid             0.052307
residual sugar          0.038095
dtype: float64
```

Grafik batang dan output numerik di atas menunjukkan peringkat kepentingan fitur berdasarkan kontribusinya dalam model Random Forest yang telah di-tuning:

1. `alcohol` **(Skor ~0.231)**: Fitur ini adalah yang **paling berpengaruh** dalam menentukan kualitas anggur. Semakin tinggi kandungan alkohol, semakin besar kemungkinannya anggur dianggap "Baik". Ini konsisten dengan temuan kita dari analisis korelasi.
2. `chlorides` **(Skor ~0.144)**: Klorida menjadi fitur terpenting kedua. Ini sedikit mengejutkan karena korelasinya dengan `quality` (-0.13) tidak sebesar fitur lain

seperti

`sulphates` atau `volatile acidity`. Namun, dalam model non-linear seperti Random Forest, interaksi antar fitur dapat membuat fitur dengan korelasi individual yang lebih rendah menjadi penting.

3. **`total sulfur dioxide` (Skor ~0.103)**: Jumlah total $SO_2$ juga memiliki kontribusi yang signifikan.
4. **`sulphates` (Skor ~0.102)**: Sulfat, yang juga berkorelasi positif dengan kualitas, berada di peringkat keempat.
5. **`fixed acidity` (Skor ~0.073)**
6. **`density` (Skor ~0.072)**
7. **`volatile acidity` (Skor ~0.069)**
8. **`pH` (Skor ~0.063)**
9. **`free sulfur dioxide` (Skor ~0.052)**
10. **`citric acid` (Skor ~0.052)**
11. **`residual sugar` (Skor ~0.038)**: Sisa gula memiliki pengaruh paling kecil dalam model ini.

**Kesimpulan Feature Importance:**

- Seperti pada analisis korelasi, `alcohol` kembali menjadi fitur yang paling penting, diikuti oleh `chlorides` dan `total sulfur dioxide`.
- Kepentingan fitur ini memberikan wawasan tentang atribut mana yang paling menjadi fokus model dalam melakukan klasifikasi. Ini bisa menjadi dasar untuk eksplorasi lebih lanjut atau rekayasa fitur di masa depan.
- Penting untuk diingat bahwa kepentingan fitur di sini dihitung berdasarkan bagaimana model Random Forest *menggunakan* fitur tersebut untuk mengurangi ketidakmurnian (*impurity*) pada *node-node* pohonnya, dan mungkin berbeda dengan korelasi linear sederhana.

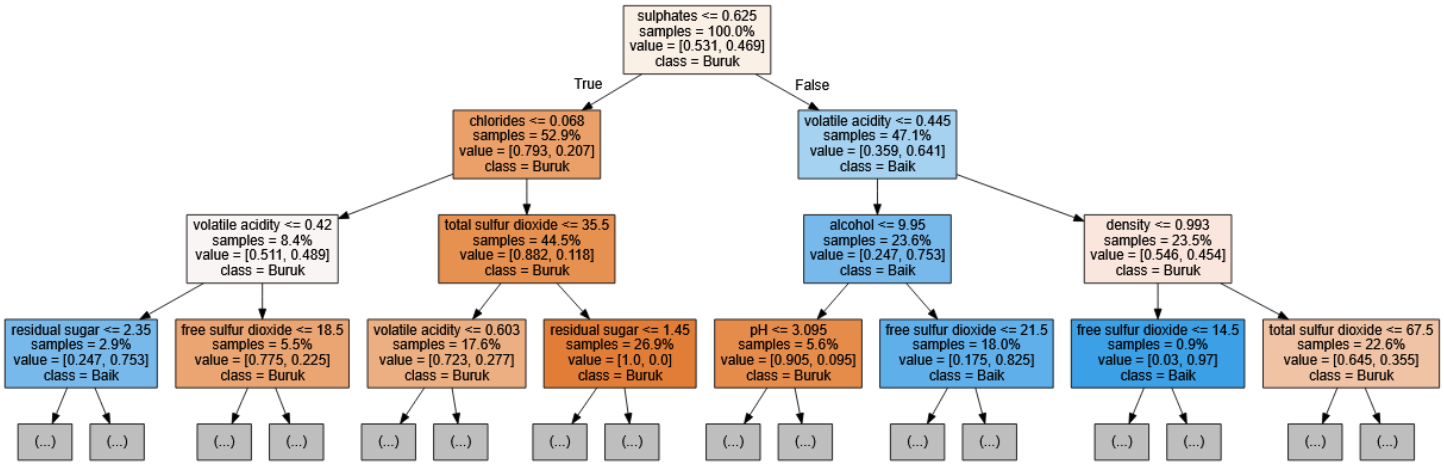
# 7. Visualisasi Tree

Untuk memahami bagaimana model Random Forest membuat keputusan, kita bisa memvisualisasikan beberapa *decision tree* (pohon keputusan) pertama yang membentuk *forest* tersebut. Random Forest adalah kumpulan dari banyak pohon keputusan, dan prediksi akhirnya adalah hasil mayoritas (untuk klasifikasi) atau rata-rata (untuk regresi) dari prediksi semua pohon.

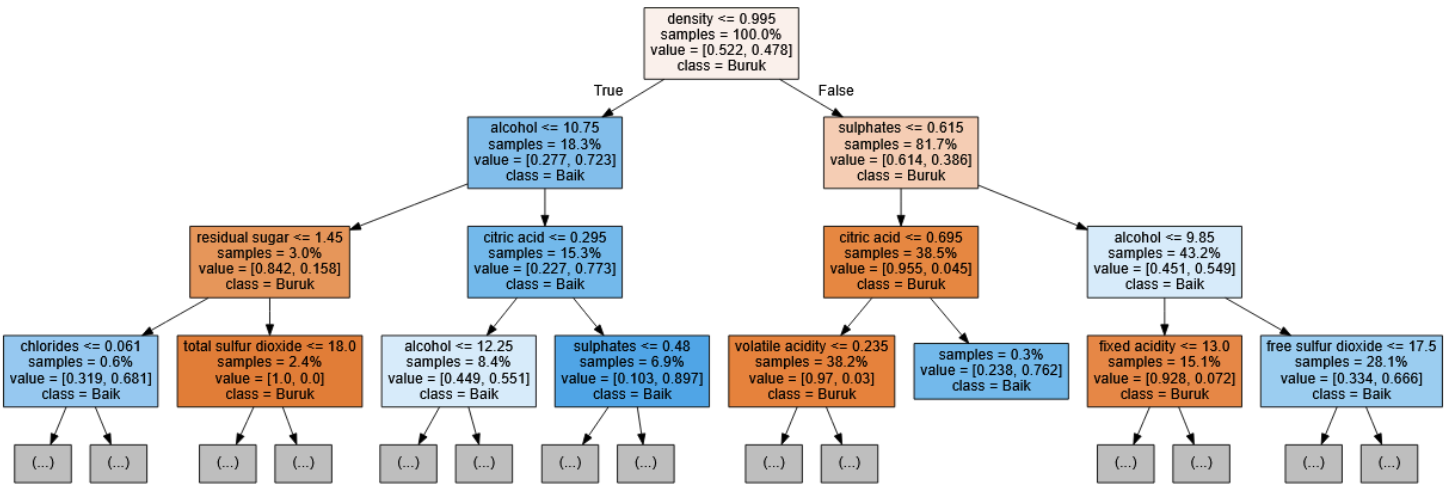
Dengan melihat struktur pohon individual, kita bisa mendapatkan intuisi tentang aturan-aturan (*rules*) yang dipelajari model berdasarkan fitur-fitur yang ada.

```
 # Visualisasi 3 pohon keputusan pertama dari model Random Forest
for i in range(3):
tree = best_rf.estimators_[i] dot_data
     = export_graphviz(tree,
feature_names=X_train.columns, # Nama-
 nama fitur
class_names=['Buruk', 'Baik'], # Nama
 kelas target
                                filled=True,      # Memberi
```

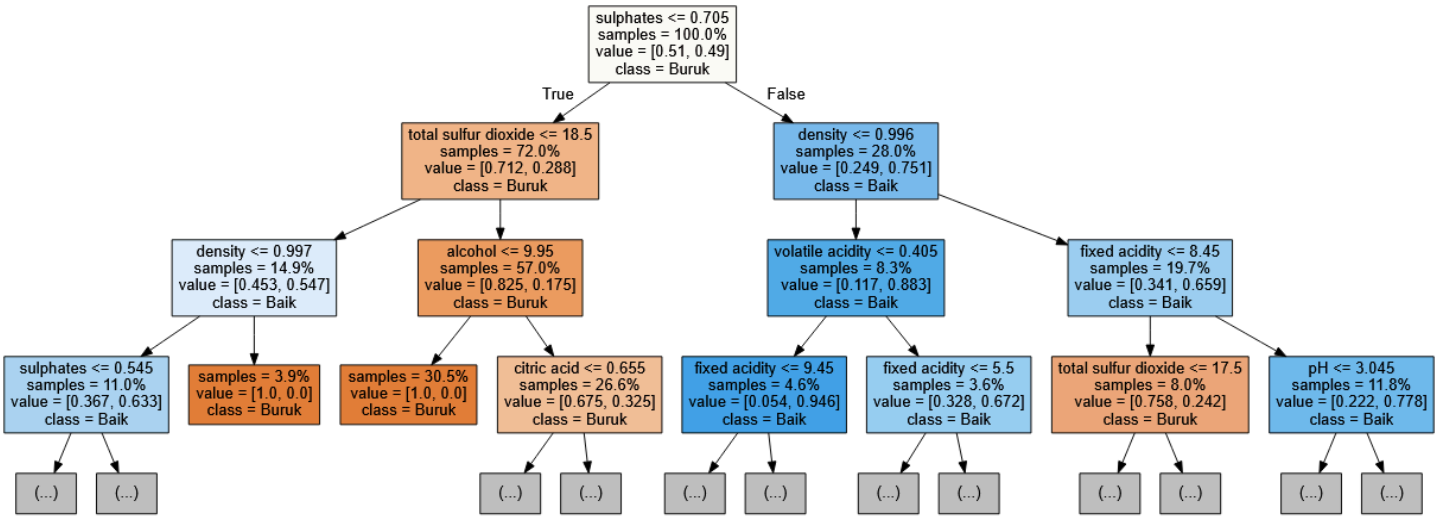
```
warna pada node berdasarkan kelas mayoritas
                                max_depth=3,                    #
Membatasi kedalaman untuk visualisasi yang lebih sederhana    # Tidak
                                impurity=False,                 #
menampilkan impurity
                                proportion=True)
Menampilkan proporsi sampel, bukan jumlah absolut

    graph = graphviz.Source(dot_data)
    print(f'Visualisasi Pohon
    ke-{i+1}:') display(graph)

Visualisasi Pohon ke-1:
```




```
Visualisasi Pohon ke-2:
```



```
Visualisasi Pohon ke-3:
```




Dengan melihat beberapa pohon, kita bisa melihat bahwa fitur yang berbeda mungkin menjadi penting di pohon yang berbeda, atau pada kedalaman yang berbeda. Ini adalah salah satu kekuatan Random Forest – ia menggabungkan "pandangan" dari banyak pohon yang beragam.

```
# Hasil dari X_test dataframe yang digabung dengan prediksi dan target
column yg asli(quality)
final_df = pd.concat([X_test.reset_index(drop=True),
                      y_test.reset_index(drop=True)], axis=1)
final_df['quality_prediction'] = pd.Series(y_pred)

# Print dataframe X_test dengan y_pred dan quality
print(f"Hasil prediksi dari X_test :")
display(final_df)
```

```
Hasil prediksi dari X_test :
```

Hasil prediksi dari X_test :

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 8.0 | 1.18 | 0.21 | 1.9 | 0.083 | 14.0 | 41.0 | 0.99532 | 3.34 |
| **1** | 7.5 | 0.55 | 0.24 | 2.0 | 0.078 | 10.0 | 28.0 | 0.99830 | 3.45 |
| **2** | 7.2 | 0.34 | 0.24 | 2.0 | 0.071 | 30.0 | 52.0 | 0.99576 | 3.44 |
| **3** | 9.0 | 0.60 | 0.29 | 2.0 | 0.069 | 32.0 | 73.0 | 0.99654 | 3.34 |
| **4** | 7.1 | 0.66 | 0.00 | 3.9 | 0.086 | 17.0 | 45.0 | 0.99760 | 3.46 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **267** | 7.4 | 0.64 | 0.07 | 1.8 | 0.100 | 8.0 | 23.0 | 0.99610 | 3.30 |
| **268** | 10.4 | 0.28 | 0.54 | 2.7 | 0.105 | 5.0 | 19.0 | 0.99880 | 3.25 |
| **269** | 5.4 | 0.42 | 0.27 | 2.0 | 0.092 | 23.0 | 55.0 | 0.99471 | 3.78 |
| **270** | 9.6 | 0.60 | 0.50 | 2.3 | 0.079 | 28.0 | 71.0 | 0.99970 | 3.50 |
| **271** | 6.5 | 0.51 | 0.15 | 3.0 | 0.064 | 12.0 | 27.0 | 0.99290 | 3.33 |

**Interpretasi Tabel Hasil Prediksi:**

Tabel `final_df` di atas menampilkan:

- Kolom-kolom fitur dari `X_test`.
- Kolom `quality`: Ini adalah label kualitas **sebenarnya** dari anggur tersebut (0 untuk "Buruk", 1 untuk "Baik").
- Kolom `quality_prediction`: Ini adalah label kualitas yang **diprediksi** oleh model Random Forest kita.

Dengan membandingkan kolom `quality` dan `quality_prediction`, kita bisa melihat secara langsung sampel mana yang berhasil diprediksi dengan benar dan mana yang salah.

**Contoh**:

- Jika pada suatu baris `quality` adalah 1 dan `quality_prediction` juga 1, berarti model berhasil memprediksi anggur "Baik" dengan benar (True Positive).
- Jika `quality` adalah 0 dan `quality_prediction` adalah 1, berarti model salah memprediksi anggur "Buruk" sebagai "Baik" (False Positive).

Tabel ini berguna untuk analisis kesalahan (*error analysis*) secara lebih mendalam jika diperlukan, untuk memahami kasus-kasus spesifik di mana model kita mungkin membuat kesalahan.

# 8. Simulasi Prediksi

Untuk menunjukkan bagaimana model yang telah dilatih dapat digunakan dalam praktiknya, kita akan melakukan simulasi prediksi pada beberapa sampel data hipotetis. Di bawah ini disajikan tiga skenario input data yang berbeda untuk melihat bagaimana model memberikan prediksinya.

## 8.1. Mendefinisikan Fungsi Simulasi

Pertama, kita akan membuat fungsi yang lebih modular. Fungsi ini akan menerima sebuah *dictionary* data sebagai input, melakukan prediksi, dan menampilkan hasilnya dengan rapi.

```
def jalankan_simulasi_prediksi(data_input, nama_sampel):
    """
    Fungsi untuk menjalankan dan menampilkan hasil simulasi prediksi.

    Parameters:
    - data_input (dict): Dictionary yang berisi nilai fitur untuk satu
      sampel.
    - nama_sampel (str): Nama atau label untuk sampel yang diuji.
    """
```

```
  label_string = {0: "Buruk", 1: "Baik"}

  # Membuat DataFrame dari input
  df_sim = pd.DataFrame([data_input])

  # Melakukan prediksi menggunakan model terbaik
  prediksi = best_rf.predict(df_sim)
  prediksi_proba = best_rf.predict_proba(df_sim)

  # Menampilkan hasil
  print(f"\n\n--- Hasil Prediksi untuk {nama_sampel} ---")
  print("Data Input:")
  display(df_sim)

  kualitas_prediksi = prediksi[0]
  label_kualitas = label_string[kualitas_prediksi]
  prob_buruk = prediksi_proba[0][0]
  prob_baik = prediksi_proba[0][1]

  print(f"\nPrediksi Kualitas: {kualitas_prediksi}
({label_kualitas})")
  print(f"Keyakinan Model (Probabilitas):")
  print(f" - Kualitas Buruk: {prob_buruk:.2%}")
  print(f" - Kualitas Baik: {prob_baik:.2%}")
  print("-" * 40)
```

## 8.2. Menyiapkan Data Sampel untuk Simulasi

Kita akan mendefinisikan tiga set data. Masing-masing set data ini memiliki karakteristik yang berbeda untuk menguji respon model.

1. **Sampel 1: Potensi Kualitas Baik** Karakteristiknya mengarah pada anggur berkualitas baik (contoh: alkohol tinggi, keasaman volatil rendah).
2. **Sampel 2: Potensi Kualitas Buruk** Karakteristiknya cenderung menghasilkan anggur berkualitas buruk (contoh: alkohol rendah, keasaman volatil tinggi).
3. **Sampel 3: Kualitas Ambigu/Tengah** Karakteristik campuran yang membuat prediksi lebih menantang bagi model.

```
 # Skenario 1: Input dengan potensi kualitas "Baik"
sampel_baik = {
'fixed acidity': 8.5,
'volatile acidity': 0.35,
'citric acid': 0.45,
'residual sugar': 2.0,
'chlorides': 0.07,
'free sulfur dioxide': 15, 'total
    sulfur dioxide': 40,
    'density': 0.995,
'pH': 3.3,
```

```
'sulphates': 0.75,
    'alcohol': 12.5 # Nilai alkohol tinggi
}

# Skenario 2: Input dengan potensi kualitas "Buruk"
sampel_buruk = {
'fixed acidity': 7.0,
    'volatile acidity': 0.8, # Keasaman volatil tinggi
'citric acid': 0.1,
'residual sugar': 2.5,
'chlorides': 0.09,
'free sulfur dioxide': 8, 'total
    sulfur dioxide': 60,
    'density': 0.998,
'pH': 3.5,
'sulphates': 0.5,
    'alcohol': 9.2 # Nilai alkohol rendah
}

# Skenario 3: Input dengan nilai rata-rata/ambigu
sampel_tengah = {
'fixed acidity': 8.3,
'volatile acidity': 0.53,
'citric acid': 0.27,
'residual sugar': 2.5,
'chlorides': 0.08,
'free sulfur dioxide': 16, 'total
    sulfur dioxide': 47,
    'density': 0.996,
'pH': 3.3,
'sulphates': 0.65,
'alcohol': 10.4
}
```

## 8.3. Menjalankan Simulasi

Sekarang kita panggil fungsi yang telah dibuat untuk setiap sampel data.

```
# Jalankan simulasi untuk setiap sampel
jalankan_simulasi_prediksi(sampel_baik, "Sampel Potensi Baik")
jalankan_simulasi_prediksi(sampel_buruk, "Sampel Potensi Buruk")
jalankan_simulasi_prediksi(sampel_tengah, "Sampel Ambigu")
```

```
--- Hasil Prediksi untuk Sampel Potensi Baik
--- Data Input:

{"summary":"{\n \"name\": \"jalankan_simulasi_prediksi(sampel_tengah,
\\\"Sampel Ambigu\\\")\",\n \"rows\": 1,\n \"fields\": [\n {\n
```

```
Prediksi Kualitas: 1 (Baik)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 22.97%
  - Kualitas Baik: 77.03%


--- Hasil Prediksi untuk Sampel Potensi Buruk
--- Data Input:
Prediksi Kualitas: 0 (Buruk)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 100.00%
  - Kualitas Baik: 0.00%


--- Hasil Prediksi untuk Sampel Ambigu ---
Data Input:


Prediksi Kualitas: 0 (Buruk)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 93.41%
```

```
--- Hasil Prediksi untuk Sampel Potensi Baik ---
Data Input:
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sul |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 8.5 | 0.35 | 0.45 | 2.0 | 0.07 | 15 | 40 | 0.995 | 3.3 | |

```
Prediksi Kualitas: 1 (Baik)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 22.97%
  - Kualitas Baik: 77.03%
----------------------------------------


--- Hasil Prediksi untuk Sampel Potensi Buruk ---
Data Input:
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sul |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.0 | 0.8 | 0.1 | 2.5 | 0.09 | 8 | 60 | 0.998 | 3.5 | |

```
Prediksi Kualitas: 0 (Buruk)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 100.00%
  - Kualitas Baik: 0.00%
----------------------------------------
--- Hasil Prediksi untuk Sampel Ambigu ---
Data Input:
```

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sul |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 8.3 | 0.53 | 0.27 | 2.5 | 0.08 | 16 | 47 | 0.996 | 3.3 | |

```
Prediksi Kualitas: 0 (Buruk)
Keyakinan Model (Probabilitas):
  - Kualitas Buruk: 93.41%
  - Kualitas Baik: 6.59%
----------------------------------------
```

# 9. Kesimpulan Akhir

Proyek ini bertujuan untuk membangun model klasifikasi guna memprediksi kualitas anggur merah "Vinho Verde" berdasarkan atribut fisikokimianya, dengan fokus pada penggunaan *Random Forest Classifier*. Proses analisis dan pemodelan telah melalui serangkaian tahapan yang komprehensif, mulai dari pemahaman dataset, pra-pemrosesan data, eksplorasi data, pelatihan model, hingga evaluasi kinerja.

## 9.1. Ringkasan Metodologi dan Pra-pemrosesan Data

Data yang digunakan bersumber dari UCI Machine Learning Repository dan diakses melalui Kaggle, terdiri dari 11 fitur input (karakteristik fisikokimia) dan 1 variabel output (kualitas sensorik). Langkah pra-pemrosesan data awal meliputi:

### 9.1.1. Pembersihan Data

Dari 1599 baris data awal, teridentifikasi dan dihapus 240 baris data duplikat, menghasilkan dataset akhir sebanyak 1359 sampel. Tidak ditemukan adanya nilai yang hilang (*missing values*) pada setiap kolom, sehingga tidak diperlukan proses imputasi.

### 9.1.2. Transformasi Variabel Target:

Variabel `quality` yang awalnya memiliki skala 0-10 ditransformasikan menjadi variabel biner untuk keperluan klasifikasi: nilai 1 (Baik) untuk skor asli >= 7, dan nilai 0 (Buruk) untuk skor asli < 7.

### 9.1.3. Analisis Data Eksploratif (EDA)

- Visualisasi distribusi fitur mengungkapkan bahwa sebagian besar fitur seperti `density`, `pH`, dan `fixed acidity` mendekati distribusi normal, sementara fitur lain seperti `residual sugar`, `chlorides`, `free sulfur dioxide`, dan `total sulfur dioxide` menunjukkan kemiringan ke kanan (*right-skewed*), mengindikasikan adanya *outlier*.
- Analisis korelasi menunjukkan bahwa `alcohol` memiliki korelasi positif tertinggi (0.48) dengan `quality`, sedangkan `volatile acidity` memiliki korelasi negatif tertinggi (-0.40). Teridentifikasi pula korelasi antar fitur input, seperti antara `fixed acidity` dan `citric acid` (0.67).

### 9.1.4. Penanganan Ketidakseimbangan Kelas

Transformasi variabel target menghasilkan ketidakseimbangan kelas yang signifikan, dengan 1175 sampel untuk anggur berkualitas "Buruk" (Kelas 0) dan hanya 184 sampel untuk anggur berkualitas "Baik" (Kelas 1).

### 9.1.5. Pemisahan Data

Dataset dibagi menjadi 80% data latih dan 20% data uji dengan menggunakan stratifikasi (`stratify=y`) untuk memastikan proporsi kelas tetap terjaga pada kedua set data.

## 9.2. Pelatihan Model dan Optimalisasi

Model `RandomForestClassifier` dipilih dan dilatih menggunakan data latih. Untuk mendapatkan performa optimal dan mengatasi ketidakseimbangan kelas, diterapkan strategi `class_weight='balanced'` pada model. Selanjutnya, dilakukan *tuning hyperparameter* menggunakan `GridSearchCV` dengan 5-fold *cross-validation*. Pencarian hyperparameter

difokuskan untuk memaksimalkan metrik `recall_weighted`. Kombinasi hyperparameter terbaik yang ditemukan adalah:

- `max_depth`: 15
- `min_samples_leaf`: 2
- `min_samples_split`: 2
- `n_estimators`: 500

## 9.3. Evaluasi Kinerja Model

Model terbaik hasil *tuning* dievaluasi pada data uji:

### 9.3.1. Laporan Klasifikasi

- Akurasi keseluruhan model mencapai **88%**.
- Untuk Kelas 0 ("Buruk"): *precision* 0.91, *recall* 0.96, dan *F1-score* 0.93.
- Untuk Kelas 1 ("Baik"): *precision* 0.60, *recall* 0.41, dan *F1-score* 0.48. Hasil ini menunjukkan peningkatan signifikan dalam kemampuan model mengidentifikasi kelas minoritas ("Baik") dibandingkan model awal (tanpa penanganan ketidakseimbangan dan *tuning*), terutama pada *recall* dan *F1-score*.

### 9.3.2. Confusion Matrix

- *True Positives* (Kelas 1): 15
- *True Negatives* (Kelas 0): 225
- *False Positives* (Kelas 1): 10
- *False Negatives* (Kelas 1): 22

### 9.3.3. Kurva ROC dan AUC

Skor AUC (Area Under the ROC Curve) yang dicapai adalah **0.8771 (atau 0.88)**, yang menunjukkan kemampuan diskriminatif model yang sangat baik dalam membedakan antara anggur berkualitas "Baik" dan "Buruk".

## 9.4. Analisis Kepentingan Fitur

Analisis kepentingan fitur dari model Random Forest mengungkapkan bahwa fitur yang paling berpengaruh dalam menentukan kualitas anggur adalah:

1. `alcohol` (skor kepentingan ~0.231)
2. `chlorides` (skor kepentingan ~0.144)
3. `total sulfur dioxide` (skor kepentingan ~0.103)
4. `sulphates` (skor kepentingan ~0.102) Fitur-fitur lain seperti `fixed acidity`, `density`, dan `volatile acidity` juga memberikan kontribusi yang cukup berarti, sementara `residual sugar` memiliki pengaruh paling kecil.

## 9.5. Keterbatasan

Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan:

### 9.5.1. Lingkup Data

Dataset tidak mencakup faktor non-fisikokimia (misalnya, merek, harga, atau jenis

anggur spesifik) yang mungkin juga memengaruhi persepsi kualitas.

### 9.5.2. Definisi Kualitas Biner

Pembagian kualitas menjadi "Baik" dan "Buruk" berdasarkan ambang batas skor >= 7 adalah diskritisasi yang dapat memengaruhi hasil; ambang batas yang berbeda mungkin menghasilkan performa model yang berbeda.

### 9.5.3. Outlier

Meskipun Random Forest umumnya robust terhadap *outlier*, keberadaan *outlier* yang teridentifikasi dalam beberapa fitur tidak ditangani secara khusus dan berpotensi memengaruhi interpretasi statistik deskriptif, meskipun dampaknya pada model Random Forest mungkin minimal.

## 9.6. Kesimpulan

Secara keseluruhan, penelitian ini berhasil mengembangkan model *Random Forest Classifier* yang mampu memprediksi kualitas anggur merah "Vinho Verde" dengan kinerja yang baik, terutama setelah penerapan teknik penanganan ketidakseimbangan kelas dan optimalisasi hyperparameter. Model mencapai akurasi keseluruhan 88% dan skor AUC 0.88 pada data uji, menunjukkan kapabilitas prediktif dan diskriminatif yang solid. Meskipun *recall* untuk kelas minoritas ("Baik") sebesar 0.41 masih menyisakan ruang untuk perbaikan, hal ini merupakan peningkatan yang substansial berkat strategi pemodelan yang diterapkan.

Kandungan alkohol, klorida, total sulfur dioksida, dan sulfat teridentifikasi sebagai faktor-faktor fisikokimia yang paling signifikan dalam menentukan klasifikasi kualitas anggur berdasarkan model yang dibangun. Hasil ini memberikan dasar yang kuat untuk pemahaman lebih lanjut mengenai atribut-atribut penentu kualitas anggur dan dapat menjadi landasan untuk pengembangan model prediktif yang lebih canggih di masa depan.

## 9.7. Saran Pengembangan

Untuk pengembangan lebih lanjut, beberapa langkah dapat dipertimbangkan:

### 9.7.1. Eksplorasi Teknik Imbalance Learning:

Menerapkan teknik *oversampling* (misalnya SMOTE) atau *undersampling* yang lebih lanjut untuk melihat dampaknya pada *recall* kelas minoritas.

### 9.7.2. Pengujian Model Lain:

Membandingkan performa Random Forest dengan model klasifikasi lain, termasuk model yang mungkin memerlukan *feature scaling* atau penanganan *outlier* yang lebih eksplisit.

### 9.7.3. Feature Engineering dan Selection:

Melakukan rekayasa fitur tambahan atau seleksi fitur berdasarkan hasil *feature importance* untuk membangun model yang lebih parsimoni atau lebih akurat.

### 9.7.4. Analisis Kesalahan:

Investigasi lebih mendalam terhadap kasus-kasus di mana model melakukan kesalahan klasifikasi untuk mendapatkan pemahaman yang lebih baik mengenai keterbatasan model.