

Tugas Besar - IF2220 - Probabilitas dan Statistika

Penarikan Kesimpulan dan Pengujian Hipotesis

Enam Langkah Testing:

1. Tentukan Hipotesis nol ($H_0: \theta = \theta_0$), dimana θ bisa berupa μ , σ^2 , p , atau data lain berdistribusi tertentu (normal, binomial, dsc.).
2. Pilih hipotesis alternatif H_1 salah dari $\theta > \theta_0$, $\theta < \theta_0$, atau $\theta \neq \theta_0$
3. Tentukan tingkat signifikan α .
4. Tentukan uji statistik yang sesuai dan tentukan daerah kritis.
5. Hitung nilai uji statistik dari data sample. Hitung p -value sesuai dengan uji statistik yang digunakan.
6. Ambil keputusan dengan **TOLAK** H_0 jika nilai uji terletak di daerah kritis atau dengan tes signifikan, **TOLAK** H_0 jika p -value lebih kecil dibanding tingkat signifikansi α yang diinginkan.

Contributors

NIM	Nama
13522053	Erdianti Wiga Putri Andini
13522063	Shazya Audrea Taufik

Problem-Set

Diberikan sebuah data candy.csv yang merupakan data metrik popularitas candy (permen) yang mengandung 9 kolom sebagai berikut:

1. Calories
2. Serving
3. Protein
4. Sugar
5. Sodium
6. Fat
7. Fiber
8. Flavour
9. Popularity

Kolom 1-8 adalah kolom atribut (non-target), sedangkan kolom 9 adalah kolom target. Kami diminta untuk melakukan analisis statistika sebagai berikut:

Deskripsi Statistika

Menulis deskripsi statistika (Descriptive Statistics) dari semua kolom pada data. Data yang bersifat numerik dapat diberikan nilai mean, median, modus, standar deviasi, variansi, range, nilai minimum, maksimum, kuartil, IQR, skewness dan kurtosis. Data dalam bentuk string dapat dicari unique values, dan proporsi nya.

Outlier Data

Apakah pada data tersebut terdapat outlier? Jika ya, dapatkah anda menanganinya? Jelaskan apa yang umumnya dilakukan untuk menangani outlier.

Visualisasi Plot Distribusi

Membuat Visualisasi plot distribusi. Berikan uraian penjelasan kondisi setiap kolom berdasarkan kedua plot tersebut. Jika numerik dapat dibuat dalam bentuk histogram dan box plot, dan jika string dengan histogram.

Menentukan Setiap Kolom Numerik Berdistribusi Normal atau Tidak

Menentukan distribusi setiap kolom numerik menggunakan hasil visualisasi histogram. Apakah kolom tersebut berdistribusi normal? Jika bukan, terdistribusi seperti apa kolom tersebut?

Melakukan Tes Hipotesis 1 Sampel

- Perusahaan menerima beberapa keluhan bahwa permennya kurang manis. Periksalah anda mengecek apakah rata-rata nilai Sugar di bawah 25?
- Pada umumnya, rata-rata Serving untuk permen adalah 40 gram. Oleh karena itu, periksalah apakah rata-rata Serving permen yang diproduksi tidak bernilai 40 gram!
- Periksalah apakah rata-rata Sodium untuk permen 20 baris terakhir tidak sama dengan 74!
- Periksalah apakah proporsi nilai Protein yang lebih besar dari 3 tidak sama dengan 60% dari total dataset!

Melakukan Tes Hipotesis 2 Sampel

Perusahaan ingin membandingkan kualitas permen yang diproduksi pada paruh awal dan paruh akhir produksi. Hal ini dapat dilakukan dengan membagi 1 dataset menjadi 2 bagian yang sama panjang.

- Periksalah apakah rata-rata Sugar dari permen yang diproduksi bernilai sama pada kedua kurun waktu!
- Bandingkan rata-rata Protein dari permen pada paruh awal dan akhir. Apakah rata-rata Protein pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.3 unit?
- Periksalah apakah variansi dari Sodium dari permen sama pada paruh awal dan akhir!
- Periksalah apakah proporsi Calories dari permen yang lebih dari 200 pada paruh awal lebih besar daripada proporsi di paruh akhir!

Library and Data Loading

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm, t, f, normaltest
import scipy.stats as stats
import numpy as np
import seaborn as sns
from IPython.display import display, Markdown

# read dataset
data = pd.read_csv('candy.csv')
```

Deskripsi Statistika

Penjelasan Atribut Tabel

- Count: Jumlah data non-null dalam setiap kolom. Menggunakan data.describe() untuk memberikan ringkasan statistik deskriptif, termasuk jumlah non-null (Count).
- Mean (Rata-rata): Nilai rata-rata dari setiap kolom numerik dalam dataset.
- Std (Standar Deviasi): Nilai rata-rata dari selisih antara setiap data dengan rata-rata.
- Min (Minimum): Nilai terkecil dalam setiap kolom.
- 25%: Nilai persentil ke-25 (kuartil pertama), membagi data menjadi empat bagian yang sama dalam urutan statistik.
- 50%: Nilai median (nilai tengah) dalam setiap kolom.
- 75%: Nilai persentil ke-75 (kuartil ketiga), membagi data menjadi empat bagian yang sama dalam urutan statistik.
- Max (Maksimum): Nilai terbesar dalam setiap kolom.
- Modus (Modus): Nilai yang paling sering muncul dalam setiap kolom.
- Variansi (Variance): Nilai rata-rata dari selisih kuadrat antara setiap data dengan rata-rata. Nilai ini adalah kuadrat dari standar deviasi.
- Skewness (Kesimpangan): Nilai yang menunjukkan seberapa simetris distribusi data.
- Kurtosis (Kurtosis): Nilai yang menunjukkan seberapa tajam puncak distribusi data.

```
In [3]: # NUMERIK

data_stats = data.describe(include=[np.number])
data_stats.loc['modus'] = data.mode().loc[0]
data_stats.loc['std'] = data.std(numeric_only=True)
data_stats.loc['variansi'] = data.var(numeric_only=True)
data_stats.loc['range'] = data_stats.loc['max'] - data_stats.loc['min']
```

```
data_stats.loc['IQR'] = data_stats.loc['75%'] - data_stats.loc['25%']
data_stats.loc['skewness'] = data.skew(numeric_only=True)
data_stats.loc['kurtosis'] = data.kurt(numeric_only=True)

data_stats = data_stats.iloc[:, 1:]

data_stats
```

Out[3]:

	Calories	Serving	Protein	Sugar	Sodium	Fat	Fiber	Popularity
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	199.996046	39.989139	2.974415	20.004080	74.982853	6.748226	0.490851	49.967119
std	1.127015	1.073182	1.012686	0.888751	1.007484	0.658202	0.292242	10.145767
min	195.897565	36.424752	-0.571292	16.925305	71.405873	5.162155	0.000487	-3.000000
25%	199.212548	39.272877	2.245751	19.409965	74.276512	6.279173	0.238509	42.948183
50%	200.001306	39.977855	2.979177	20.004798	75.000056	6.667726	0.489901	49.955385
75%	200.777167	40.734991	3.655002	20.588626	75.677900	7.170426	0.736093	57.153024
max	204.713237	43.561506	6.392027	22.922444	78.232301	9.498653	3.000000	86.469784
modus	195.897565	36.424752	-0.571292	16.925305	71.405873	5.162155	0.000487	-3.000000
variansi	1.270162	1.151719	1.025532	0.789878	1.015025	0.433230	0.085406	102.936586
range	8.815673	7.136754	6.963319	5.997139	6.826428	4.336498	2.999513	89.469784
IQR	1.564619	1.462113	1.409251	1.178660	1.401387	0.891253	0.497585	14.204841
skewness	0.041929	0.031222	0.089544	0.006416	-0.030713	0.537265	0.324107	-0.148562
kurtosis	0.198919	-0.119331	-0.099028	-0.048001	-0.025462	0.235368	1.396380	0.807764

In [4]:

```
# NON NUMERIK

# Kolom flavour
non_numeric_data = data.select_dtypes(exclude=[np.number])

unique_val = pd.DataFrame()

for column in non_numeric_data.columns:
    unique_counts = non_numeric_data[column].value_counts()
    proportions = unique_counts / unique_counts.sum()

    # Bikin dataframe baru buat storing
    column_stats = pd.DataFrame({
        'Value': unique_counts.index,
        'Count': unique_counts.values,
        'Proportion': proportions.values
    })

    column_stats['Column'] = column

    unique_val = pd.concat([unique_val, column_stats], axis=0, ignore_index=True)

unique_val = unique_val[['Column', 'Value', 'Count', 'Proportion']]

print(unique_val)
```

	Column	Value	Count	Proportion
0	Flavour	Chocolate	678	0.339
1	Flavour	Vanilla	664	0.332
2	Flavour	Strawberry	656	0.328
3	Flavour	undefined	2	0.001

Outlier Data

Terdapat dua outlier pada dataset tersebut yaitu:

1. Kolom Flavour merupakan outlier karena data pada kolom tersebut bukan bertipe numerik, sehingga tidak dapat dihitung statistik datanya. Cara untuk mengatasi hal tersebut adalah dengan menyeleksi kolom yang bertipe numerik saja untuk dilakukan analisis statistik data.
2. Kolom Flavour juga memiliki data yang berisi undefined sehingga membuat data menjadi tidak valid. Pada kolom tersebut seharusnya nilai atribut berisi "Vanilla", "Strawberry", atau "Chocolate". Cara untuk mengatasi hal tersebut adalah dengan menghapus baris yang memiliki atribut Flavour "undefined".
3. Terdapat nilai negatif pada kolom-kolom tertentu yang menyebabkan data bersifat kurang valid. Cara untuk mengatasi hal tersebut

adalah dengan menghapus baris yang mengandung nilai negatif pada suatu kolom. Handle kasus ini lebih cocok untuk menghapus baris daripada mengubah nilai negatif menjadi 0 karena bila diubah menjadi 0, ada kemungkinan statistik data yang dihasilkan tidak valid.

Hasil penanganan outlier tertera sebagai berikut.

```
In [5]: # Select data numerik
numeric_data = data.select_dtypes(include=[np.number])

# Hilangkan baris yang memiliki atribut numerik < 0
numeric_data_cleaned = numeric_data[(numeric_data >= 0).all(axis=1)]

# Select data non-numerik
non_numeric_data = data.select_dtypes(exclude=[np.number])

# Hilangkan baris yang memiliki atribut Flavour "undefined"
non_numeric_data_cleaned = non_numeric_data[~non_numeric_data.isin(['undefined']).any(axis=1)]

# Final cleaned data
data_cleaned = pd.concat([numeric_data_cleaned, non_numeric_data_cleaned], axis=1).dropna()
```

```
In [6]: data_stats = data_cleaned.describe(include=[np.number])
data_stats.loc['modus'] = data_cleaned.mode().loc[0]
data_stats.loc['std'] = data_cleaned.std(numeric_only=True)
data_stats.loc['variansi'] = data_cleaned.var(numeric_only=True)
data_stats.loc['range'] = data_stats.loc['max'] - data_stats.loc['min']
data_stats.loc['IQR'] = data_stats.loc['75%'] - data_stats.loc['25%']
data_stats.loc['skewness'] = data_cleaned.skew(numeric_only=True)
data_stats.loc['kurtosis'] = data_cleaned.kurt(numeric_only=True)

data_stats = data_stats.iloc[:, 1:]

data_stats
```

```
Out[6]:
```

	Calories	Serving	Protein	Sugar	Sodium	Fat	Fiber	Popularity
count	1993.000000	1993.000000	1993.000000	1993.000000	1993.000000	1993.000000	1993.000000	1993.000000
mean	199.996299	39.988716	2.977455	20.002050	74.982441	6.749323	0.491457	50.033018
std	1.126253	1.071637	1.004996	0.888232	1.007393	0.658485	0.292187	9.930054
min	195.897565	36.424752	0.089544	16.925305	71.405873	5.162155	0.000487	13.941228
25%	199.213774	39.274049	2.246481	19.407079	74.276814	6.279367	0.238627	43.040614
50%	199.998292	39.978320	2.979126	19.997650	75.000053	6.668204	0.490859	50.024621
75%	200.776433	40.734437	3.654718	20.587944	75.680876	7.172174	0.736135	57.152033
max	204.713237	43.561506	6.392027	22.922444	78.232301	9.498653	3.000000	86.469784
modus	195.897565	36.424752	0.089544	16.925305	71.405873	5.162155	0.000487	13.941228
variansi	1.268447	1.148405	1.010016	0.788957	1.014841	0.433602	0.085373	98.605965
range	8.815673	7.136754	6.302484	5.997139	6.826428	4.336498	2.999513	72.528556
IQR	1.562660	1.460388	1.408237	1.180865	1.404062	0.892807	0.497508	14.111419
skewness	0.045942	0.028907	0.130969	0.007070	-0.034556	0.537370	0.323935	0.021537
kurtosis	0.205563	-0.108116	-0.184298	-0.043897	-0.027662	0.231130	1.404239	-0.022240

```
In [7]: non_numeric_data = data_cleaned.select_dtypes(exclude=[np.number])

unique_val = pd.DataFrame()

for column in non_numeric_data.columns:
    unique_counts = non_numeric_data[column].value_counts()
    proportions = unique_counts / unique_counts.sum()

    # Bikin dataframe baru buat storing
    column_stats = pd.DataFrame({
        'Value': unique_counts.index,
        'Count': unique_counts.values,
        'Proportion': proportions.values
    })

    column_stats['Column'] = column

    unique_val = pd.concat([unique_val, column_stats], axis=0, ignore_index=True)

unique_val = unique_val[['Column', 'Value', 'Count', 'Proportion']]

print(unique_val)
```

	Column	Value	Count	Proportion
0	Flavour	Chocolate	677	0.339689
1	Flavour	Vanilla	661	0.331661
2	Flavour	Strawberry	655	0.328650

Visualisasi Plot Distribusi

```
In [8]: def dist_plot(attribute, num):
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))

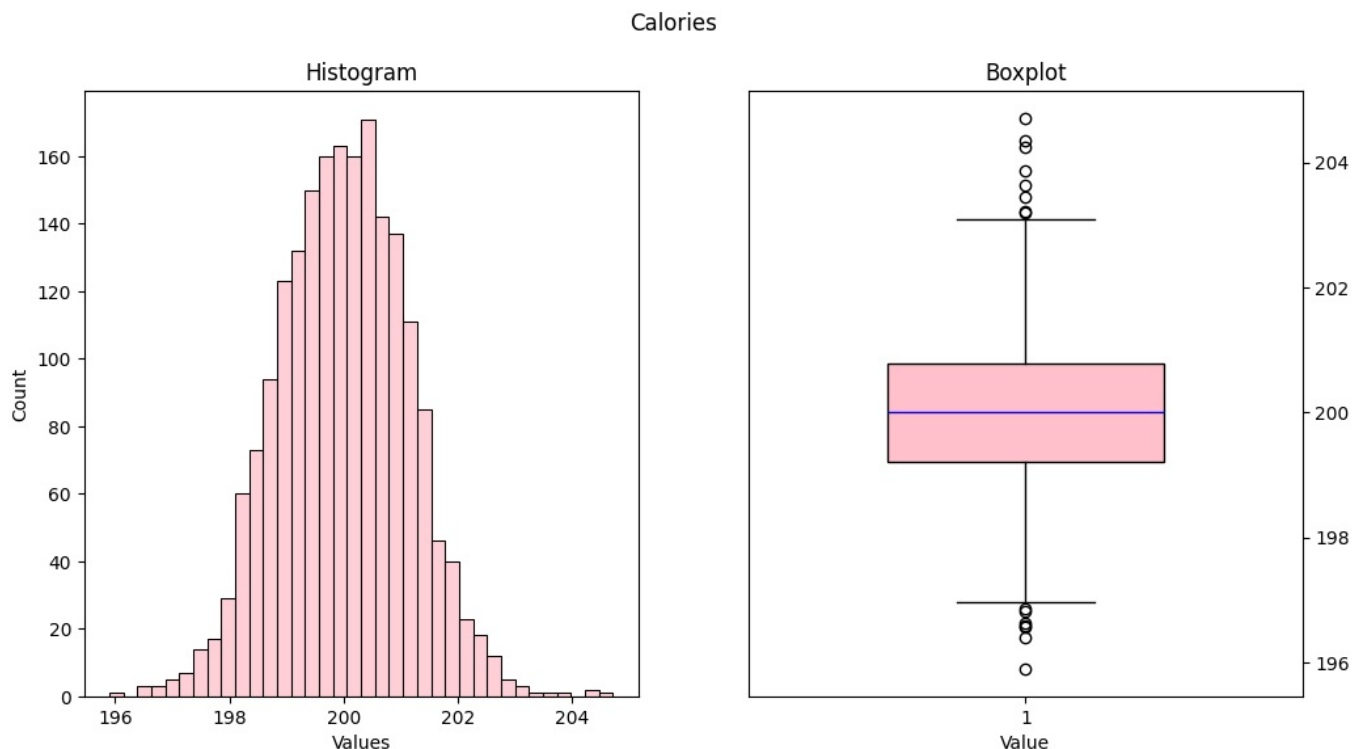
# histogram
sns.histplot(attribute, ax=ax1, color='pink')
ax1.set_title("Histogram")
ax1.set_ylabel("Count")
ax1.set_xlabel("Values")

if (num):
    # boxplot
    ax2.boxplot(attribute, vert=True, widths=0.5, patch_artist=True, boxprops=dict(facecolor='pink', color=
ax2.set_title('Boxplot')
ax2.set_xlabel('Value')
ax2.yaxis.tick_right()

# display plot
plt.suptitle(attribute.name)
plt.show()
```

1. Calories

```
In [9]: dist_plot(data_cleaned['Calories'], True)
```



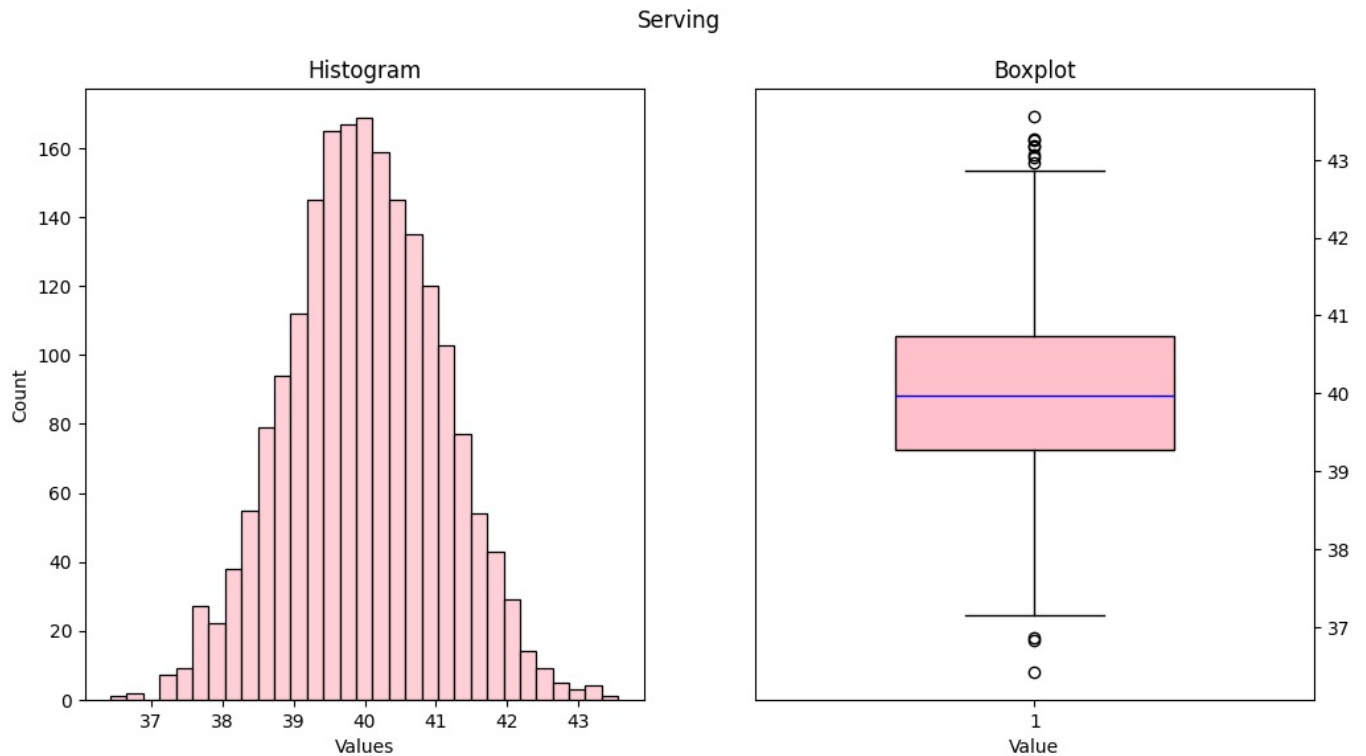
Terlihat pada histogram bahwa distribusi data **Calories** memiliki bentuk yang cukup simetris dan bentuknya mendekati distribusi normal. Hal ini ditandai dengan puncak yang tinggi di sekitar nilai 200 dan kurva yang menyebar secara merata di kedua sisi puncak tersebut. Dari histogram ini, kita dapat menyimpulkan bahwa sebagian besar data berkumpul di sekitar nilai tengah dan menunjukkan variasi yang terbatas dari nilai tengah tersebut, yang merupakan ciri khas **distribusi normal**.

Pada boxplot, garis median (Q2) berada hampir di tengah kotak interkuartil (IQR), yang menunjukkan bahwa separuh data berada di bawah nilai ini dan separuh lainnya di atasnya. Hal ini menunjukkan bahwa data cenderung terdistribusi secara simetris sekitar median. Selain itu, panjang whisker atas lebih panjang daripada whisker bawah yang mengindikasikan bahwa ada lebih banyak variasi atau rentang nilai di atas median dibandingkan di bawahnya. Outlier yang terlihat pada bagian atas boxplot menunjukkan adanya beberapa nilai yang jauh melebihi rentang nilai kebanyakan data, namun tidak cukup untuk mengubah kesimpulan bahwa distribusi tersebut adalah normal.

Nilai data **Calories** berkonsentrasi di sekitar 200, dengan distribusi yang cenderung normal dan beberapa nilai outlier.

2. Serving

```
In [10]: dist_plot(data_cleaned['Serving'], True)
```



Dari histogram, terlihat bahwa distribusi data **Serving** memiliki bentuk yang cukup simetris dan bentuknya mendekati distribusi normal. Hal ini ditandai dengan puncak yang tinggi di sekitar nilai 40 dan kurva yang menyebar secara merata di kedua sisi puncak tersebut. Distribusi ini, dengan sebagian besar data yang berkumpul di sekitar nilai tengah dan menunjukkan variasi yang terbatas dari nilai tengah, adalah ciri khas **distribusi normal**.

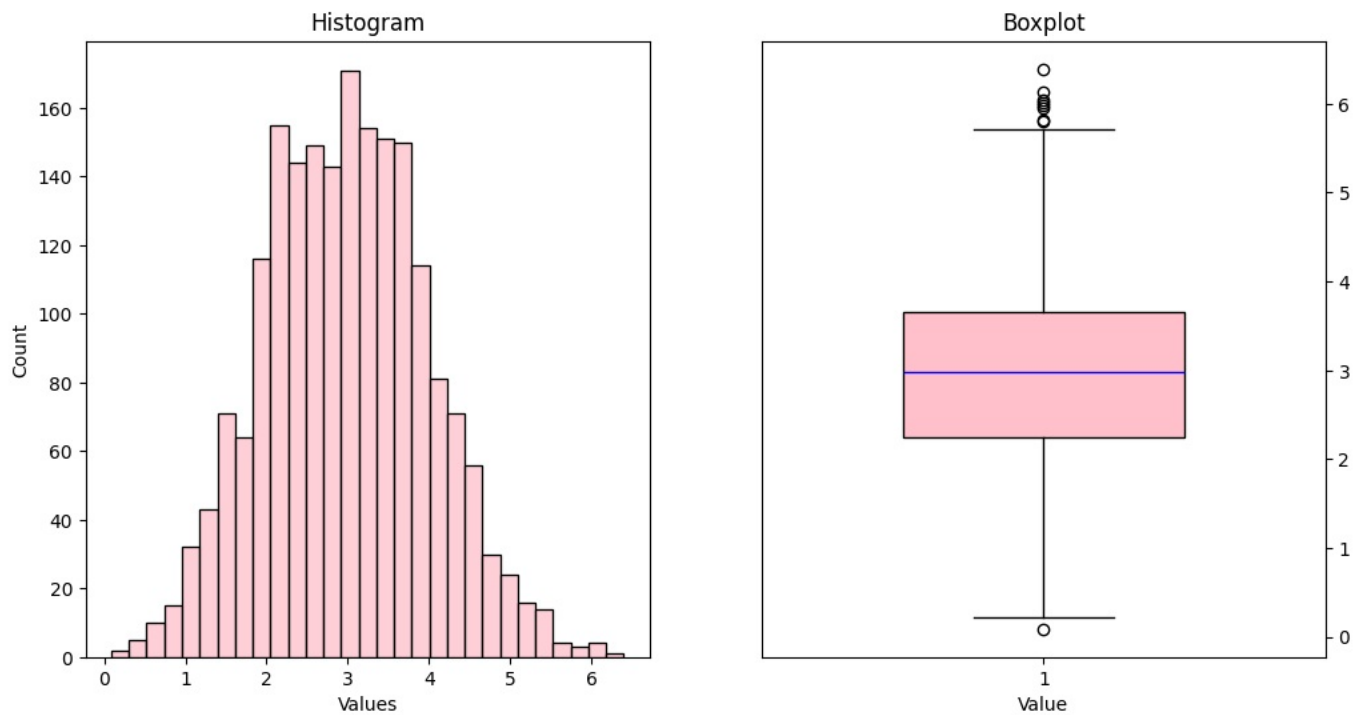
Pada boxplot, garis median (Q2) berada hampir di tengah kotak interkuartil (IQR). Hal ini menunjukkan bahwa separuh data berada di bawah nilai ini dan separuh lainnya di atasnya, mengindikasikan distribusi data yang simetris sekitar median. Whisker atas boxplot lebih panjang dibandingkan whisker bawah, mengindikasikan adanya variasi nilai yang lebih besar di atas median. Selain itu, keberadaan beberapa outlier menunjukkan adanya nilai yang jauh melebihi rentang nilai mayoritas data. Namun, jumlah outlier yang relatif sedikit ini tidak cukup untuk mengubah kesimpulan bahwa distribusi secara keseluruhan adalah normal.

Data **serving** berkisar terutama di sekitar 40 dengan distribusi yang cukup simetris dan normal.

3. Protein

```
In [11]: dist_plot(data_cleaned['Protein'], True)
```

Protein



Dari histogram yang diberikan untuk data **Protein**, kita dapat melihat bahwa distribusi data memiliki bentuk yang cukup simetris dan bentuknya mendekati distribusi normal. Puncak histogram ini berada di sekitar nilai 3, dengan kurva yang merata di kedua sisi, mencerminkan sebagian besar data berkumpul di sekitar nilai tengah tersebut. Kurva menunjukkan variasi terbatas dari nilai tengah, yang merupakan karakteristik dari **distribusi normal**.

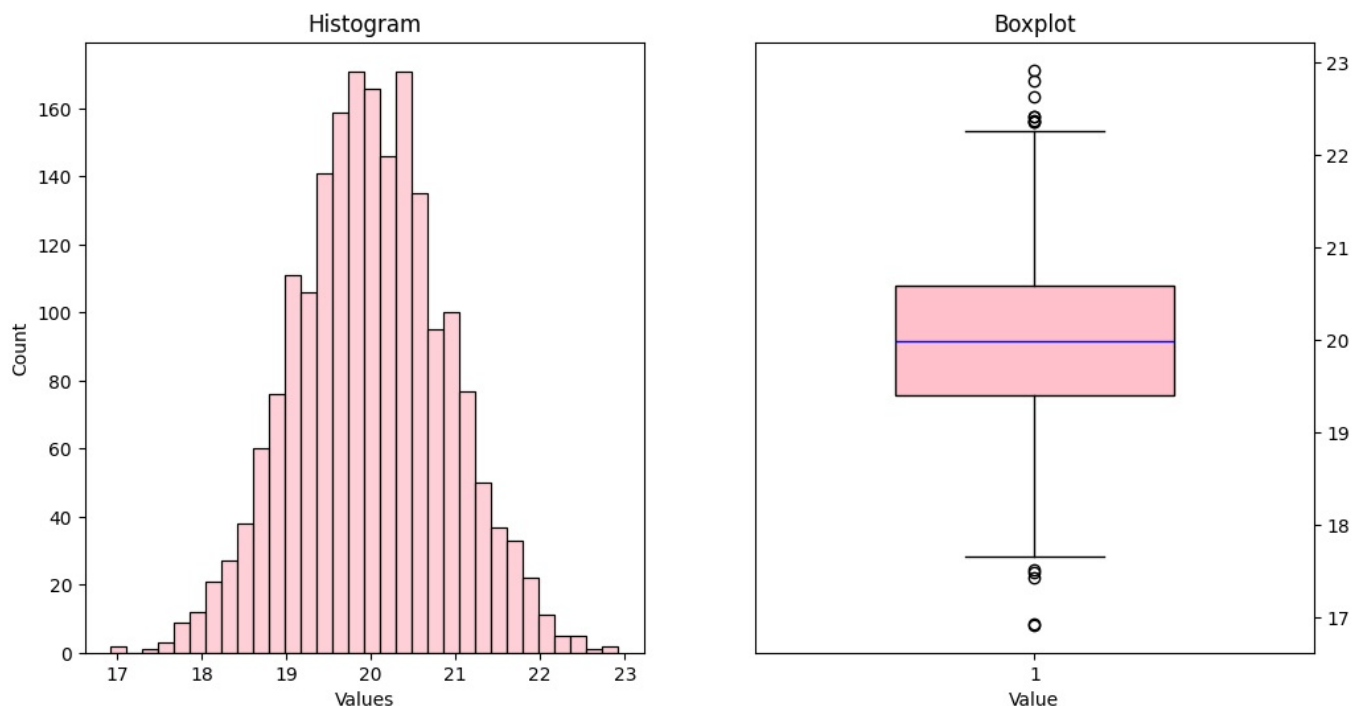
Pada boxplot, garis median (Q2) berada tepat di tengah kotak interkuartil (IQR). Penempatan median ini menunjukkan bahwa data terdistribusi secara simetris sekitar nilai median. Whisker atas dan bawah boxplot memiliki panjang yang hampir sama, yang menunjukkan distribusi nilai atas dan bawah yang serupa terhadap median. Terdapat beberapa outlier yang terlihat yang menunjukkan adanya beberapa nilai yang tidak tipikal dibandingkan dengan mayoritas data. Namun, jumlah outlier yang terbatas ini tidak cukup untuk mengubah kesimpulan bahwa distribusi secara keseluruhan adalah normal.

Distribusi data **Protein** cenderung bersifat normal dengan keberadaan beberapa outlier yang tidak mengubah karakteristik distribusi utama. Data protein pada umumnya berkisar antara 2 hingga 4, dengan nilai median 3.

4. Sugar

```
In [12]: dist_plot(data_cleaned['Sugar'], True)
```

Sugar



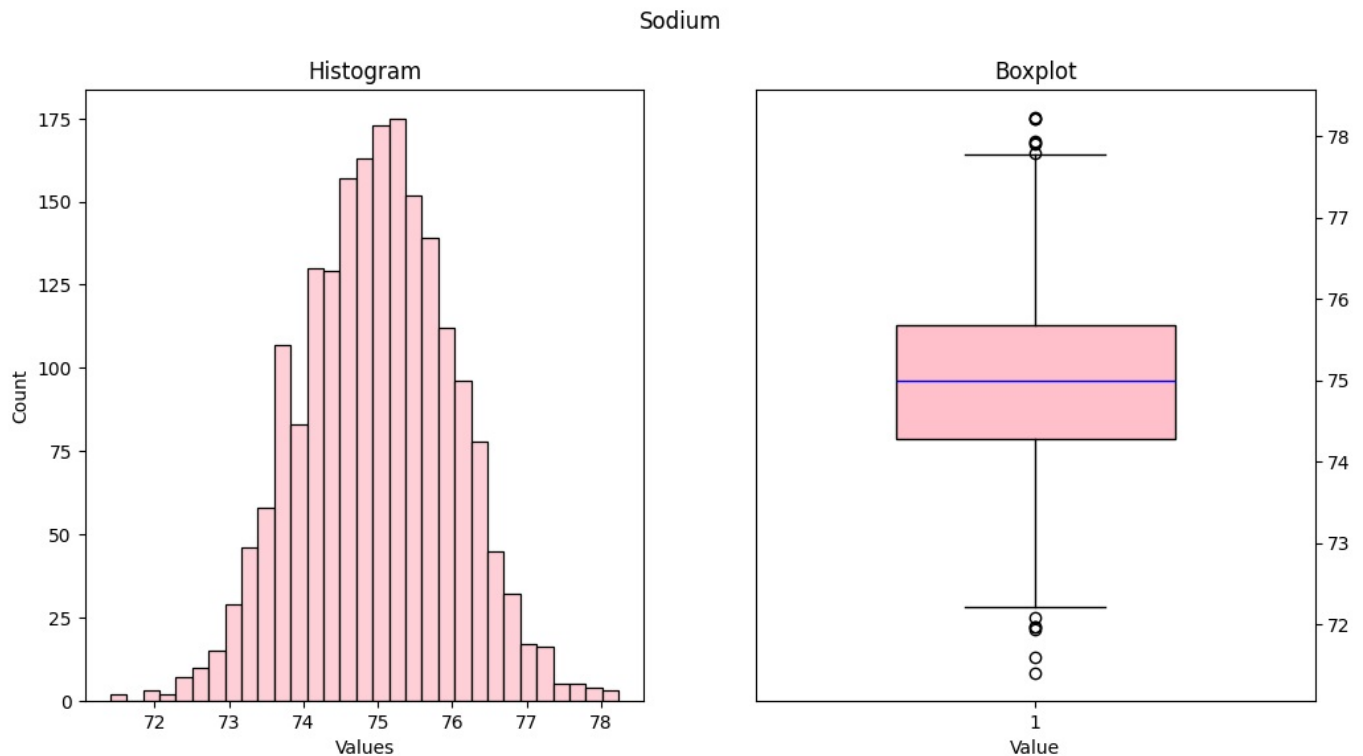
Dari histogram yang menunjukkan distribusi data **Sugar**, kita dapat melihat bahwa distribusi memiliki bentuk yang cukup simetris dengan puncak di sekitar nilai 20. Hal ini menandakan sebagian besar data terkonsentrasi di sekitar nilai tengah ini, dengan bentuk distribusi yang mendekati bentuk **distribusi normal** karena kurva yang merata di kedua sisi puncak.

Dalam boxplot, median (Q2) berada tepat di tengah kotak interkuartil (IQR). Hal ini menunjukkan bahwa data terdistribusi secara simetris sekitar nilai median, dengan whisker atas dan bawah yang membentang hampir sama panjang dari kotak, menunjukkan bahwa data atas dan bawah tersebar serupa dari median. Terdapat beberapa outlier yang terlihat yang menandakan adanya beberapa nilai yang tidak tipikal dibandingkan dengan mayoritas data. Namun, outlier ini tidak cukup banyak untuk mengganggu kesimpulan bahwa distribusi umumnya normal.

Distribusi data **Sugar** cenderung normal dengan beberapa outlier yang tidak mengubah karakteristik utama distribusi. Data umumnya berkisar antara 19 hingga 21 dengan median 20.

5. Sodium

```
In [13]: dist_plot(data_cleaned['Sodium'], True)
```



Dari histogram yang menampilkan distribusi data **Sodium**, dapat terlihat bahwa histogram memiliki puncak yang cukup simetris dengan titik tertinggi sekitar nilai 75. Ini menunjukkan bahwa kebanyakan data terkonsentrasi di sekitar nilai ini, dengan distribusi yang cukup simetris di kedua sisi puncak, yang menandakan distribusi yang mendekati **normal**.

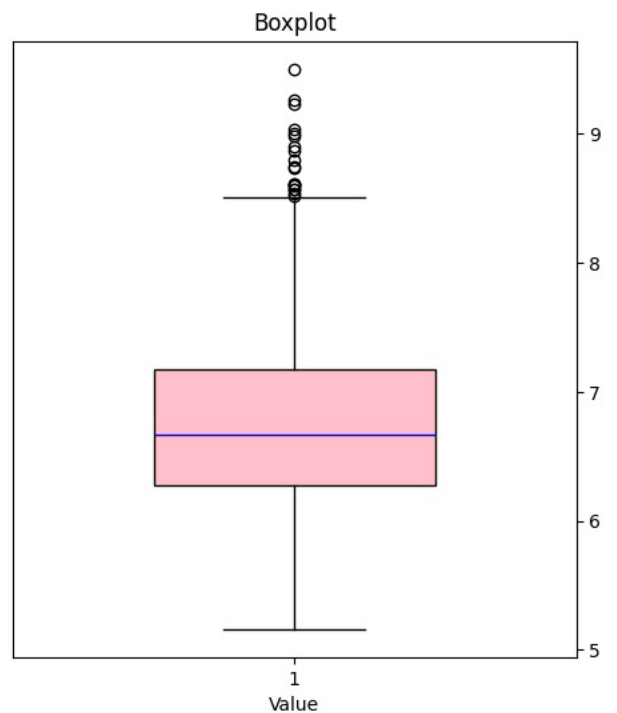
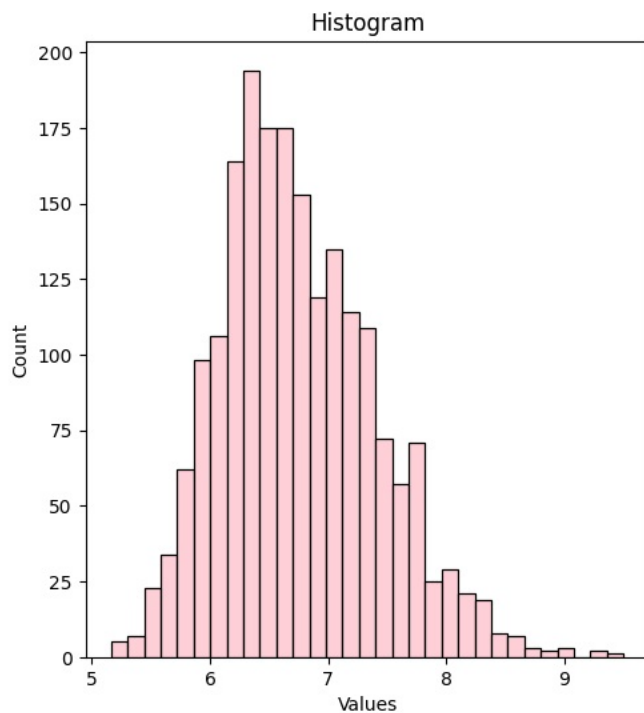
Pada boxplot, garis median (Q2) berada tepat di tengah kotak interkuartil (IQR). Posisi median ini menegaskan distribusi yang simetris sekitar nilai median tersebut. Whisker atas dan bawah boxplot menunjukkan variasi nilai yang serupa di atas dan di bawah median, dengan whisker yang membentang hampir sama panjang dari kotak, yang menunjukkan distribusi nilai atas dan bawah yang relatif seragam dari median. Terdapat beberapa outlier yang terlihat yang menandakan keberadaan beberapa nilai yang tidak sesuai dibandingkan dengan mayoritas data. Namun, jumlah outlier yang terbatas ini tidak cukup untuk mengganggu kesimpulan bahwa distribusi umumnya normal.

Distribusi data **Sodium** cenderung normal dengan beberapa outlier. Data secara umum berkisar antara 75 hingga 77, dengan median 76.

6. Fat

```
In [14]: dist_plot(data_cleaned['Fat'], True)
```


Fat



Dari histogram yang menampilkan data **Fat**, dapat disimpulkan bahwa distribusi data yang tidak sepenuhnya simetris dan terlihat skewed ke kanan (positif). Hal ini ditunjukkan oleh puncak sekitar nilai 6 dengan penurunan frekuensi yang lebih tajam pada nilai yang lebih tinggi daripada nilai yang lebih rendah. Histogram memiliki puncak utama di sekitar 0.5 dan secara bertahap menurun frekuensinya menuju nilai yang lebih tinggi.

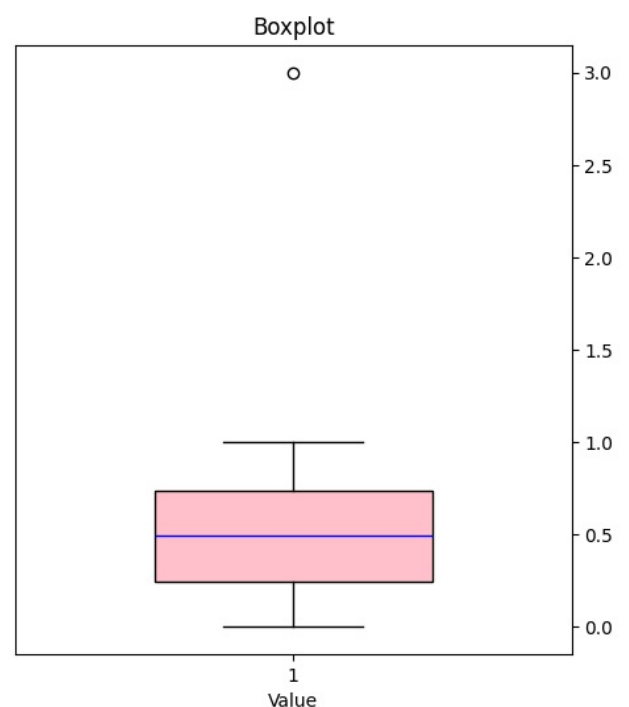
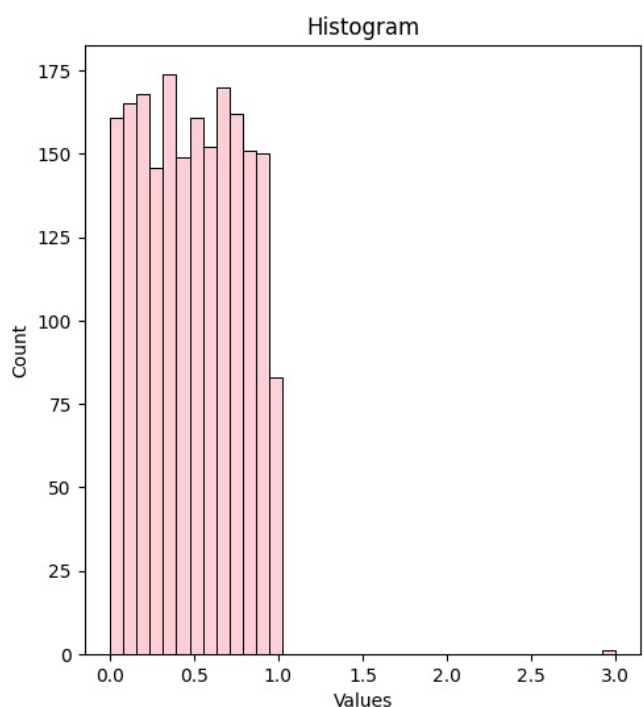
Posisi median menunjukkan distribusi yang tidak sepenuhnya simetris, dengan kotak interkuartil yang lebih pendek pada sisi bawah dan lebih panjang pada sisi atas, mengindikasikan distribusi yang cenderung memiliki ekor lebih panjang ke nilai yang lebih tinggi. Kehadiran beberapa outlier, terutama pada sisi atas, lebih jelas daripada yang biasanya ditemukan dalam distribusi normal. Outlier-outlier ini menunjukkan adanya nilai-nilai ekstrem yang mungkin mempengaruhi bentuk distribusi keseluruhan.

Data **Fat** menunjukkan skewness positif. Hal ini berarti bahwa ada lebih banyak distribusi nilai yang lebih tinggi yang melampaui nilai tengah atau median dari data, dengan frekuensi yang menurun secara bertahap menuju nilai yang lebih tinggi. Ini juga diindikasikan oleh lokasi median yang lebih rendah relatif terhadap keseluruhan rentang nilai dan oleh keberadaan beberapa nilai ekstrem di sisi lebih tinggi dari distribusi.

7. Fiber

```
In [15]: dist_plot(data_cleaned['Fiber'], True)
```

Fiber



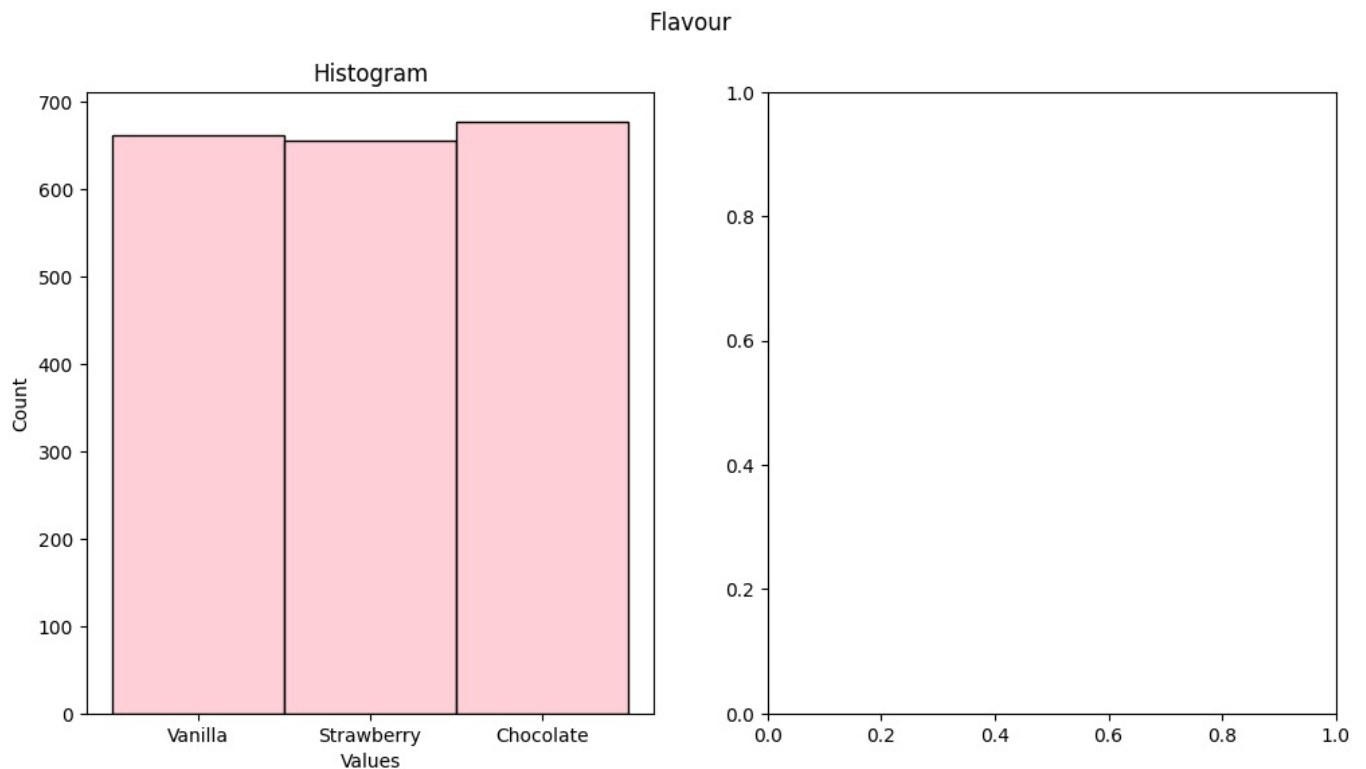
Dari histogram yang menampilkan data **Fiber**, dapat disimpulkan bahwa distribusi data yang tidak sepenuhnya simetris dan terlihat skewed ke kanan (positif). Histogram memiliki puncak utama di sekitar 0.5 dan secara bertahap menurun frekuensinya menuju nilai yang lebih tinggi. Tidak ada penumpukan yang signifikan pada nilai lebih rendah atau nol, menandakan bahwa hampir semua entri memiliki nilai serat yang terdeteksi.

Boxplot menunjukkan median yang berada dekat dengan 0.5. Kotak interkuartil (IQR) memiliki whisker yang memperluas jangkauan ke nilai yang lebih rendah dan lebih tinggi, namun tetap dalam jarak yang wajar dari IQR. Terdapat beberapa outlier yang ditandai di atas 2.5, yang menunjukkan keberadaan beberapa entri dengan kadar serat yang jauh lebih tinggi dibandingkan dengan sebagian besar data.

Data **Fiber** menunjukkan distribusi yang cenderung skewed ke kanan, dengan sebagian besar data berkonsentrasi pada nilai rendah tetapi dengan beberapa nilai yang jauh lebih tinggi sebagai outlier. Median dan IQR menunjukkan bahwa meskipun ada variasi, sebagian besar data cukup terkumpul di sekitar nilai rendah. Outlier menunjukkan bahwa sementara sebagian besar sampel memiliki kadar serat yang relatif rendah, ada beberapa sampel dengan kadar serat yang jauh lebih tinggi.

8. Flavour

```
In [16]: dist_plot(data_cleaned['Flavour'], False)
```

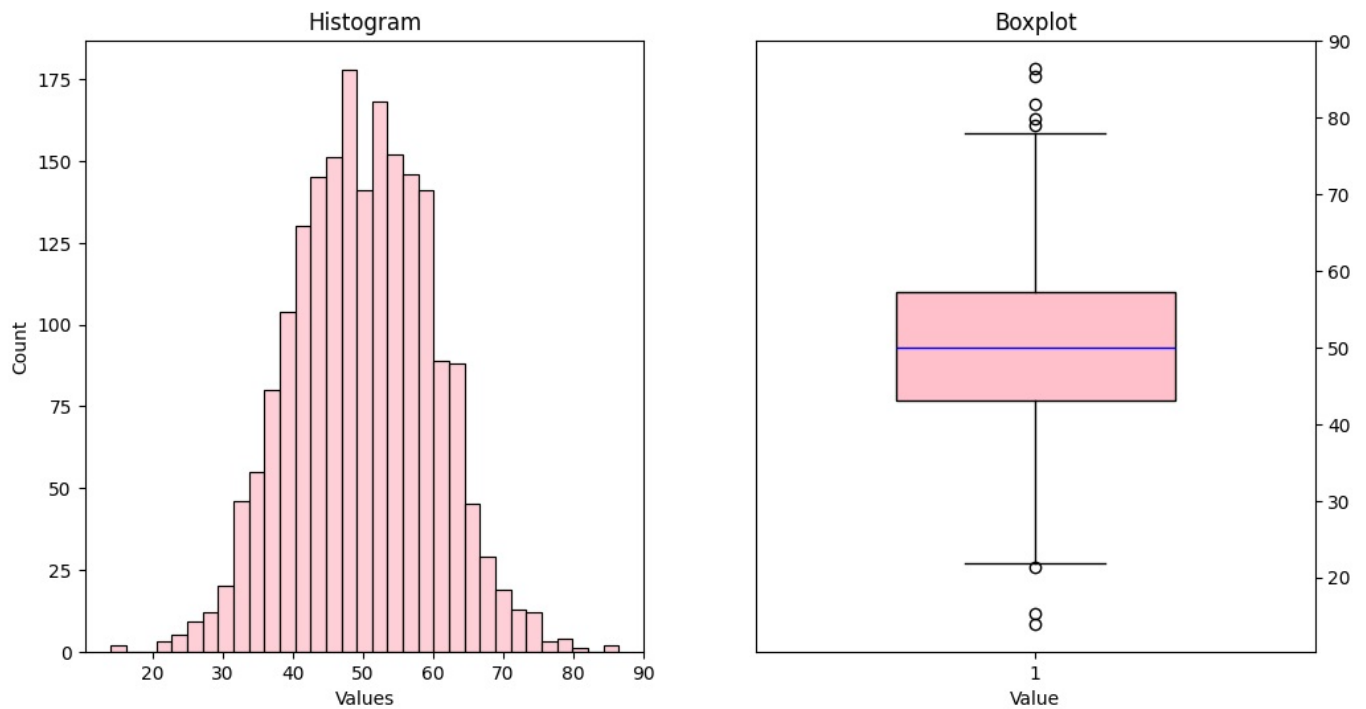


Sebelum dilakukan data cleansing, terdapat data undefined. Data cleansing nya dilakukan dengan menghapus data undefined tersebut. Berdasarkan histogram, ketiga kategori memiliki frekuensi yang hampir sama, yang menunjukkan distribusi yang merata. Hal ini bisa menunjukkan bahwa tidak ada bias signifikan terhadap rasa tertentu dalam pengumpulan data atau dalam preferensi yang dilaporkan oleh subjek data. Ketinggian bar yang hampir seragam untuk semua kategori menunjukkan konsistensi dalam jumlah entri per kategori.

9. Popularity

```
In [17]: dist_plot(data_cleaned['Popularity'], True)
```

Popularity



Dari analisis histogram dan boxplot, data **Popularity** menunjukkan karakteristik distribusi yang tidak normal. Adanya skewness, variasi yang luas dalam nilai-nilai data, dan outlier menandakan bahwa data memiliki ciri-ciri **distribusi tidak normal**.

Pada boxplot, adanya beberapa outlier di kedua sisi whiskers menunjukkan variasi yang ekstrem di luar apa yang diharapkan dalam distribusi normal. Outliers ini mungkin menunjukkan skewness atau keberadaan sub-grup dalam data.

Menentukan Setiap Kolom Numerik Berdistribusi Normal atau Tidak

```
In [18]: def check_normal_dist(atr, alpha=0.05):
    stat, p = normaltest(atr)
    display(Markdown(f'Nilai p = **{p}**'))
    if p > alpha:
        display(Markdown(f'Data **{atr.name}** Terdistribusi Normal'))
    else:
        display(Markdown(f'Data **{atr.name}** Tidak Terdistribusi Normal'))

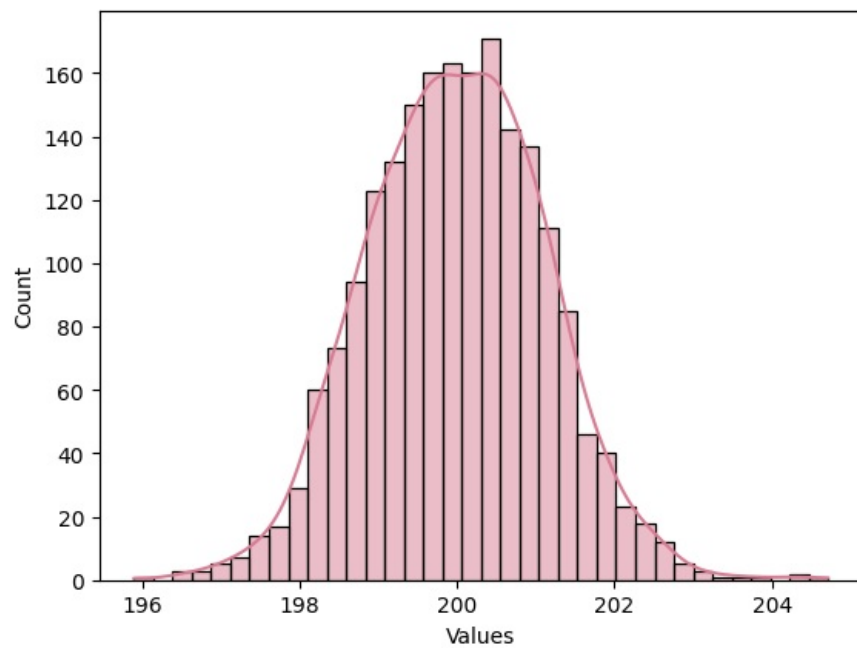
    sns.histplot(atr, kde=True, color='#d97c93')
    plt.xlabel('Values')
    plt.ylabel('Count')
    plt.show()
```

Calories

```
In [19]: check_normal_dist(data_cleaned['Calories'])
```

Nilai p = **0.14766995157903615**

Data **Calories** Terdistribusi Normal

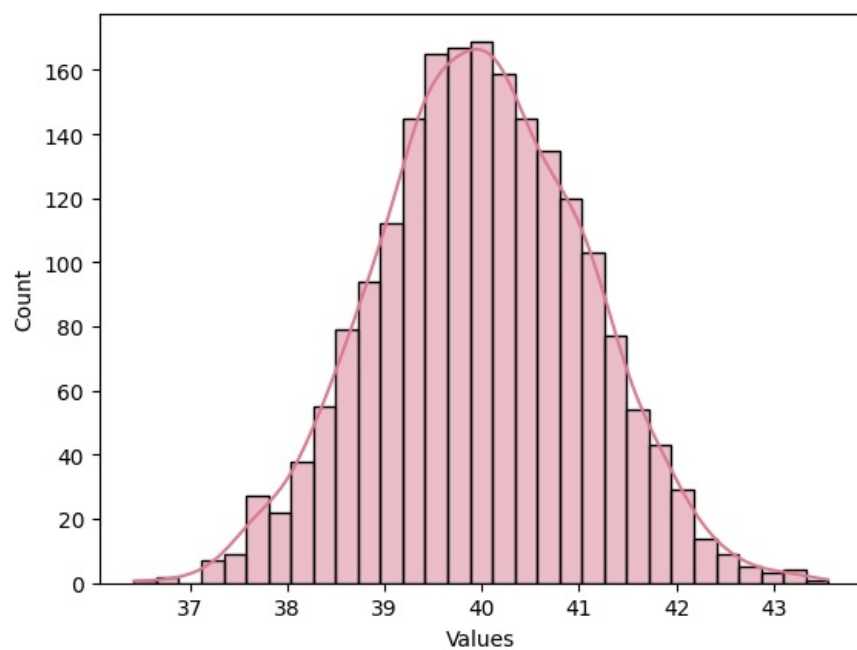


Serving

In [20]: `check_normal_dist(data_cleaned['Serving'])`

Nilai p = **0.5310783310002011**

Data **Serving** Terdistribusi Normal

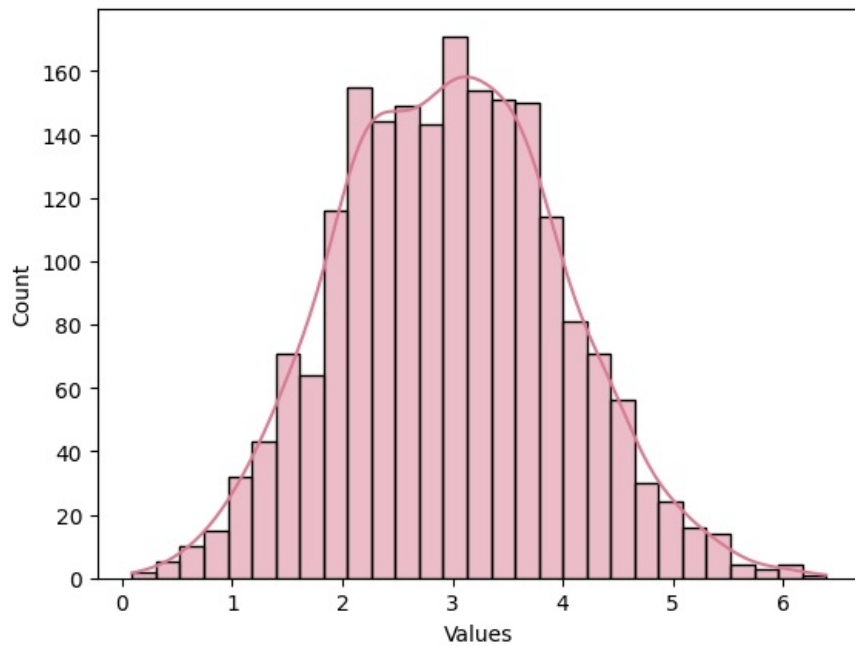


Protein

In [21]: `check_normal_dist(data_cleaned['Protein'])`

Nilai $p = 0.011394796506515483$

Data **Protein** Tidak Terdistribusi Normal



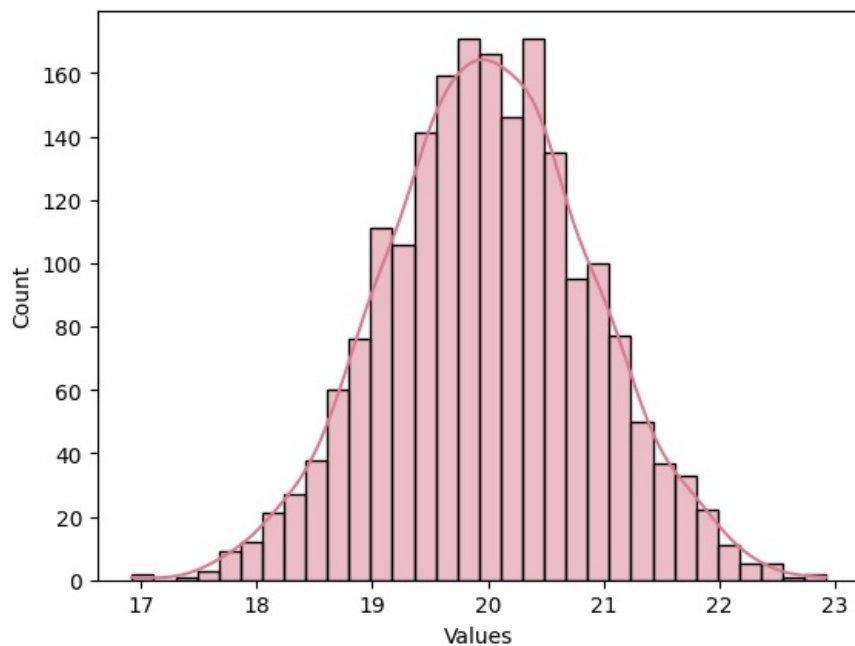
Bentuk distribusi **Protein** memiliki skewness positif dimana ekor di sisi kanan (positif) lebih panjang. Sebagian besar nilai data cenderung berkumpul di sisi kiri distribusi, dengan beberapa nilai yang lebih tinggi memperpanjang ekor ke kanan.

Sugar

```
In [22]: check_normal_dist(data_cleaned['Sugar'])
```

Nilai $p = 0.9303671406710812$

Data **Sugar** Terdistribusi Normal

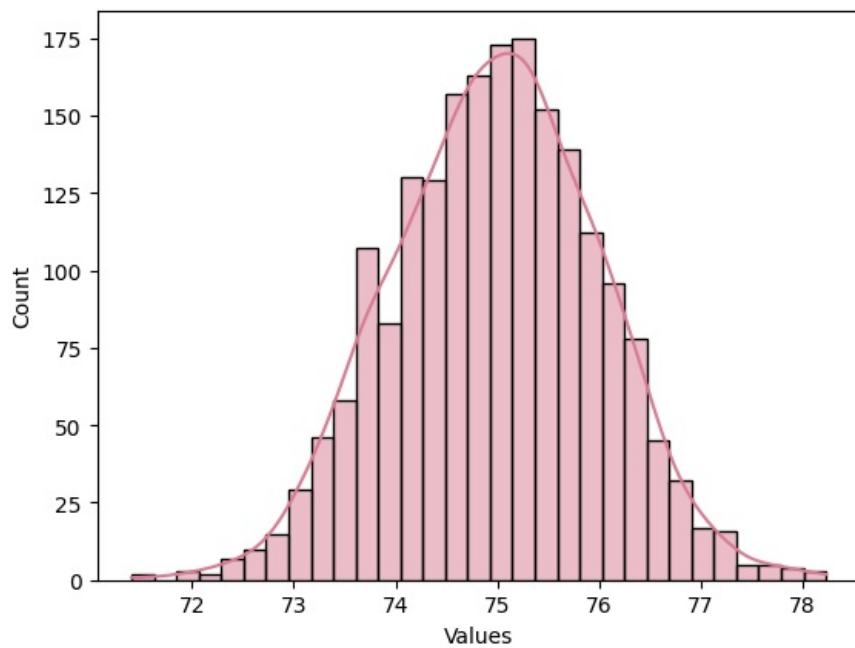


Sodium

```
In [23]: check_normal_dist(data_cleaned['Sodium'])
```

Nilai $p = 0.8025105855887815$

Data **Sodium** Terdistribusi Normal

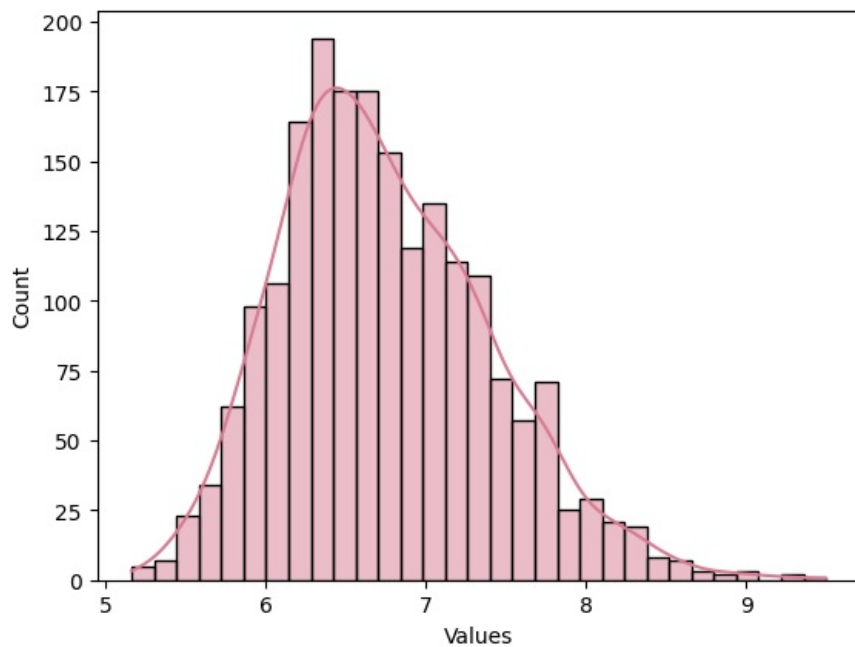


Fat

In [24]: `check_normal_dist(data_cleaned['Fat'])`

Nilai p = **4.616530804199059e-20**

Data **Fat** Tidak Terdistribusi Normal



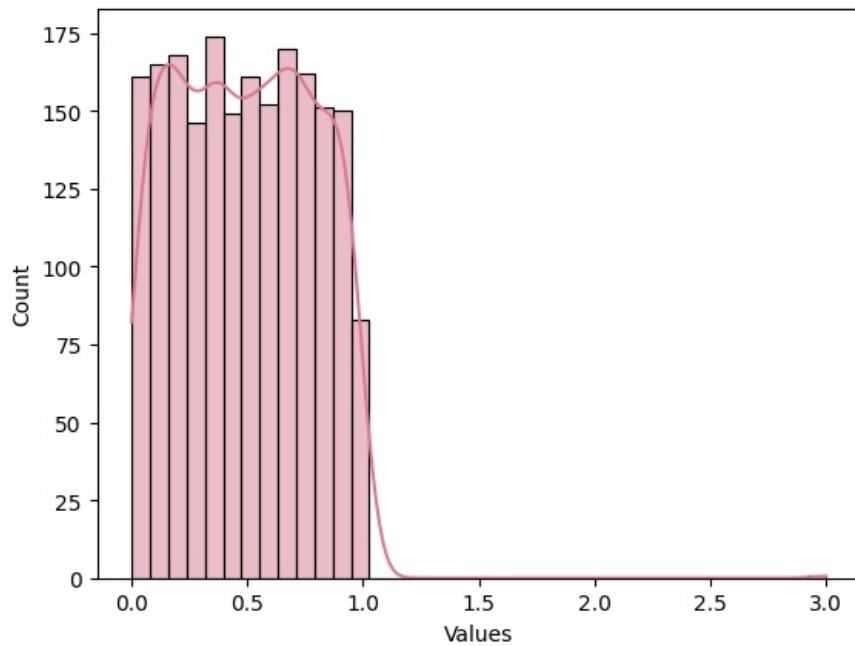
Bentuk distribusi **Fat** memiliki skewness positif dimana ekor di sisi kanan (positif) lebih panjang. Sebagian besar nilai data cenderung berkumpul di sisi kiri distribusi, dengan beberapa nilai yang lebih tinggi memperpanjang ekor ke kanan.

Fiber

In [25]: `check_normal_dist(data_cleaned['Fiber'])`

Nilai p = **1.5081211051373181e-21**

Data **Fiber** Tidak Terdistribusi Normal



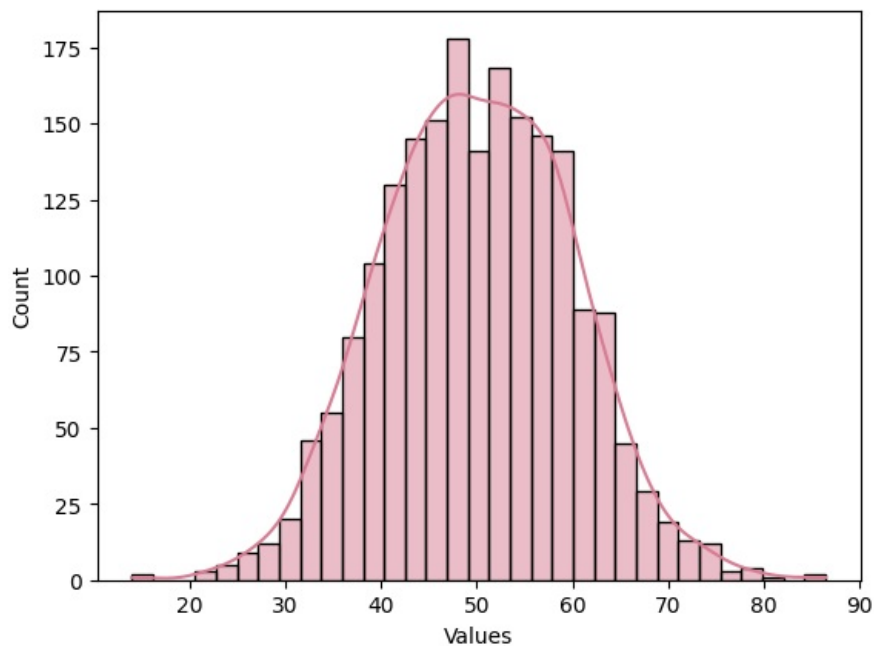
Bentuk distribusi **Fiber** memiliki skewness positif dimana ekor di sisi kanan (positif) lebih panjang. Sebagian besar nilai data cenderung berkumpul di sisi kiri distribusi, dengan beberapa nilai yang lebih tinggi memperpanjang ekor ke kanan.

Popularity

```
In [26]: check_normal_dist(data_cleaned['Popularity'])
```

Nilai p = **0.9147961218303744**

Data **Popularity** Terdistribusi Normal



Tes Hipotesis 1 Sampel

a. Apakah rata-rata nilai Sugar di bawah 25?

Langkah Uji:

1. $H_0: \mu = 25$
2. $H_1: \mu < 25$

3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil:

```
In [27]: # inisiasi
mu = 25
alpha = 0.05
mean_sugar = data_cleaned['Sugar'].mean() # pakai data yg udh diclean
std_sugar = data_cleaned['Sugar'].std()
n = data_cleaned['Sugar'].count()

# uji Z
Z0 = norm.ppf(1 - alpha)
Z = (mean_sugar - mu) / (std_sugar / np.sqrt(n))
p_val = norm.sf(abs(Z))

# hasil
print(f"Nilai Z = {Z}")
print(f"Nilai Z0 = {Z0}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")
if mean_sugar < mu: # jika rata-rata < mu, daerah penolakan berada di kiri distribusi
    if Z > Z0:
        print("Z > Z0")
        print("Maka gagal menolak H0")
    else:
        print("Z < Z0")
        print("Maka menolak H0")
else: # jika rata-rata > mu, daerah penolakan berada di kiri distribusi
    if Z > Z0:
        print("Z > Z0")
        print("Maka menolak H0")
    else:
        print("Z < Z0")
        print("Maka gagal menolak H0")

if p_val > alpha:
    print("p >  $\alpha$ ")
    print("Maka gagal menolak H0")
else:
    print("p <  $\alpha$ ")
    print("Maka menolak H0")
```

Nilai Z = -251.19960008622357
Nilai Z0 = 1.6448536269514722
Nilai p = 0.0

Disimpulkan bahwa
Z < Z0
Maka menolak H0
p < α
Maka menolak H0

Menolak H_0 , maka rata-rata nilai Sugar **di bawah 25**.

b. Apakah rata-rata Serving permen yang diproduksi tidak bernilai 40 gram?

Langkah Uji:

1. $H_0: \mu = 40$
2. $H_1: \mu \neq 40$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil:

```
In [28]: # inisiasi
mu = 40
alpha = 0.05
mean_serving = data_cleaned['Serving'].mean()
std_serving = data_cleaned['Serving'].std()
n = data_cleaned['Serving'].count()

# uji Z
Z0 = norm.ppf(1 - (alpha/2)) # dibagi 2 karena dua sisi
Z = (mean_serving - mu) / (std_serving / np.sqrt(n))
p_val = norm.sf(abs(Z))*2
```



```
# hasil
print(f"Nilai Z = {Z}")
print(f"Nilai Z0 = {Z0}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")
if Z < Z0 and Z > -Z0:
    print("Z < Z0 dan Z > -Z0")
    print("Maka gagal menolak H0")
else:
    print("Z > Z0 atau Z < -Z0")
    print("Maka menolak H0")

if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

Nilai Z = -0.47008086197356785

Nilai Z0 = 1.959963984540054

Nilai p = 0.6382972467733623

Disimpulkan bahwa

Z < Z0 dan Z > -Z0

Maka gagal menolak H0

p > α

Maka gagal menolak H0

Gagal menolak H_0 , maka rata-rata nilai Serving bernilai 40.

c. Apakah rata-rata Sodium untuk permen 20 baris terakhir tidak sama dengan 74?

Langkah Uji:

1. $H_0: \mu = 74$
2. $H_1: \mu \neq 74$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji T karena ukuran data < 30

Hasil:

```
In [29]: # inisiasi
mu = 74
alpha = 0.05
sodium_20 = data_cleaned['Sodium'].tail(20)
mean_20_sodium = sodium_20.mean()
std_20_sodium = sodium_20.std()
print(f"mean: {mean_20_sodium}")
print(f"std: {std_20_sodium}")
n = 20
df = n - 1

# uji T_val
T_kritis = stats.t.ppf(1 - alpha, df) # pakai derajat kebebasan untuk tabel t
T_val = (mean_20_sodium - mu) / (std_20_sodium / np.sqrt(n))
p_val = 1 - stats.t.cdf(T_val, df)

# hasil
print(f"Nilai T = {T_val}")
print(f"Nilai T kritis = {T_kritis}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")

if abs(T_val) <= T_kritis:
    print("|T| ≤ T Kritis")
    print("Maka gagal menolak H0")
else:
    print("|T| > T Kritis")
    print("Maka menolak H0")

if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

mean: 74.52736407351043
std: 0.9387872225600429
Nilai T = 2.5122240459234404
Nilai T kritis = 1.7291328115213673
Nilai p = 0.010593456946071833

Disimpulkan bahwa
 $|T| > T_{\text{Kritis}}$
Maka menolak H_0
 $p < \alpha$
Maka menolak H_0

Menolak H_0 , maka rata-rata nilai Serving **tidak bernilai 74**.

d. Apakah proporsi nilai Protein yang lebih besar dari 3 tidak sama dengan 60% dari total dataset?

Langkah Uji:

1. $H_0: p = 60$
2. $H_1: p \neq 60$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil:

```
In [30]: # inisiasi
alpha = 0.05
n_total = data_cleaned['Protein'].count() # total jumlah data
protein_3 = data_cleaned.loc[data_cleaned['Protein'] > 3, 'Protein'] # data yang proteinnya > 3
n_protein = protein_3.count()
mean_protein_3 = protein_3.mean()
std_protein_3 = protein_3.std()

# proporsi yang diharapkan
expected_prop = 0.60

p_hat = n_protein/n_total
q = 1 - expected_prop

# uji Z
Z = (p_hat - expected_prop) / np.sqrt(expected_prop * q / n_total)
Z0 = norm.ppf(1 - (alpha / 2))

p_val = 2 * norm.sf(abs(Z))

print(f"Nilai Z = {Z}")
print(f"Nilai Z0 = {Z0}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")

if Z < Z0 and Z > -Z0:
    print("Z < Z0 dan Z > -Z0")
    print("Maka gagal menolak H0")
else:
    print("Z > Z0 atau Z < -Z0")
    print("Maka menolak H0")

if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

Nilai Z = -9.82143635842022
Nilai Z0 = 1.959963984540054
Nilai p = 9.103703256986499e-23

Disimpulkan bahwa
 $Z > Z_0$ atau $Z < -Z_0$
Maka menolak H_0
 $p < \alpha$
Maka menolak H_0

Menolak H_0 , maka proporsi nilai Protein > 3 **tidak sama dengan 60%**.

Tes Hipotesis 2 Sampel

Perusahaan ingin membandingkan kualitas permen yang diproduksi pada paruh awal dan paruh akhir produksi. Hal ini dapat dilakukan dengan membagi 1 dataset menjadi 2 bagian yang sama panjang.

a. Apakah rata-rata Sugar dari permen yang diproduksi bernilai sama pada kedua kurun waktu!

Langkah Uji:

1. $H_0: \mu_{awal} = \mu_{akhir}$
2. $H_1: \mu_{awal} \neq \mu_{akhir}$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil Uji:

```
In [31]: n = len(data_cleaned)
alpha = 0.05

awal = data_cleaned.iloc[:n//2] # ambil setengah awal data
akhir = data_cleaned.iloc[n//2:] # ambil setengah akhir data

mean_awal = awal['Sugar'].mean()
mean_akhir = akhir['Sugar'].mean()

std_awal = awal['Sugar'].std()
std_akhir = akhir['Sugar'].std()

count_awal = awal['Sugar'].count()
count_akhir = akhir['Sugar'].count()

var_awal = awal['Sugar'].var()
var_akhir = akhir['Sugar'].var()

# uji Z
Z0 = norm.ppf(1 - (alpha/2))
Z = (mean_awal - mean_akhir) / np.sqrt((var_awal/count_awal) + (var_akhir/count_akhir))
p_val = norm.sf(abs(Z))*2

print(f"Nilai Z = {Z}")
print(f"Nilai Z0 = {Z0}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")
if Z < Z0 and Z > -Z0:
    print("Z < Z0 dan Z > -Z0")
    print("Maka gagal menolak H0")
else:
    print("Z > Z0 atau Z < -Z0")
    print("Maka menolak H0")

if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

Nilai Z = -1.422432944475768
Nilai Z0 = 1.959963984540054
Nilai p = 0.15490060495902155

Disimpulkan bahwa
Z < Z0 dan Z > -Z0
Maka gagal menolak H0
p > α
Maka gagal menolak H0

Gagal menolak H_0 , maka rata-rata Sugar dari permen yang diproduksi **bernilai sama** pada kedua kurun waktu.

b. Bandingkan rata-rata Protein dari permen pada paruh awal dan akhir.
Apakah rata-rata Protein pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.3 unit?

Langkah Uji:

1. $H_0: \mu_{awal} - \mu_{akhir} = 0.3$
2. $H_1: \mu_{awal} - \mu_{akhir} \neq 0.3$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil Uji:

```
In [32]: n = len(data_cleaned)
delta = 0.3

awal = data_cleaned.iloc[:n//2] # ambil setengah awal data
akhir = data_cleaned.iloc[n//2:] # ambil setengah akhir data

mean_awal = awal['Protein'].mean()
mean_akhir = akhir['Protein'].mean()

std_awal = awal['Protein'].std()
std_akhir = akhir['Protein'].std()

count_awal = awal['Protein'].count()
count_akhir = akhir['Protein'].count()

var_awal = awal['Protein'].var()
var_akhir = akhir['Protein'].var()

std_protein = data_cleaned['Protein'].std()

# uji Z
Z0 = norm.ppf(1 - (alpha/2))
Z = (mean_awal - mean_akhir - delta) / (std_protein / np.sqrt(n//2))
p_val = norm.sf(abs(Z))*2

print(f"Nilai Z = {Z}")
print(f"Nilai Z0 = {Z0}")
print(f"Nilai p = {p_val}")
print("\nDisimpulkan bahwa")
if Z < Z0 and Z > -Z0:
    print("Z < Z0 dan Z > -Z0")
    print("Maka gagal menolak H0")
else:
    print("Z > Z0 atau Z < -Z0")
    print("Maka menolak H0")

if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

Nilai Z = -8.562347817456981
 Nilai Z0 = 1.959963984540054
 Nilai p = 1.105919673596848e-17

Disimpulkan bahwa
 Z > Z0 atau Z < -Z0
 Maka menolak H0
 p < α
 Maka menolak H0

Menolak H_0 , tidak ada bukti signifikan yang mendukung bahwa rata-rata kadar Protein pada paruh awal lebih besar daripada paruh akhir sebesar 0.3 unit.

c. Periksalah apakah variansi dari Sodium dari permen sama pada paruh awal dan akhir!

Langkah Uji:

1. $H_0: \sigma_{awal}^2 = \sigma_{akhir}^2$
2. $H_1: \sigma_{awal}^2 \neq \sigma_{akhir}^2$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji f karena yang akan diuji adalah variansi

Hasil Uji:

```
In [37]: n = len(data_cleaned)
```

```

delta = 0.3
alpha = 0.05

awal = data_cleaned.iloc[:n//2] # ambil setengah awal data
akhir = data_cleaned.iloc[n//2:] # ambil setengah akhir data

mean_awal = awal['Sodium'].mean()
mean_akhir = akhir['Sodium'].mean()

std_awal = awal['Sodium'].std()
std_akhir = akhir['Sodium'].std()

count_awal = awal['Sodium'].count()
count_akhir = akhir['Sodium'].count()

var_awal = awal['Sodium'].var()
var_akhir = akhir['Sodium'].var()

dk_awal = count_awal - 1 # daerah kritis awal
dk_akhir = count_akhir - 1 # daerah kritis akhir

# Menggunakan uji f
F_alpha = f.ppf(1-alpha/2, dk_awal, dk_akhir)
F_lminalpha = f.ppf(alpha/2, dk_awal, dk_akhir)
F = var_awal / var_akhir
p_val = 2 * min(f.cdf(F, dk_awal, dk_akhir), 1 - f.cdf(F, dk_akhir, dk_awal))

# print(f"F(alpha){dk_awal, dk_akhir} = {F_alpha}")
# print(f"F(1-alpha){dk_awal, dk_akhir} = {F_lminalpha}")
print(f"Nilai f = {F}")
print(f"Nilai p = {p_val}")

print("\nDisimpulkan bahwa")
# Hasil Akhir
if F > F_lminalpha and F < F_alpha and p_val > alpha:
    print(f"F > F(1 - α){dk_awal, dk_akhir} dan F < F(α){dk_akhir, dk_awal} dan p > α")
    print("Maka gagal menolak H0")
else:
    print(f"F < F(1 - α){dk_awal, dk_akhir} atau F > F(α){dk_akhir, dk_awal} dan p < α")
    print("Maka menolak H0")

```

Nilai f = 1.051793517526536
Nilai p = 0.4258008066005443

Disimpulkan bahwa
 $F > F(1 - \alpha)(995, 996)$ dan $F < F(\alpha)(995, 996)$ dan $p > \alpha$
Maka gagal menolak H_0

Gagal menolak H_0 , tidak ada perbedaan yang signifikan antara variansi kadar Sodium pada paruh awal dan paruh akhir produksi.

d. Periksalah apakah proporsi Calories dari permen yang lebih dari 200 pada paruh awal lebih besar daripada proporsi di paruh akhir!

Langkah Uji:

1. $H_0: p_{awal} \leq p_{akhir}$
2. $H_1: p_{awal} > p_{akhir}$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

Hasil Uji:

```

In [34]: n = len(data_cleaned)
awal = data_cleaned.iloc[:n//2]
akhir = data_cleaned.iloc[n//2:]

# Menghitung jumlah permen dengan kalori lebih dari 200 pada kedua paruh
count_awal_200 = np.sum(awal['Calories'] > 200)
count_akhir_200 = np.sum(akhir['Calories'] > 200)

# Menghitung proporsi permen dengan kalori lebih dari 200 pada kedua paruh
count_awal = len(awal['Calories'])
count_akhir = len(akhir['Calories'])
prop_awal = count_awal_200 / count_awal
prop_akhir = count_akhir_200 / count_akhir

# Menghitung proporsi gabungan
prop = (count_awal_200 + count_akhir_200) / (count_awal + count_akhir)

```

```
# Menghitung standard error
SE = np.sqrt(prop * (1 - prop) * (1 / count_awal + 1 / count_akhir))

# Menghitung z-statistic
Z = (prop_awal - prop_akhir) / SE

# Menghitung p-value untuk uji satu sisi
p_val = norm.sf(Z)

print(f"Nilai Z = {Z}")
print(f"Nilai p = {p_val}")

print("\nDisimpulkan bahwa")
if p_val > alpha:
    print("p > α")
    print("Maka gagal menolak H0")
else:
    print("p < α")
    print("Maka menolak H0")
```

Nilai Z = 1.7247825717415601

Nilai p = 0.0422833318656981

Disimpulkan bahwa

$p < \alpha$

Maka menolak H_0

Menolak H_0 , terdapat bukti yang cukup untuk menyatakan bahwa proporsi permen dengan kalori lebih dari 200 pada paruh awal lebih besar daripada paruh akhir.