k2-if2220-13522032-13522074

May 24, 2024

1 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 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.

1.1 Author

NIM	Nama
13522032	Tazkia Nizami
13522074	Muhammad Naufal Aulia

1.2 Daftar Isi

2 Problem-Set

Seorang mahasiswa tingkat 3 sedang menjalani kerja praktek di sebuah perusahaan yang bergerak di industri buah-buahan. Perusahaan tersebut menghadapi kekhawatiran terkait kualitas buah pisang yang diberikan oleh pemasoknya. Sebagai bagian dari tugasnya, mahasiswa tersebut diberikan sebuah dataset yang berisikan informasi dan atribut-atribut yang terkait dengan buah pisang yang diberikan oleh pemasok. Mahasiswa diminta untuk melakukan analisis statistika terhadap dataset tersebut guna membantu perusahaan dalam memahami kualitas buah pisang yang mereka terima serta membantu dalam melakukan berbagai pengujian berbagai hipotesis

Diberikan sebuah data banana.csv yang dapat diakses pada utas berikut. banana.csv merupakan data metrik kualitas pisang dengan 11 kolom atribut sebagai berikut: 1. Acidity 2. Weight 3. Length 4. Appearance 5. Tannin 6. Ripeness 7. Sweetness 8. Country_of_Origin 9. Firmness 10. Grade 11. Price

2.0.1 Deskripsi Statistika

Menulis deskripsi statistika (Descriptive Statistics) dari semua kolom pada data yang bersifat numerik, terdiri dari mean, median, modus, standar deviasi, variansi, range, nilai minimum, maksimum, kuartil, IQR, skewness dan kurtosis. Boleh juga ditambahkan deskripsi lain.

2.0.2 Visualisasi Plot Distribusi

Membuat Visualisasi plot distribusi, dalam bentuk histogram dan boxplot untuk setiap kolom numerik. Berikan uraian penjelasan kondisi setiap kolom berdasarkan kedua plot tersebut.

2.0.3 Menentukan setiap kolom numerik berdistribusi normal atau tidak

Menentukan setiap kolom numerik berdistribusi normal atau tidak. Gunakan normality test yang dikaitkan dengan histogram plot.

2.0.4 Melakukan test hipotesis 1 sampel

- Perusahaan menerima beberapa keluhan bahwa buah pisang yang mereka terima akhir-akhir ini cukup asam. Dapatkah anda mengecek apakah rata-rata nilai b. Acidity di atas 6?
- Supplier menjanjikan bahwa rata-rata berat buah pisang adalah 150 gram. Pemilik mencurigai kebenaran hal ini. Apakah rata-rata buah pisang yang mereka kirim tidak bernilai 150 gram?
- Periksalah apakah rata-rata panjang buah pisang 10 baris terakhir tidak sama dengan 49!
- Apakah proporsi nilai Tannin yang lebih besar dari 8 tidak sama dengan 55% dari total dataset?

2.0.5 Melakukan test hipotesis 2 sampel,

Perusahaan ingin membandingkan kualitas buah yang diterima pada paruh awal dan paruh akhir kerjasama. Anda dapat melakukan ini dengan membagi 1 dataset menjadi 2 bagian yang sama panjang.

- Anda diminta untuk memeriksa apakah rata-rata acidity dari buah pisang yang disuplai bernilai sama pada kedua kurun waktu tersebut.
- Bandingkanlah rata-rata appearance pada bagian awal dan akhir. Apakah rata-rata appearance pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.1 unit?
- Apakah variansi dari panjang pisang yang dipasok suplier sama pada bagian awal dan akhir?

- Apakah proporsi berat pisang yang lebih dari 150 pada dataset awal lebih besar daripada proporsi di bagian dataset akhir?

3 Library and Data Loading

[425]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm, t, f, shapiro
import scipy.stats as stats
import numpy as np
import seaborn as sns
from IPython.display import display, Markdown
from scipy.stats import kstest, ttest_ind
from statsmodels.stats.proportion import proportions_ztest
!pip install fitter
from fitter import Fitter, get_common_distributions
from scipy.stats import norm, lognorm, expon, gamma, beta, weibull_min, pareto,␣
↪t, chi2, binom, poisson, uniform
data = pd.read_csv('banana.csv')
```

```
Collecting fitter
  Downloading fitter-1.7.0-py3-none-any.whl (26 kB)
Requirement already satisfied: click<9.0.0,>=8.1.6 in
/usr/local/lib/python3.10/dist-packages (from fitter) (8.1.7)
Requirement already satisfied: joblib<2.0.0,>=1.3.1 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.4.2)
Collecting loguru<0.8.0,>=0.7.2 (from fitter)
  Downloading loguru-0.7.2-py3-none-any.whl (62 kB)
                            62.5/62.5 kB
1.2 MB/s eta 0:00:00
Collecting matplotlib<4.0.0,>=3.7.2 (from fitter)
  Downloading
matplotlib-3.9.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3
MB)
                            8.3/8.3 MB
8.3 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0.0,>=1.20.0 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.25.2)
Requirement already satisfied: pandas<3.0.0,>=0.23.4 in
/usr/local/lib/python3.10/dist-packages (from fitter) (2.0.3)
Collecting rich-click<2.0.0,>=1.7.2 (from fitter)
  Downloading rich_click-1.8.2-py3-none-any.whl (34 kB)
Requirement already satisfied: scipy<2.0.0,>=0.18.0 in
/usr/local/lib/python3.10/dist-packages (from fitter) (1.11.4)
Requirement already satisfied: tqdm<5.0.0,>=4.65.1 in
/usr/local/lib/python3.10/dist-packages (from fitter) (4.66.4)
Requirement already satisfied: contourpy>=1.0.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4.0.0,>=3.7.2->fitter) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(24.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<4.0.0,>=3.7.2->fitter) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<4.0.0,>=3.7.2->fitter)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0.0,>=0.23.4->fitter) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0.0,>=0.23.4->fitter) (2024.1)
Requirement already satisfied: rich>=10.7 in /usr/local/lib/python3.10/dist-
packages (from rich-click<2.0.0,>=1.7.2->fitter) (13.7.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from rich-click<2.0.0,>=1.7.2->fitter)
(4.11.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib<4.0.0,>=3.7.2->fitter) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.7->rich-
click<2.0.0,>=1.7.2->fitter) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.7->rich-
click<2.0.0,>=1.7.2->fitter) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich>=10.7->rich-
click<2.0.0,>=1.7.2->fitter) (0.1.2)
Installing collected packages: loguru, matplotlib, rich-click, fitter
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
Successfully installed fitter-1.7.0 loguru-0.7.2 matplotlib-3.9.0 rich-
click-1.8.2
```

4 1. Deskripsi Statistika

4.0.1 Penjelasan Atribut :

- Count : Jumlah data
- Mean : Rata-rata, nilai rata-rata dari semua data
- Std : Standar deviasi, nilai rata-rata dari selisih antara setiap data dengan rata-rata
- Min : Nilai minimum, nilai terkecil dari semua data
- 25% : Nilai kuartil pertama, nilai yang membagi data menjadi 4 bagian yang sama besar, dan 25% data berada di bawah nilai ini
- 50% : Nilai kuartil kedua, nilai yang membagi data menjadi 2 bagian yang sama besar, dan 50% data berada di bawah nilai ini
- 75% : Nilai kuartil ketiga, nilai yang membagi data menjadi 4 bagian yang sama besar, dan 75% data berada di bawah nilai ini
- Max : Nilai maksimum, nilai terbesar dari semua data
- Modus : Nilai yang paling sering muncul
- Variansi : Nilai rata-rata dari selisih kuadrat antara setiap data dengan rata-rata
- Skewness : Nilai yang menunjukkan seberapa simetris distribusi data
- Kurtosis : Nilai yang menunjukkan seberapa tajam puncak distribusi data

[426]:
```python
# Deskripsi statistik data numerik
# Drop kolom pertama (ID)
data = data.iloc[:, 1:]

# Filter untuk ambil data numerik
numerical_columns = data.select_dtypes(include=[np.number])

numerical_stats = numerical_columns.describe()

numerical_stats.loc['modus'] = data.mode().iloc[0]  # Mode for all columns
numerical_stats.loc['std'] = numerical_columns.std()
numerical_stats.loc['variansi'] = numerical_columns.var()
numerical_stats.loc['range'] = numerical_stats.loc['max'] - numerical_stats.
↪loc['min']
numerical_stats.loc['IQR'] = numerical_stats.loc['75%'] - numerical_stats.
↪loc['25%']
numerical_stats.loc['skewness'] = numerical_columns.skew()
numerical_stats.loc['kurtosis'] = numerical_columns.kurt()
numerical_stats
```

[426]:
```
          Acidity       Weight       Length   Appearance       Tannin  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean      8.014830   150.011549    49.950434     4.965595     7.965435
std       1.105781     1.194980     0.894599     1.014863     1.217188
min       4.456118   146.060922    46.418052     1.775864     4.291274
25%       7.259942   149.227116    49.346508     4.258210     7.167241
50%       8.005347   150.022865    49.923682     4.979534     8.022448
75%       8.758361   150.827613    50.572027     5.653875     8.792184
```

```
max        11.418636   154.070370   53.065151    8.233968   12.416177
modus       4.456118   146.060922   46.418052    1.775864    4.291274
variansi    1.222752     1.427977    0.800307    1.029946    1.481546
range       6.962518     8.009448    6.647099    6.458104    8.124904
IQR         1.498418     1.600497    1.225519    1.395665    1.624943
skewness    0.056793    -0.084767    0.026878   -0.035389   -0.066152
kurtosis   -0.147134     0.024967   -0.053550   -0.002189    0.066349

              Ripeness    Sweetness     Firmness          Price
count      2000.000000  2000.000000  2000.000000    2000.000000
mean          6.743434     6.226319     0.507790   19969.669241
std           0.680320     0.662980     0.292226     777.347464
min           4.862560     3.033193     0.000254      -1.000000
25%           6.268258     5.808028     0.254351   19953.093529
50%           6.667618     6.312819     0.515483   19999.508312
75%           7.164813     6.714660     0.758786   20047.301949
max           9.482066     7.678689     2.000000   20281.431062
modus         4.862560     3.033193     0.000254       0.000000
variansi      0.462836     0.439543     0.085396  604269.080280
range         4.619506     4.645496     1.999746   20282.431062
IQR           0.896555     0.906632     0.504436      94.208419
skewness      0.495597    -0.663692     0.024873     -25.469237
kurtosis      0.278203     0.495115    -0.904900     652.633188
```

[427]:

```python
# Deskripsi statistik data string (kategorikal)
categorical_stats = pd.DataFrame(columns=['unique_values', 'proporsi'])

# Unique values dan proporsi untuk Country_of_Origin
country_counts = data['Country_of_Origin'].value_counts(normalize=True)
categorical_stats.loc['Country_of_Origin'] = [country_counts.index.tolist(),␣
↪country_counts.values.tolist()]

# Unique values dan proporsi untuk Grade
grade_counts = data['Grade'].value_counts(normalize=True)
categorical_stats.loc['Grade'] = [grade_counts.index.tolist(), grade_counts.
↪values.tolist()]

categorical_stats.T
```

[427]:

```
                                          Country_of_Origin  \
unique_values  [Ecuador, Costa Rica, Colombia, undefined]
proporsi                   [0.5605, 0.285, 0.153, 0.0015]

                                  Grade
unique_values               [A, C, B]
proporsi       [0.3415, 0.339, 0.3195]
```

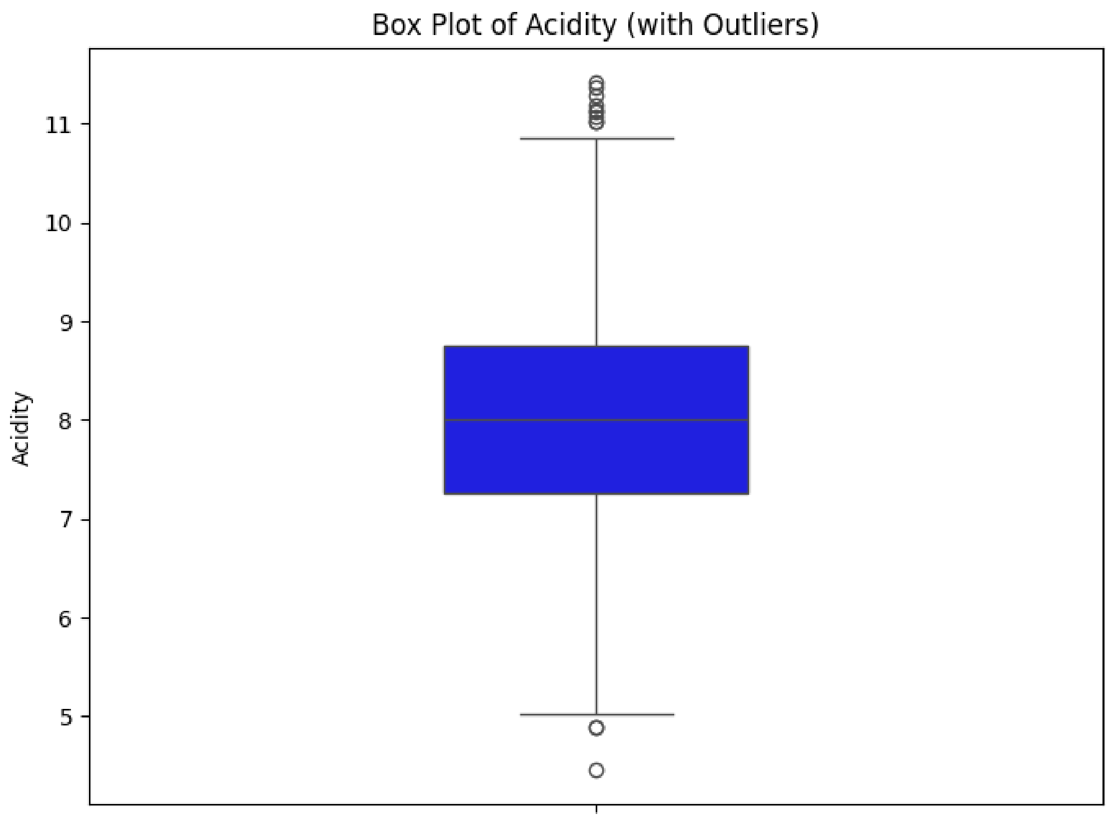
5 2. Penanganan Outlier

Fungsi untuk menampilkan outlier pada pada atribut di dataset

[428]:
```
def show_outlier(df, attribute):
    Q1 = df[attribute].quantile(0.25)
    Q3 = df[attribute].quantile(0.75)

    # Hitung IRQ
    IQR = Q3 - Q1

    # Lower and upper bounds untuk mendeteksi outlier
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Deteksi
    outliers = df[(df[attribute] < lower_bound) | (df[attribute] > upper_bound)]

    # Plot box plot dengan outliers
    plt.figure(figsize=(8, 6))
    sns.boxplot(y=df[attribute], color='blue', width=0.3)
    plt.title('Box Plot of ' + attribute + ' (with Outliers)')
    plt.show()

    # Informasi mengenai outlier
    print("Number of outliers:", len(outliers))
    if len(outliers) > 0:
        print("Example of outliers:")
        print(outliers.head())

    return outliers
```

5.0.1 1. Acidity

[429]:
```
outliers_acidity = show_outlier(data, 'Acidity')
```

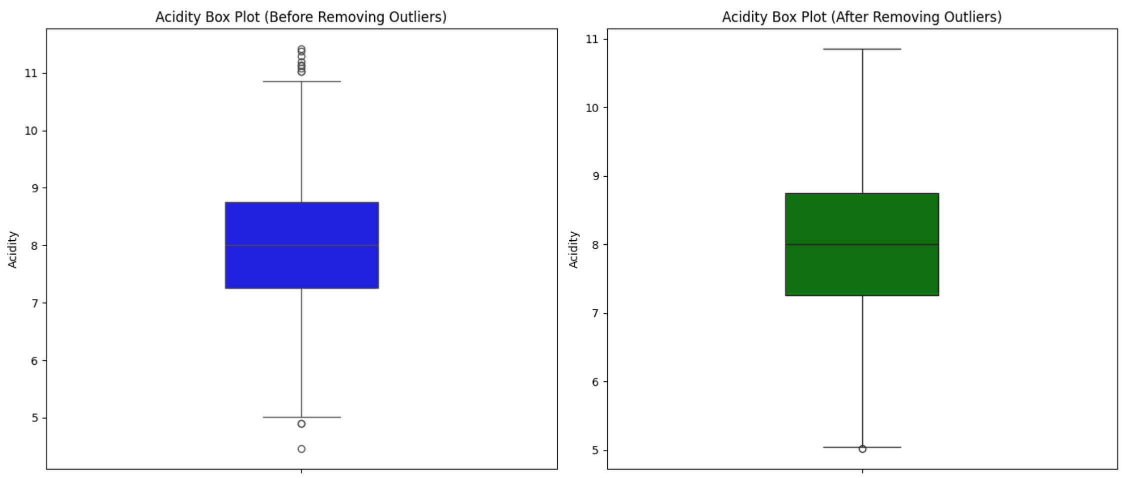



```
Number of outliers: 12
Example of outliers:
       Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
148  11.191852  150.256022  50.119257    4.942644  9.250504  5.679064
209  11.119288  149.503955  49.116418    5.564729  7.774533  5.934182
279  11.137342  151.153362  48.945558    7.169523  7.850270  6.429438
289  11.024219  149.503132  49.721594    4.516512  8.799761  6.041898
345  11.079811  150.694486  50.443629    6.271014  6.655897  7.638313

     Sweetness Country_of_Origin  Firmness Grade         Price
148   5.451147           Ecuador  0.404192     A  19988.979717
209   6.283469           Ecuador  0.925808     B  20087.416779
279   5.799971           Ecuador  0.779323     B  19972.146872
289   6.530149           Ecuador  0.821839     C  19889.974442
345   7.119831           Ecuador  0.371698     A  19991.907418
```

Dari hasil tersebut, diketahui terdapat sebanyak 12 outlier pada atribut Acidity. Outlier terletak lebih banyak di atas kuartil ketiga. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[430]:

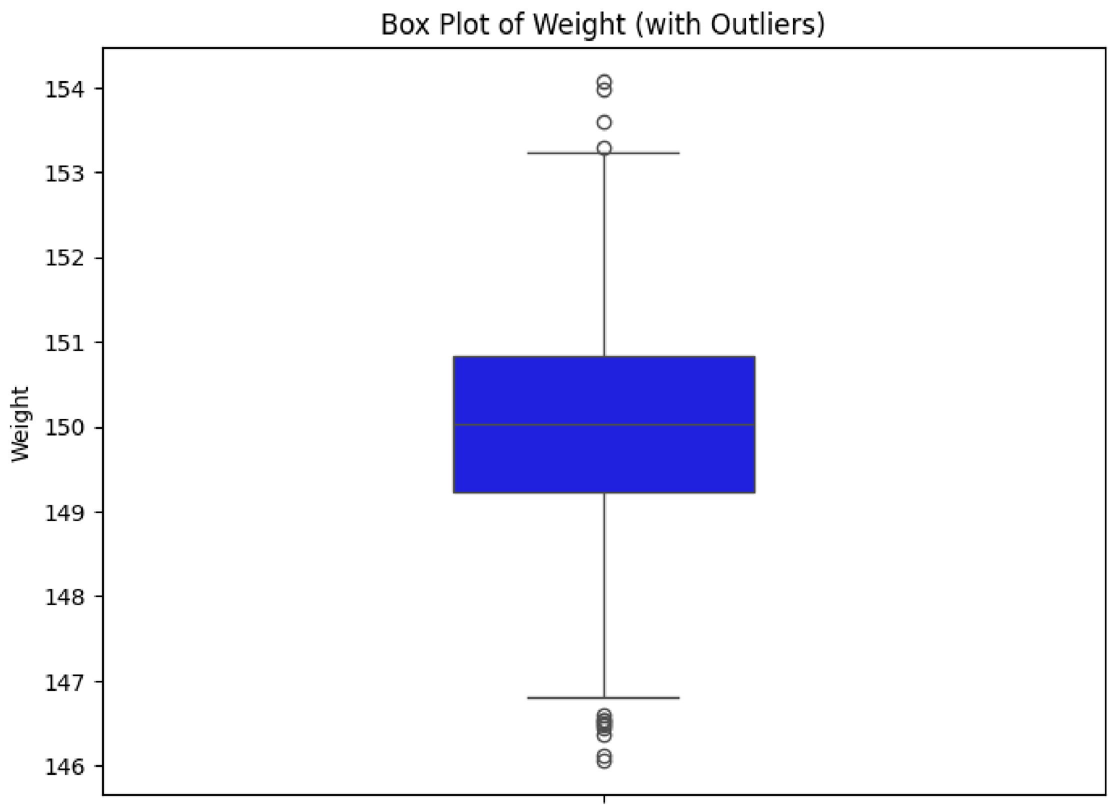
```
cleaned_acidity = data.drop(outliers_acidity.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=data['Acidity'], color='blue', width=0.3)
plt.title('Acidity Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_acidity['Acidity'], color='green', width=0.3)
plt.title('Acidity Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.2 2. Weight

[431]:

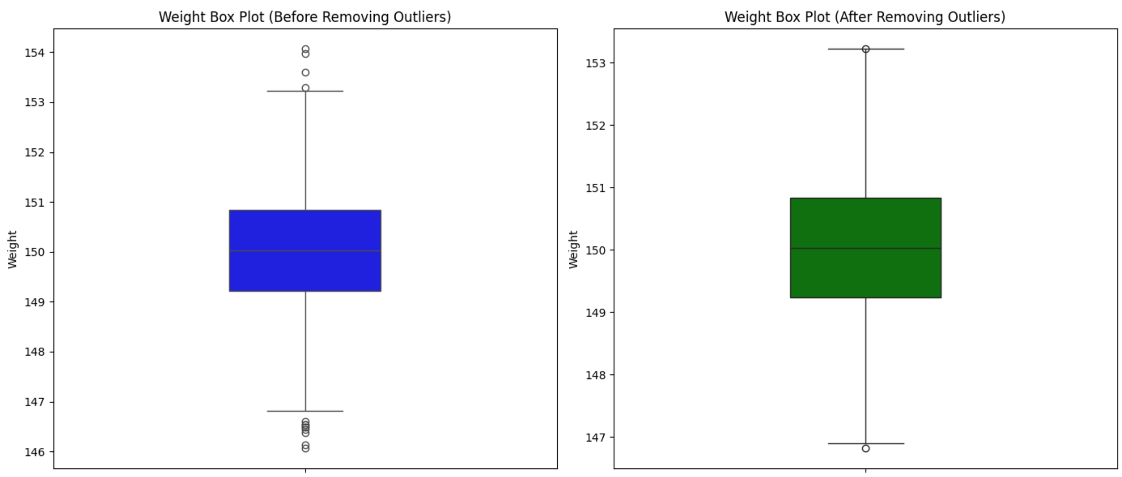
```
outliers_weight = show_outlier(cleaned_acidity, 'Weight')
```




```
Number of outliers: 13
Example of outliers:
      Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
44   6.755795  146.535963  48.072988    5.359031  6.347397  6.234711
357  8.904595  153.970493  50.321849    3.329714  5.152883  6.798645
386  8.704334  146.376184  48.614522    4.324628  7.417205  5.239580
658  7.662058  146.490788  48.536914    6.920799  8.536634  5.929132
677  7.679393  146.444130  49.816581    2.424479  7.173284  6.323959

     Sweetness Country_of_Origin  Firmness Grade         Price
44    6.131756           Ecuador  0.965369     A  19964.612816
357   6.631835           Ecuador  0.638855     A  19923.929065
386   6.348539           Ecuador  0.356976     B  19982.304077
658   6.180033           Ecuador  0.290999     B  20069.369263
677   6.081852           Ecuador  0.666101     A  19965.606599
```

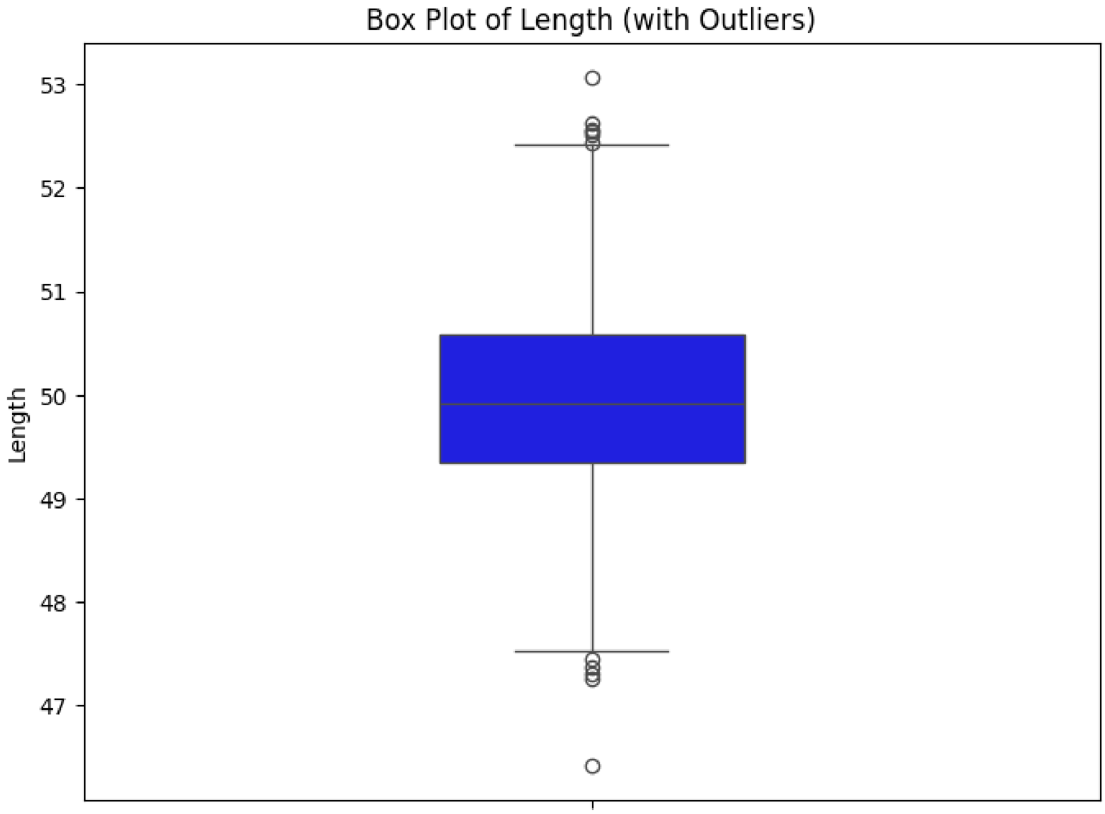
Dari hasil tersebut, diketahui terdapat sebanyak 13 outlier pada atribut Weight karena data-data tersebut terletak di luar kuartil pertama dan ketiga, dengan mayoritas berada di bawah kuartil pertama. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[432]:
```
cleaned_weight = cleaned_acidity.drop(outliers_weight.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_acidity['Weight'], color='blue', width=0.3)
plt.title('Weight Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_weight['Weight'], color='green', width=0.3)
plt.title('Weight Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.3 3. Length

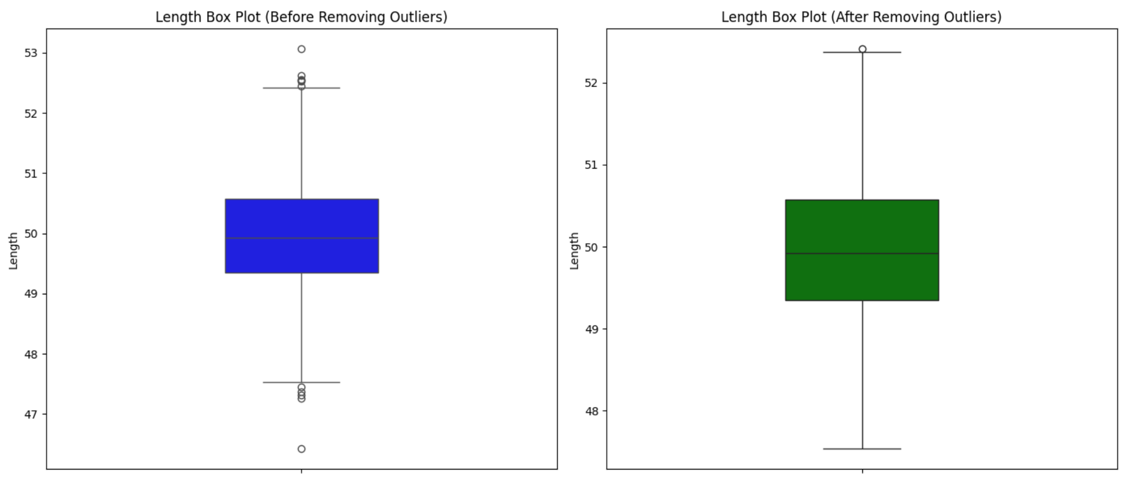
[433]:
```
outliers_length = show_outlier(cleaned_weight, 'Length')
```




```
Number of outliers: 11
Example of outliers:
      Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
40   7.990749  149.407475  53.065151    5.818155  7.778344  6.652354
522  7.505308  150.764398  47.452026    5.277505  8.168625  7.395418
637  7.079774  150.548409  52.543665    4.555585  8.943432  6.684328
747  7.370540  149.792352  52.626968    5.068159  9.285178  6.629390
792  5.935501  150.742882  47.313156    6.354260  7.586224  5.485945

     Sweetness Country_of_Origin  Firmness Grade         Price
40    6.721890           Ecuador  0.381398     C  20079.974475
522   5.457240           Ecuador  0.987490     A  20117.196240
637   5.919090           Ecuador  0.842187     B  20022.279099
747   5.944978        Costa Rica  0.827769     B  20009.159594
792   6.633735          Colombia  0.085594     C  20122.579622
```

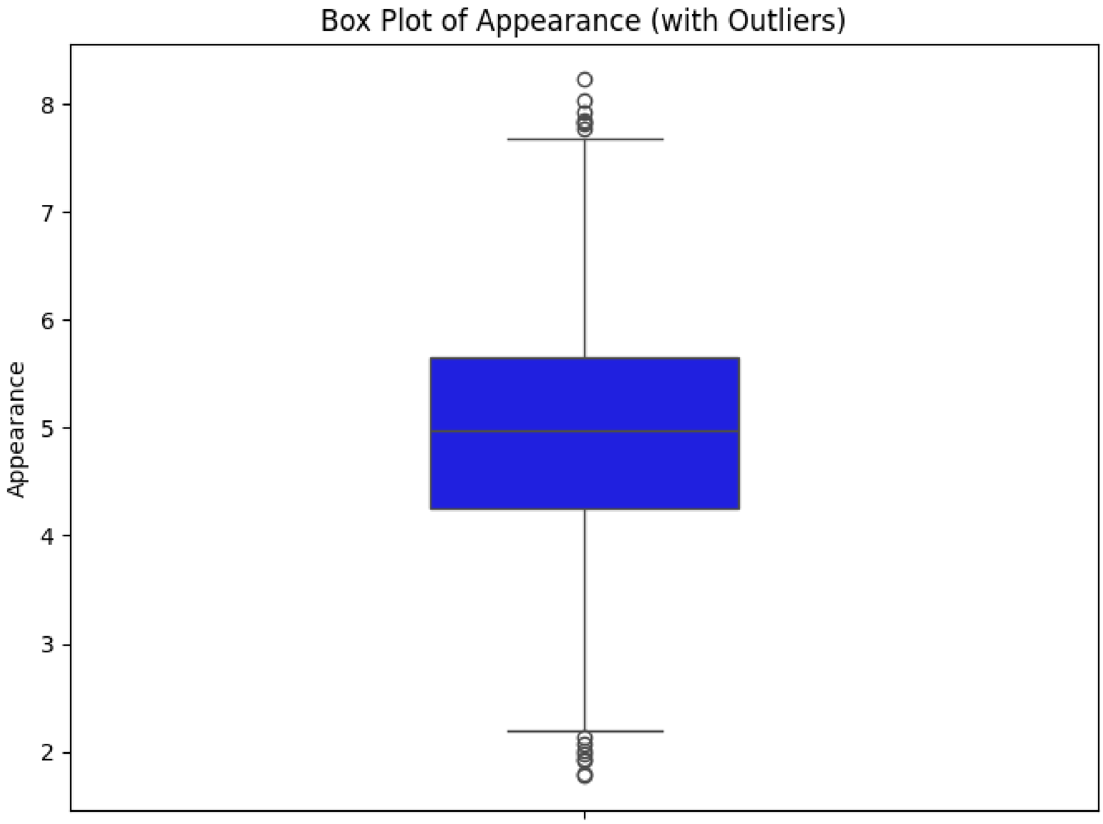
Dari hasil tersebut, diketahui terdapat sebanyak 11 outlier pada atribut Length karena data-data tersebut terletak di luar kuartil pertama dan ketiga. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[434]:
```
cleaned_length = cleaned_weight.drop(outliers_length.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_weight['Length'], color='blue', width=0.3)
plt.title('Length Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_length['Length'], color='green', width=0.3)
plt.title('Length Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.4 4. Appearance

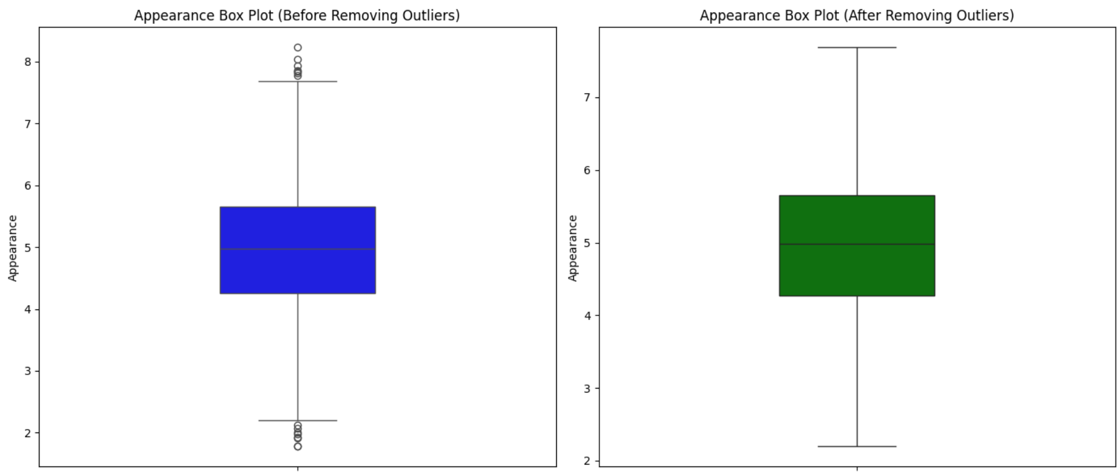
[435]:
```
outliers_appearance = show_outlier(cleaned_length, 'Appearance')
```




```
Number of outliers: 15
Example of outliers:
      Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
143  7.417676  149.129756  49.842773    8.233968  8.106042  6.562160
242  8.389403  150.325756  50.111896    2.127349  7.075064  8.022293
328  8.946531  148.557973  51.678060    7.927957  8.918308  6.642990
594  6.993142  151.444239  49.360201    7.842696  7.825411  7.529708
615  7.033057  149.908975  50.069306    2.007510  8.768558  6.683943

     Sweetness Country_of_Origin  Firmness Grade         Price
143   4.053357           Ecuador  0.779657     C  20043.252164
242   6.891265        Costa Rica  0.673737     B  19996.570126
328   5.183938           Ecuador  0.133276     A  20025.768326
594   7.402345        Costa Rica  0.877616     A  20098.980779
615   6.644307           Ecuador  0.366479     C  19947.950037
```

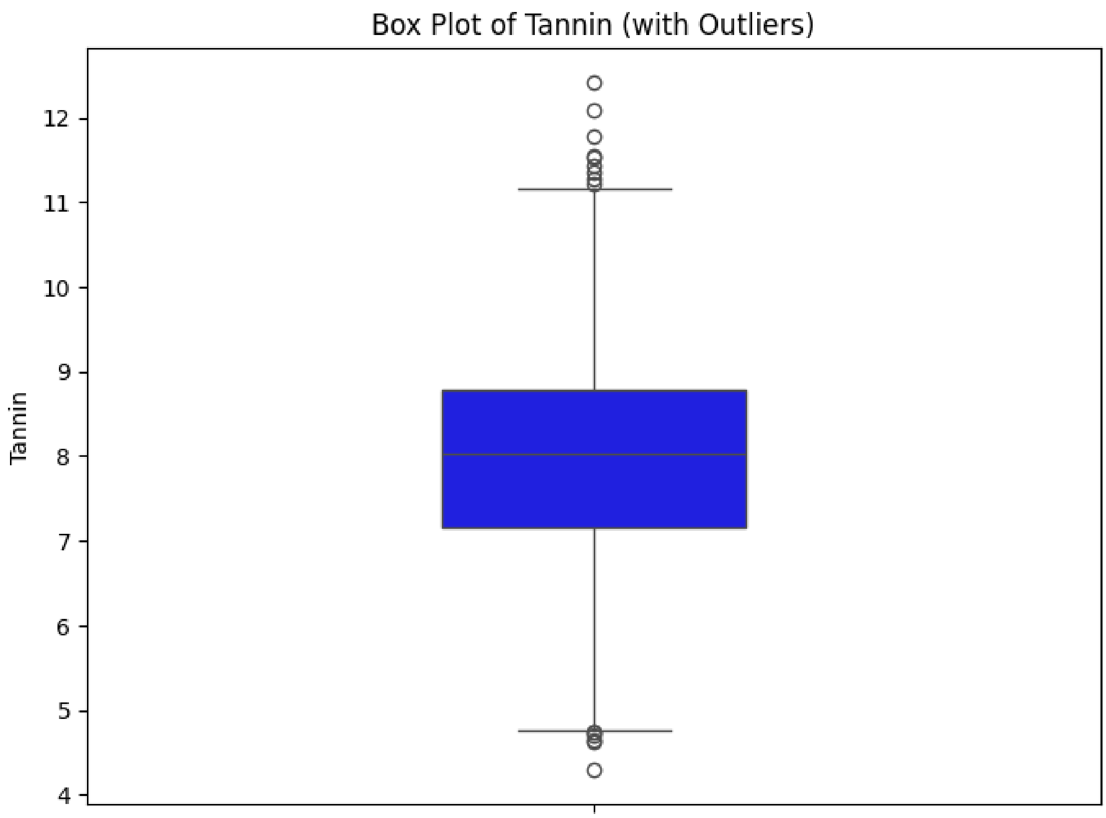
Dari hasil tersebut, diketahui terdapat sebanyak 15 outlier pada atribut Appearance karena data-data tersebut terletak di luar kuartil pertama dan ketiga. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[436]:
```
cleaned_appearance = cleaned_length.drop(outliers_appearance.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_length['Appearance'], color='blue', width=0.3)
plt.title('Appearance Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_appearance['Appearance'], color='green', width=0.3)
plt.title('Appearance Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.5 5. Tannin

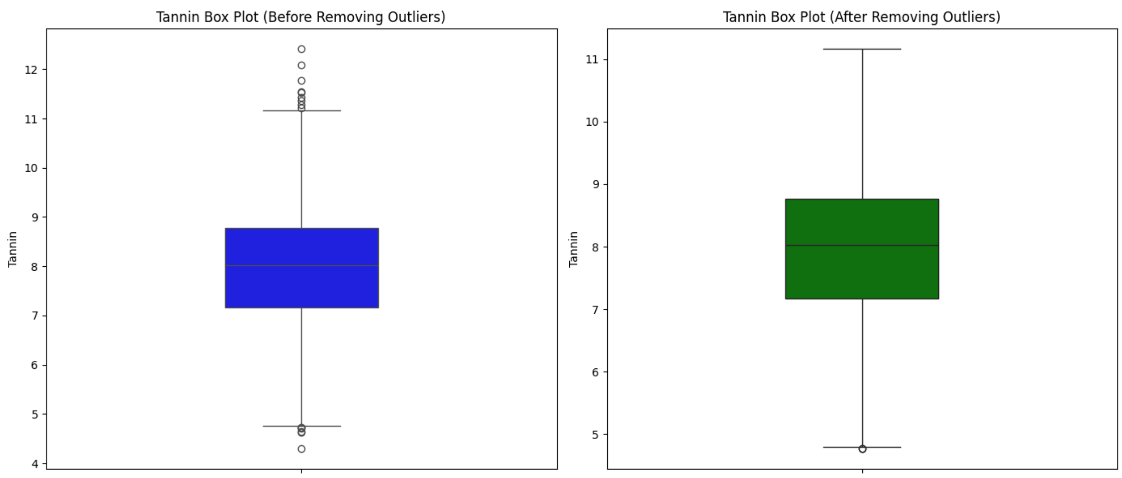
[437]:
```
outliers_tannin = show_outlier(cleaned_appearance, 'Tannin')
```




```
Number of outliers: 15
Example of outliers:
      Acidity      Weight     Length  Appearance     Tannin  Ripeness  \
30   7.296847  149.984547  47.780180    3.958296  11.217204  6.642264
187  8.148176  151.312434  49.260081    4.819624   4.732262  6.881741
217  6.161736  148.480788  50.976431    6.386963  11.273264  6.542920
400  7.745747  150.640334  49.813772    3.779888   4.291274  7.791520
466  7.323913  149.364193  51.680561    4.519369   4.731047  6.595642

     Sweetness Country_of_Origin  Firmness Grade         Price
30    6.376293           Ecuador  0.986866     C  20028.878271
187   5.057199           Ecuador  0.741092     B  20040.838113
217   6.796360          Colombia  0.395922     C  20099.464442
400   6.629639           Ecuador  0.250502     B  19961.894022
466   6.675869           Ecuador  0.248207     A  20013.944933
```

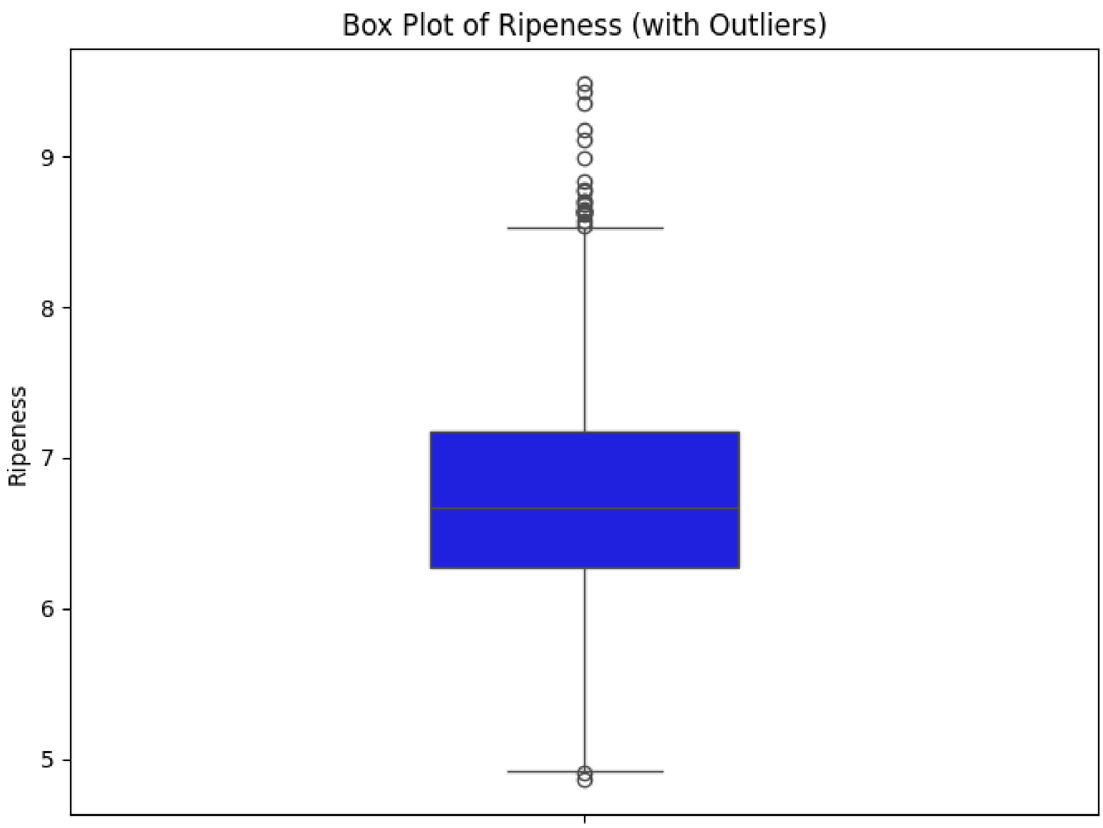
Dari hasil tersebut, diketahui terdapat sebanyak 15 outlier pada atribut Tannin karena data-data tersebut terletak di luar kuartil pertama dan ketiga, dengan mayoritas berada di atas kuartil ketiga. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[438]:
```
cleaned_tannin = cleaned_appearance.drop(outliers_tannin.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_appearance['Tannin'], color='blue', width=0.3)
plt.title('Tannin Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_tannin['Tannin'], color='green', width=0.3)
plt.title('Tannin Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.6 6. Ripeness

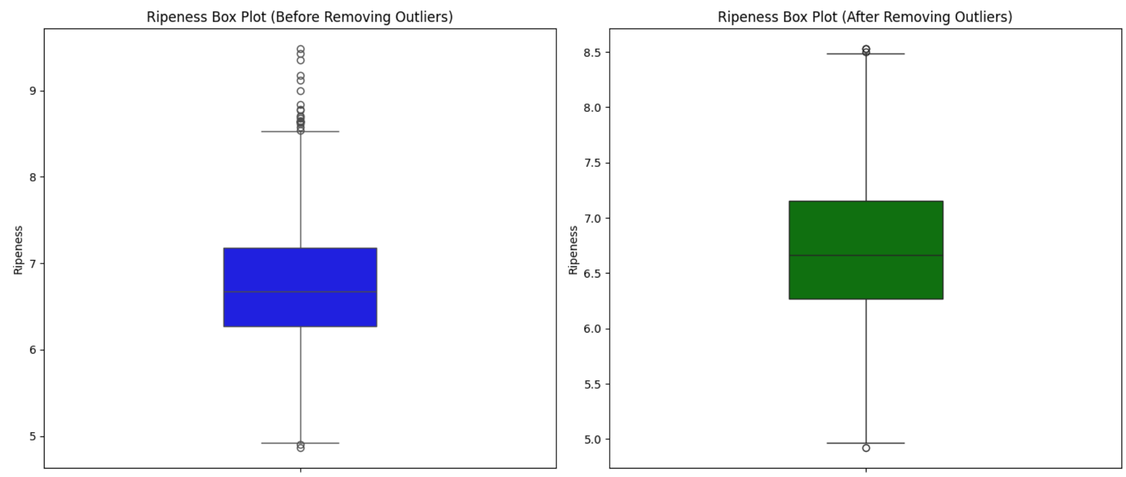
[439]:
```
outliers_ripeness = show_outlier(cleaned_tannin, 'Ripeness')
```




```
Number of outliers: 23
Example of outliers:
      Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
233  8.275414  150.761675  49.744720    4.073291  6.719979  8.767843
270  9.077867  149.824013  49.527495    3.708341  8.387320  8.991369
280  8.615655  149.937778  50.959567    5.026473  6.518194  8.645577
371  9.717588  152.139248  49.474674    4.753685  8.761573  8.676075
427  7.911172  149.781614  50.587902    5.184940  8.029231  8.628959

     Sweetness Country_of_Origin  Firmness Grade         Price
233   6.345430          Colombia  0.793434     C  19947.069570
270   6.768168           Ecuador  0.445386     C  19933.131645
280   6.112106        Costa Rica  0.707717     A  19932.173139
371   6.344848        Costa Rica  0.665029     A  19940.633405
427   6.366511           Ecuador  0.911884     A  20095.576432
```

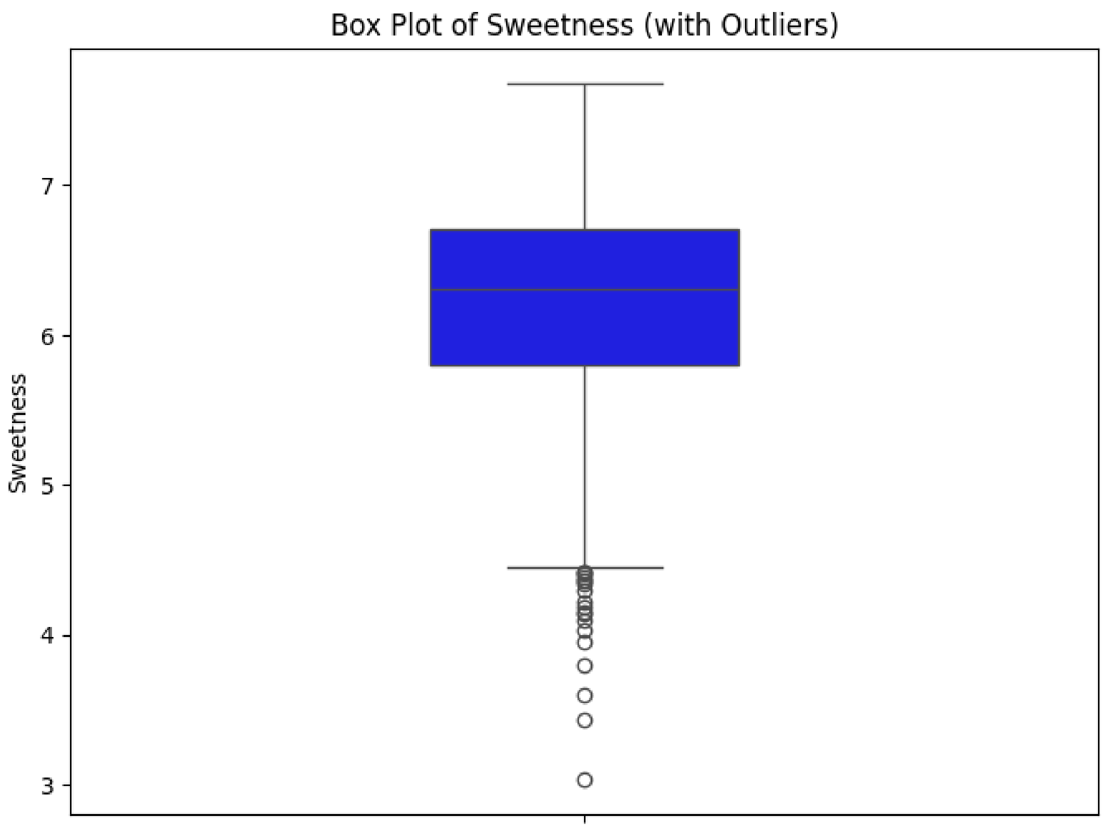
Dari hasil tersebut, diketahui terdapat sebanyak 23 outlier pada atribut Ripeness karena data-data tersebut terletak di luar kuartil pertama dan ketiga, dengan mayoritas berada di bawah kuartil pertama. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[440]:
```
cleaned_ripeness = cleaned_tannin.drop(outliers_ripeness.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_tannin['Ripeness'], color='blue', width=0.3)
plt.title('Ripeness Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_ripeness['Ripeness'], color='green', width=0.3)
plt.title('Ripeness Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.7 7. Sweetness

[441]:
```
outliers_sweetness = show_outlier(cleaned_ripeness, 'Sweetness')
```



Number of outliers: 19

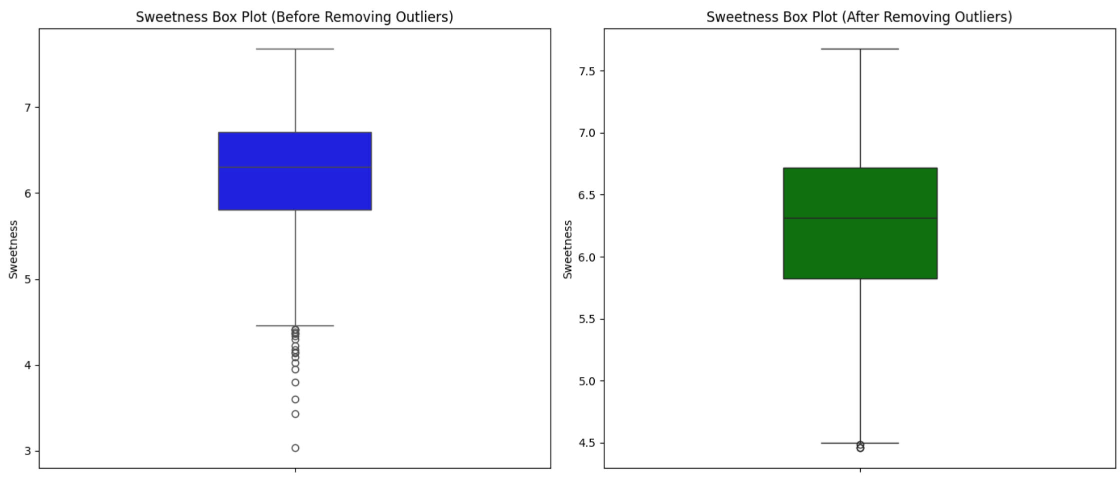
Example of outliers:

	Acidity	Weight	Length	Appearance	Tannin	Ripeness	\
29	8.179795	151.398830	49.094097	4.062627	9.074402	6.257676	
128	7.308226	149.177356	51.066010	4.774202	8.608403	7.370165	
172	8.502305	150.587300	50.735708	4.238291	6.565086	6.464917	
186	9.065962	150.122273	49.661178	4.055965	10.900618	6.681355	
232	6.310368	150.995122	50.495968	6.562654	7.069755	6.523758	

	Sweetness	Country_of_Origin	Firmness	Grade	Price
29	4.025152	Ecuador	0.680548	A	20008.017690
128	4.363350	Ecuador	0.596149	A	19919.650444
172	3.954111	Costa Rica	0.278695	C	19995.587224
186	4.411304	Costa Rica	0.289450	A	19898.199195
232	4.136793	Ecuador	0.028417	C	19914.114863

Dari hasil tersebut, diketahui terdapat sebanyak 19 outlier pada atribut Sweetness karena semua data tersebut terletak di bawah kuartil pertama. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[442]:
```
cleaned_sweetness = cleaned_ripeness.drop(outliers_sweetness.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_ripeness['Sweetness'], color='blue', width=0.3)
plt.title('Sweetness Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_sweetness['Sweetness'], color='green', width=0.3)
plt.title('Sweetness Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```

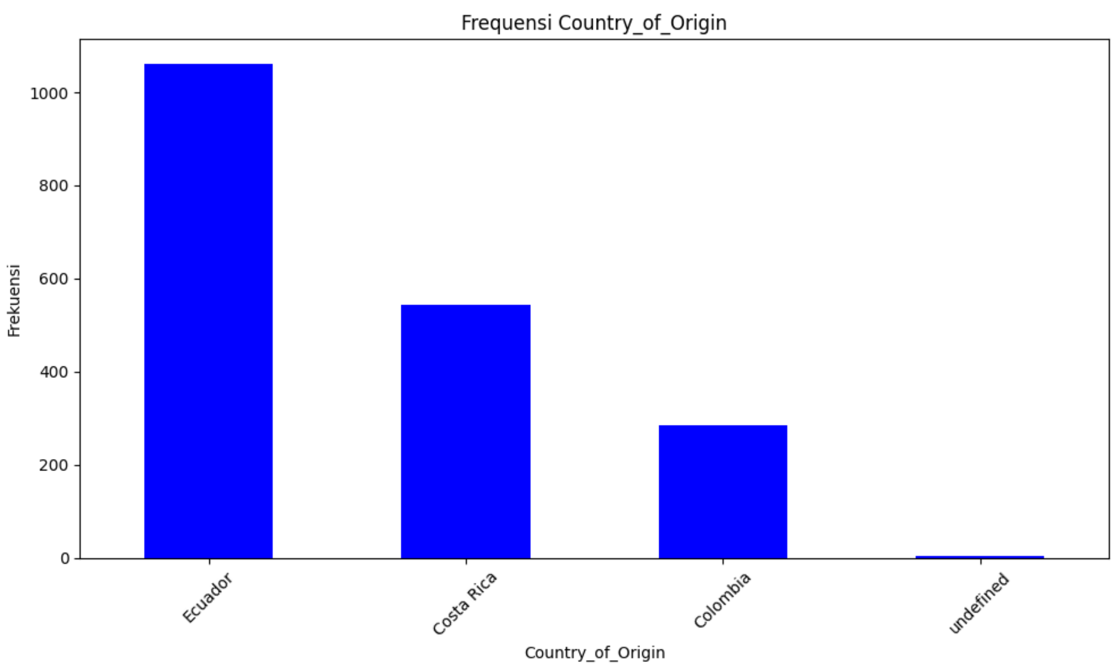


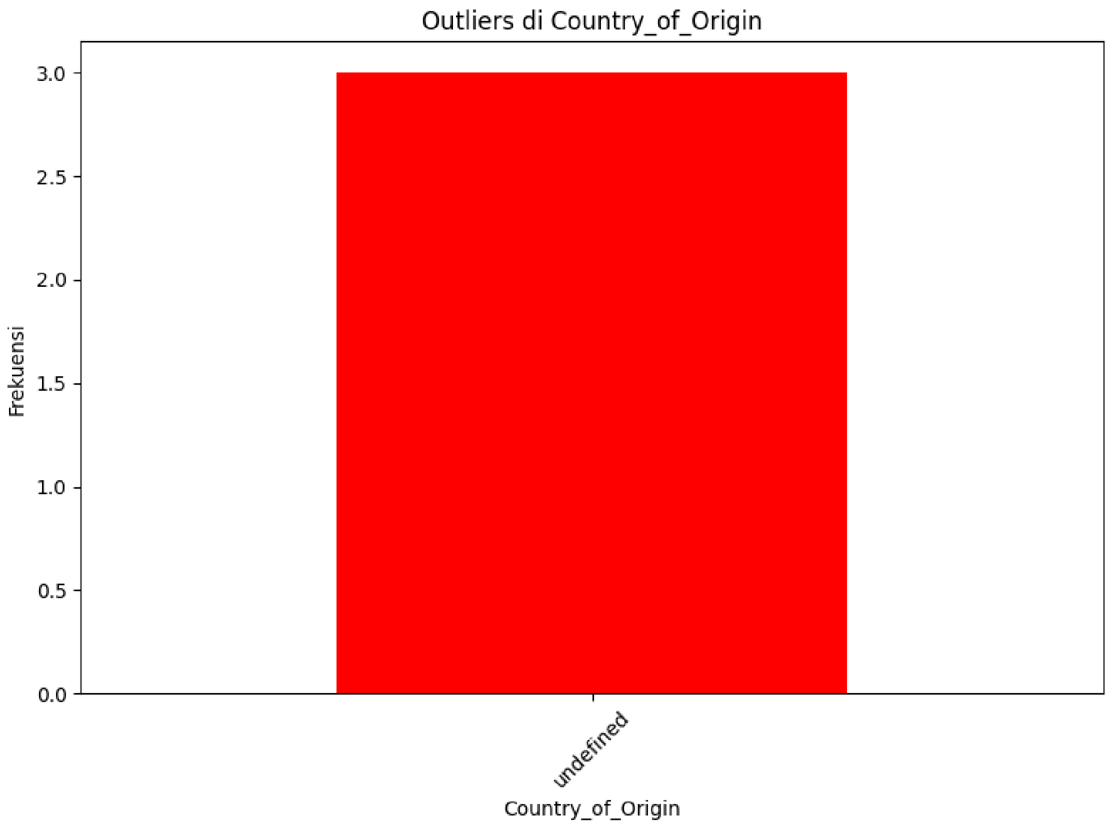
5.0.8 8. Country of Origin

Untuk mendapatkan outlier dari data dengan tipe kategorikal, akan digunakan 'threshold' untuk menentukan seberapa sering data ini muncul dalam dataset. Angka threshold diambil dari angka rata-rata kemunculan outlier pada data-data numerikal dalam dataset, yakni 14

[443]:
```
# Hitung frekuensi kemunculan data
country_counts = cleaned_sweetness['Country_of_Origin'].value_counts()

threshold = 14

# Identifikasi outlier dari frekuensi
outliers_counts = country_counts[country_counts < threshold]
outliers_country = cleaned_sweetness[cleaned_sweetness['Country_of_Origin'].
 ↪isin(outliers_counts.index)].head()

# Plot bar 'Country_of_Origin'dengan outlier
plt.figure(figsize=(10, 6))
```

```python
country_counts.plot(kind='bar', color='blue')
plt.title('Frequensi Country_of_Origin')
plt.xlabel('Country_of_Origin')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plot bar outliernya
plt.figure(figsize=(8, 6))
outliers_counts.plot(kind='bar', color='red')
plt.title('Outliers di Country_of_Origin')
plt.xlabel('Country_of_Origin')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Tampilkan data outlier
print("Number of outliers:", len(outliers_country))
if len(outliers_country) > 0:
    print("Example of outliers_country:")
    print(outliers_country.head())
```



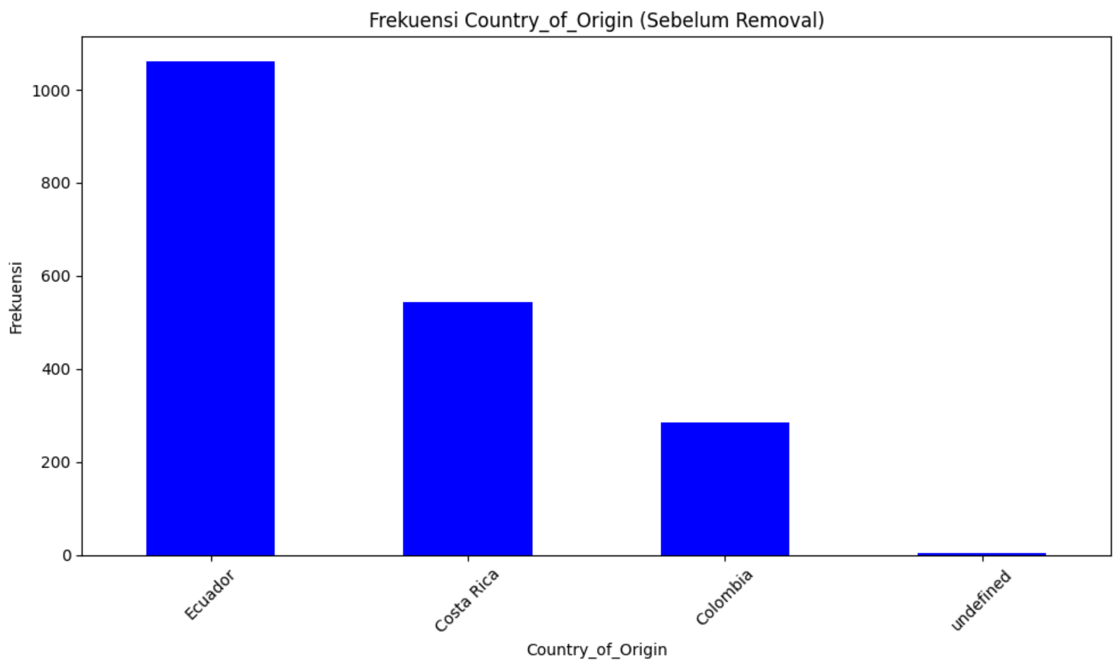


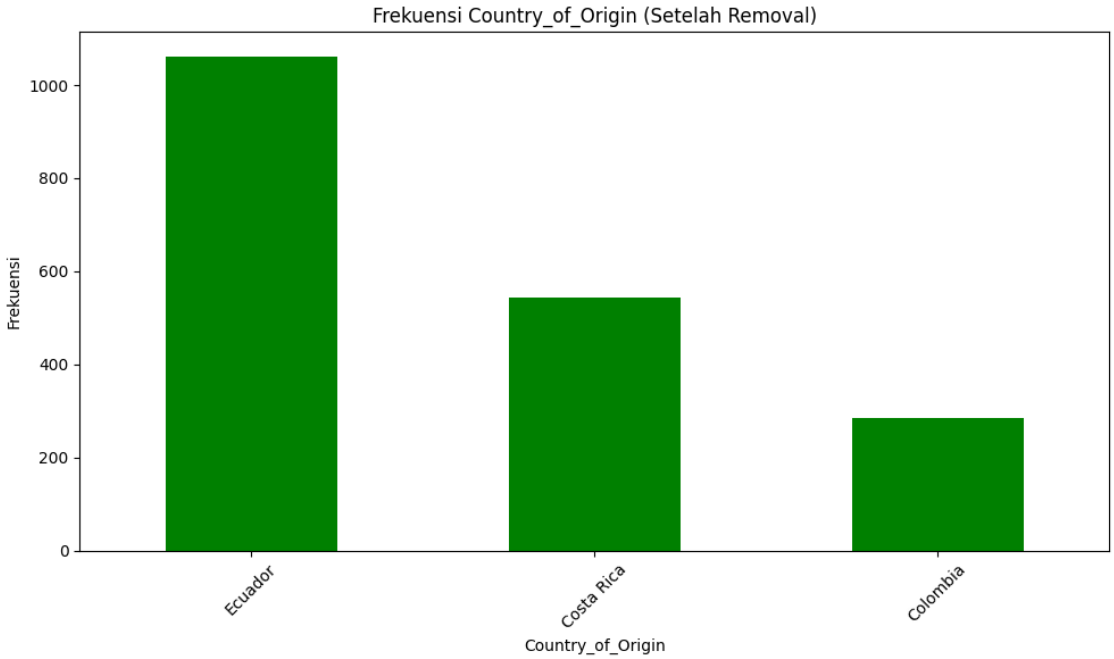

```
Number of outliers: 3
Example of outliers_country:
       Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
402   7.442386  150.698195  49.569259    5.961246  6.773352  6.971042
824   9.170880  150.101996  49.872307    5.257710  8.319038  6.326516
1853  7.191602  153.034788  49.475362    4.102973  8.850258  6.579281

      Sweetness Country_of_Origin  Firmness Grade         Price
402    6.565452         undefined  0.835268     C      0.000000
824    6.775390         undefined  0.900576     B  19977.154402
1853   6.868694         undefined  0.613568     C  20038.895670
```

Dari hasil tersebut, diketahui terdapat sebanyak 3 buah outlier pada atribut Country of Origin karena kemunculan data tersebut terbilang sedikit, dan jika dilihat, isi data tersebut bernama 'undefined'. Untuk menangani outlier ini, cukup dilakukan metode removal untuk membersihkannya dari outlier.

[444]:
```
# Hapus outlier
cleaned_country = cleaned_sweetness.drop(outliers_country.index)

# Hitung frekuensi 'Country_of_Origin' setelah hapus
country_counts_cleaned = cleaned_country['Country_of_Origin'].value_counts()
```

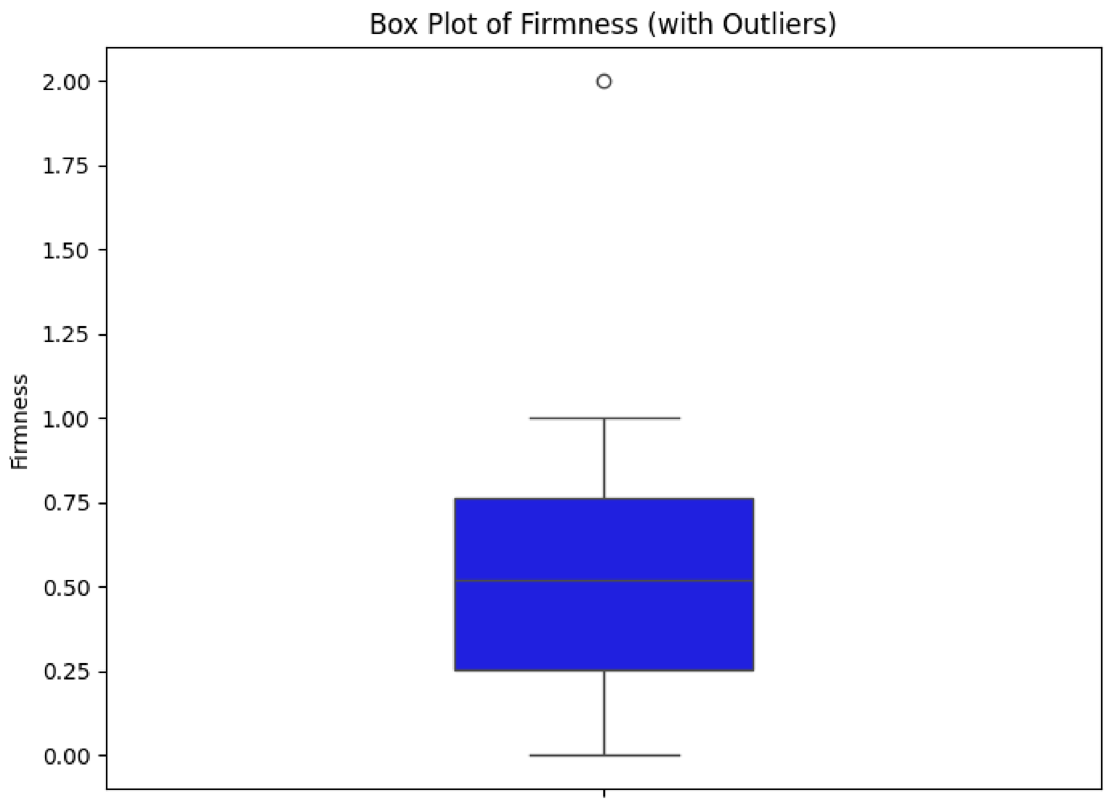
```
# Plot bar frekuensi 'Country_of_Origin' sebelum hapus
plt.figure(figsize=(10, 6))
country_counts.plot(kind='bar', color='blue')
plt.title('Frekuensi Country_of_Origin (Sebelum Removal)')
plt.xlabel('Country_of_Origin')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plot bar frekuensi 'Country_of_Origin' setelah hapus
plt.figure(figsize=(10, 6))
country_counts_cleaned.plot(kind='bar', color='green')
plt.title('Frekuensi Country_of_Origin (Setelah Removal)')
plt.xlabel('Country_of_Origin')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```





5.0.9 9. Firmness

[445]:
```
outliers_firmness = show_outlier(cleaned_country, 'Firmness')
```



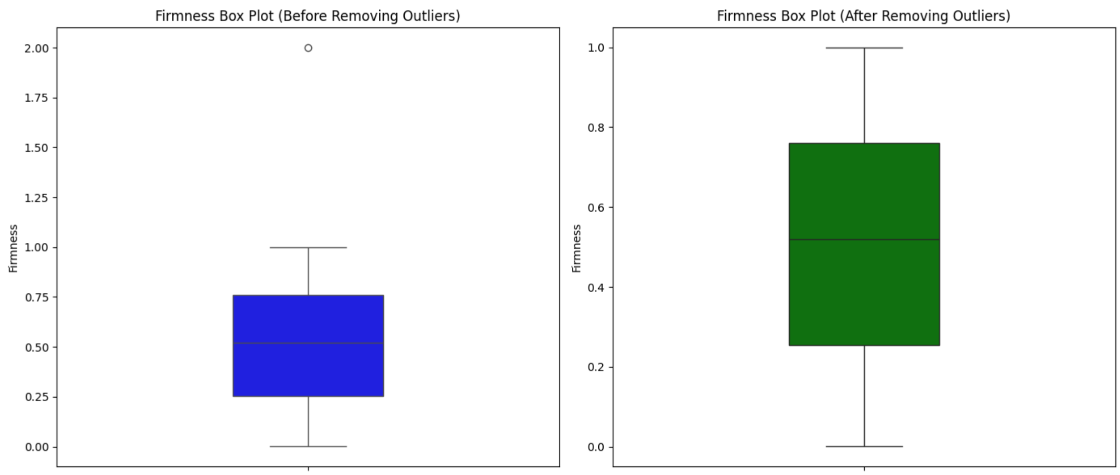

```
Number of outliers: 1
Example of outliers:
      Acidity      Weight     Length  Appearance   Tannin  Ripeness  \
283  6.890893  152.353645  49.364754    5.110521  8.51538  6.381191

     Sweetness Country_of_Origin  Firmness Grade         Price
283   6.464615          Colombia       2.0     C  19957.539273
```

Dari hasil tersebut, diketahui terdapat hanya sebuah outlier pada atribut Firmness karena data tersebut terletak jauh di atas kuartil ketiga. Untuk menangani outlier ini, cukup dilakukan metode removal untuk membersihkannya dari outlier.

[446]:

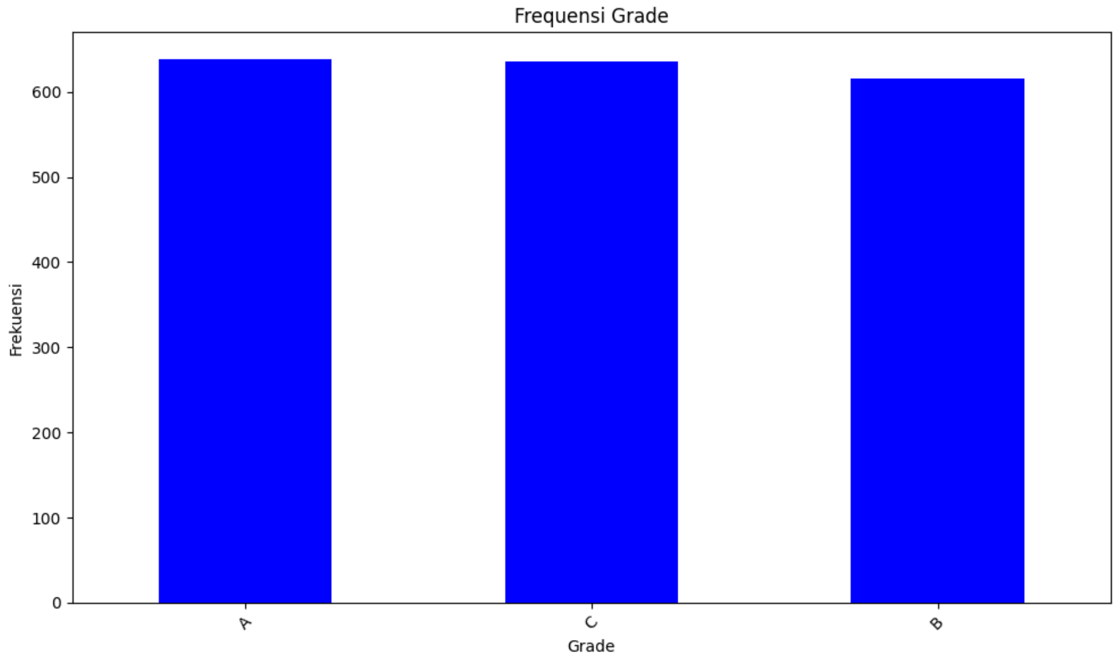
```
cleaned_firmness = cleaned_country.drop(outliers_firmness.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_country['Firmness'], color='blue', width=0.3)
plt.title('Firmness Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_firmness['Firmness'], color='green', width=0.3)
plt.title('Firmness Box Plot (After Removing Outliers)')
```

```
plt.tight_layout()
plt.show()
```



5.0.10 10. Grade

[447]:
```
# Hitung frekuensi kemunculan data
grade_counts = cleaned_firmness['Grade'].value_counts()

threshold = 101

# Identifikasi outlier dari frekuensi
outliers_counts = grade_counts[grade_counts < threshold]
outliers_grade = cleaned_firmness[cleaned_firmness['Grade'].
    ↪isin(outliers_counts.index)].head()

# Plot bar 'Grade'dengan outlier
plt.figure(figsize=(10, 6))
grade_counts.plot(kind='bar', color='blue')
plt.title('Frequensi Grade')
plt.xlabel('Grade')
plt.ylabel('Frekuensi')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Tampilkan data outlier
print("Number of outliers:", len(outliers_grade))
if len(outliers_grade) > 0:
    print("Example of outliers_grade:")
    print(outliers_grade.head())
```

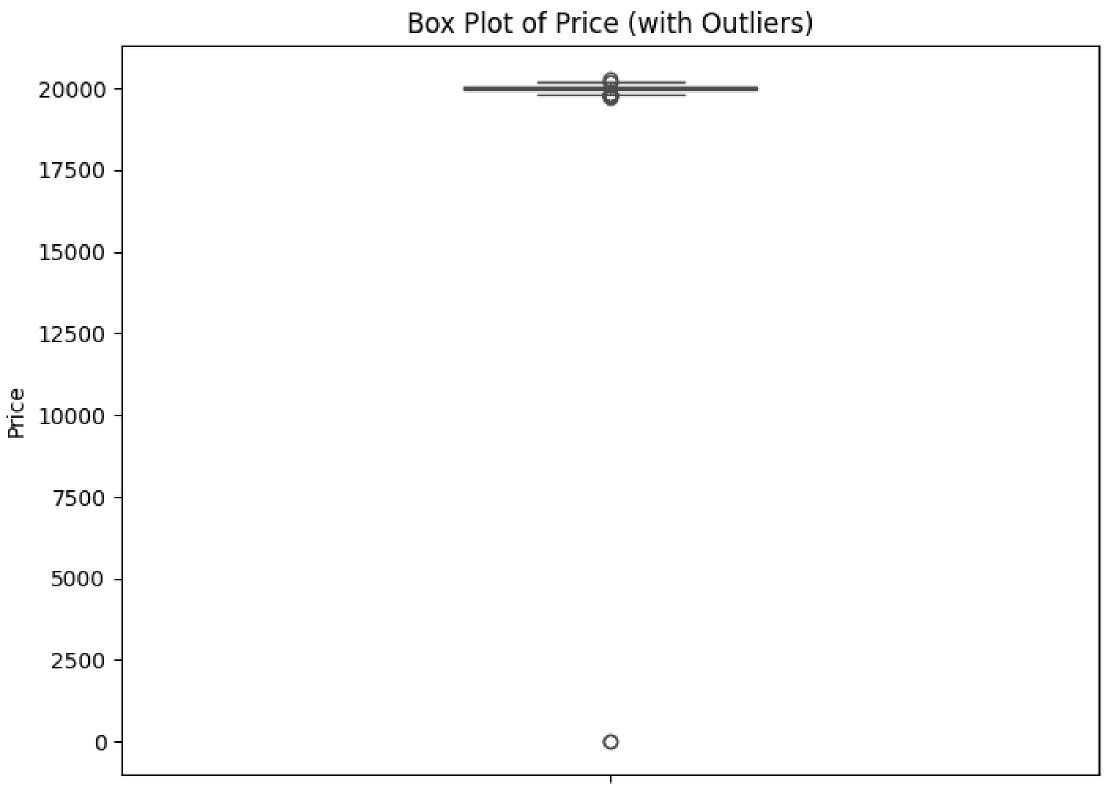



```
Number of outliers: 0
```

Dari hasil tersebut, diketahui tidak terdapat outlier pada atribut Grade dalam dataset. Sehingga tidak diperlukan penanganan apapun.

5.0.11 11. Price

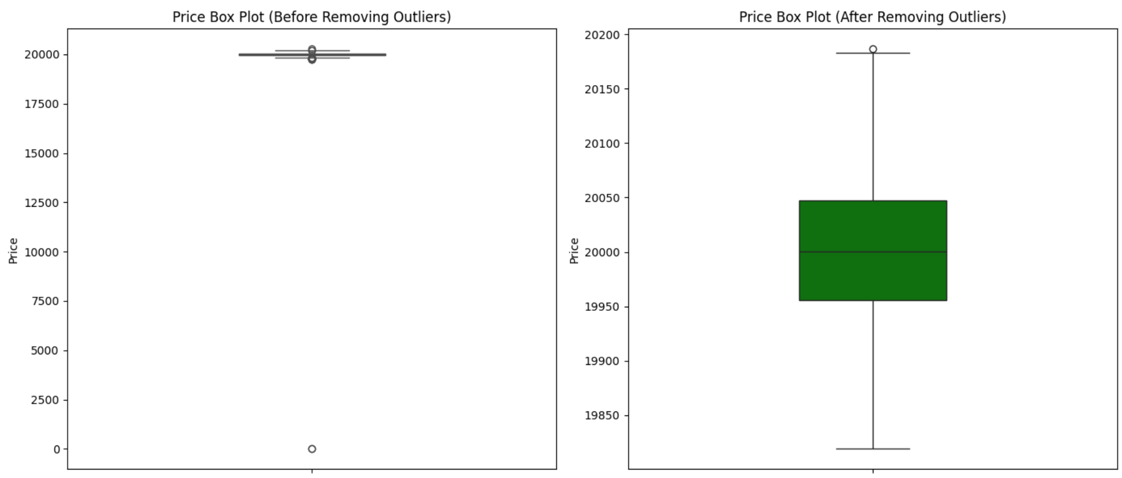
[448]:
```
outliers_price = show_outlier(cleaned_firmness, 'Price')
```




```
Number of outliers: 17
Example of outliers:
     Acidity      Weight     Length  Appearance    Tannin  Ripeness  \
53   8.934659  147.655863  50.385195    4.580007  6.420740  7.505213
378  8.949143  148.667892  49.835753    6.465544  7.761297  6.814243
689  8.435672  150.946882  48.786737    5.359197  7.639943  6.045392
690  7.264353  147.969278  49.496607    5.313865  5.819292  6.557644
759  8.772708  148.888507  51.621197    6.653752  5.518499  8.386520

     Sweetness Country_of_Origin  Firmness Grade         Price
53    6.947486           Ecuador  0.115099     A  19803.813931
378   5.417429          Colombia  0.646237     B  19781.569703
689   6.011245        Costa Rica  0.667310     B  19729.904103
690   6.069816           Ecuador  0.974611     A  19809.257798
759   5.609371          Colombia  0.170188     C  20199.676334
```

Dari hasil tersebut, diketahui terdapat sebanyak 17 outlier pada atribut Price karena data-data tersebut terletak di luar kuartil pertama dan ketiga, bahkan terdapat sebuah data yang berada jauh di bawah kuartil pertam. Untuk menangani outlier ini, dapat dilakukan metode removal untuk membersihkannya dari outlier.

[449]:
```
cleaned_price = cleaned_firmness.drop(outliers_price.index)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_firmness['Price'], color='blue', width=0.3)
plt.title('Price Box Plot (Before Removing Outliers)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_price['Price'], color='green', width=0.3)
plt.title('Price Box Plot (After Removing Outliers)')
plt.tight_layout()
plt.show()
```



5.0.12 Hasil Data yang Sudah Dibersihkan

[450]:
```
# Data sebelum dibersihkan
print("Banyak data sebelum dibersihkan: ")
print(len(data))

# Gabungkan cleaned DataFrames
cleaned_data = cleaned_price.copy()

# Reset index DataFrame
cleaned_data.reset_index(drop=True, inplace=True)

# Data setelah dibersihkan
print("Banyak data setelah dibersihkan:")
print(len(cleaned_data))
```

```
Banyak data sebelum dibersihkan:
2000
```

```
Banyak data setelah dibersihkan:
1871
```

6 3. Visualisasi Plot Distribusi

6.0.1 1. Acidity

Fungsi menampilkan visualisasi plot distribusi dari data numerik

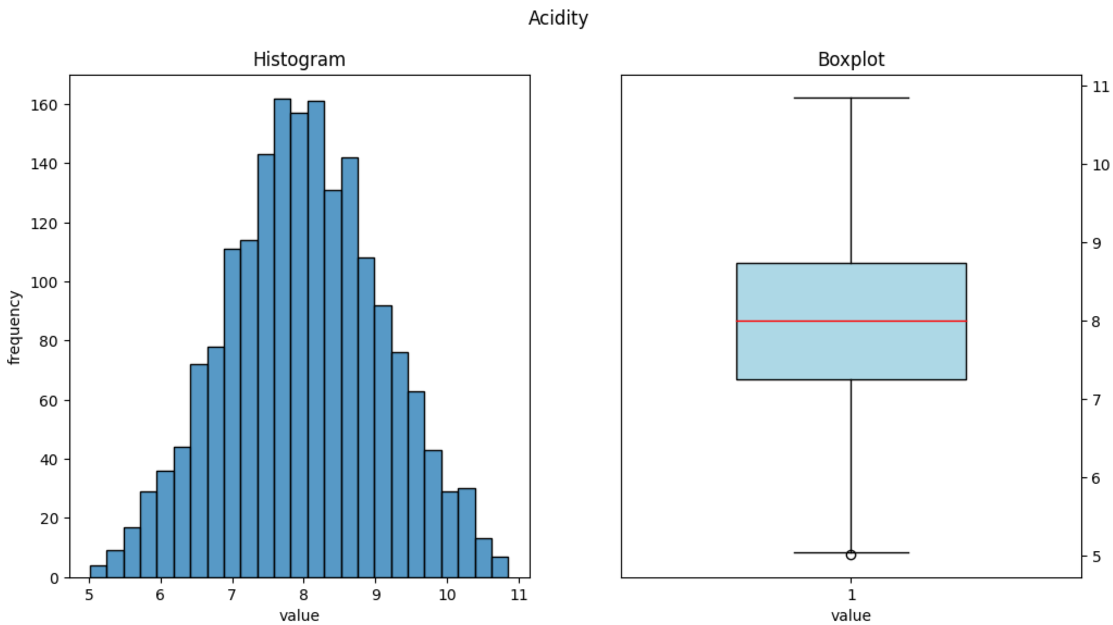
[451]:
```python
def distribution_plot_numeric(atr):
    # Buat Figure untuk Plot Histogram dan Box Plot
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))

    # Plot Histogram
    sns.histplot(atr, ax=ax1)
    ax1.set_title("Histogram")
    ax1.set_ylabel("frequency")
    ax1.set_xlabel("value")

    # Plot Box Plot
    ax2.boxplot(atr, vert=True, widths=0.5, patch_artist=True,
 ↪boxprops=dict(facecolor='lightblue', color='black'),
 ↪medianprops=dict(color='red'))
    ax2.set_title('Boxplot')
    ax2.set_xlabel('value')
    ax2.yaxis.tick_right()

    # Tampilkan hasil
    plt.suptitle(atr.name)
    plt.show()
```

Fungsi untuk menampilkan visualisasi plot distribusi dari data berupa string

[452]:
```python
def distribution_plot_string(atr):
    # Buat Figure untuk Plot Histogram
    fig, ax = plt.subplots(ncols=1, figsize=(6,6))

    # Plot Histogram
    sns.histplot(atr, ax=ax)
    ax.set_title("Histogram")
    ax.set_ylabel("frequency")
    ax.set_xlabel("value")

    # Tampilkan hasil
    plt.suptitle(atr.name)
    plt.show()
```

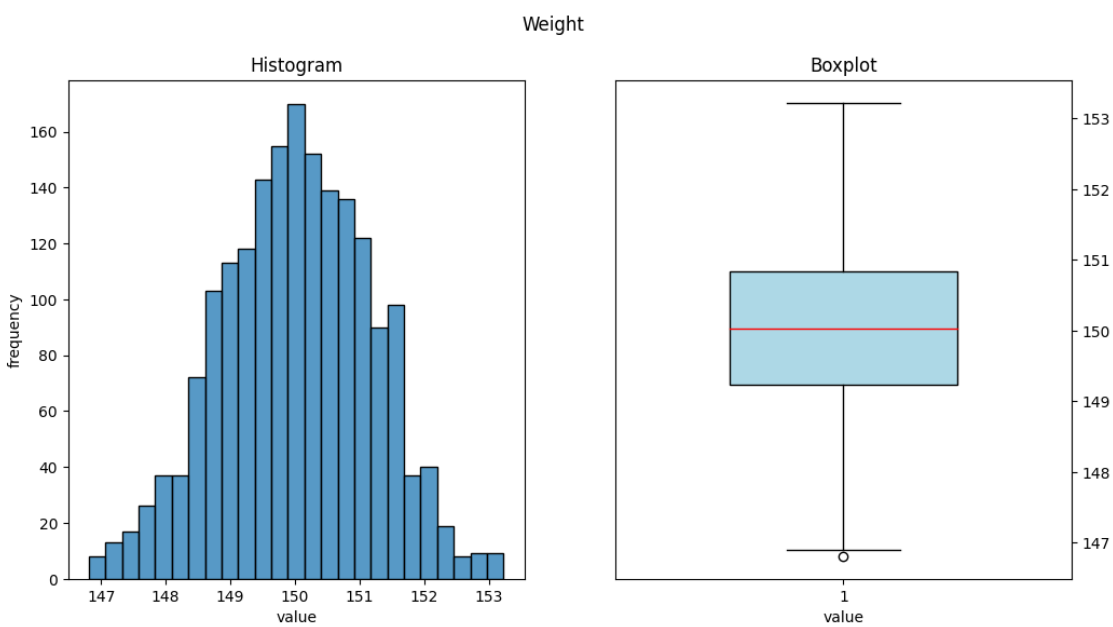
6.0.2 1. Acidity

[453]:
```
distribution_plot_numeric(cleaned_data['Acidity'])
```



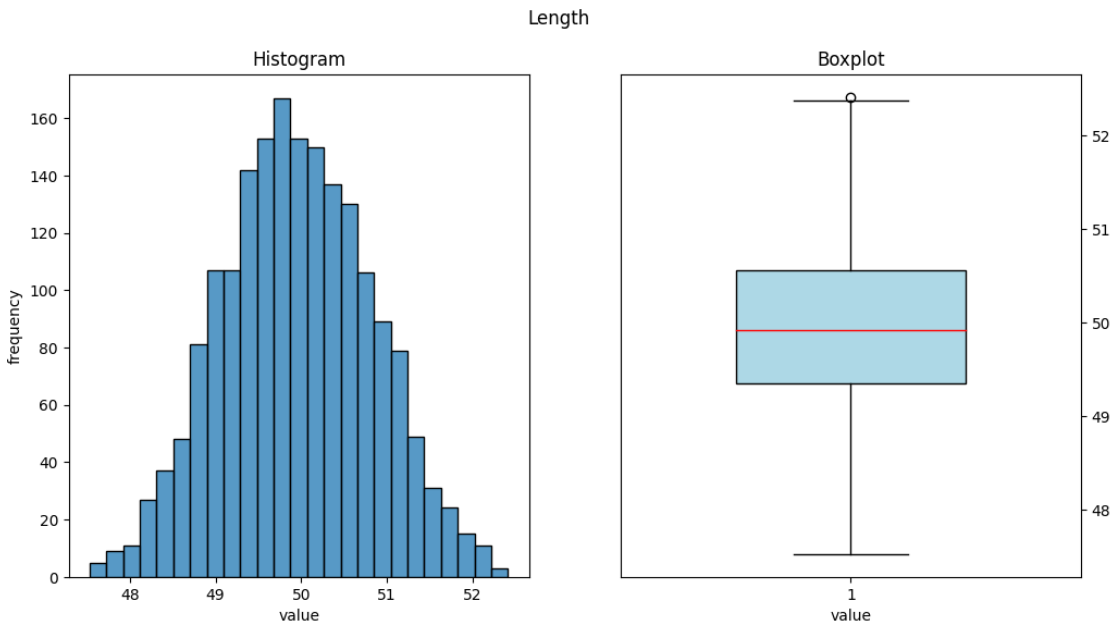
6.0.3 2. Weight

[454]:
```
distribution_plot_numeric(cleaned_data['Weight'])
```



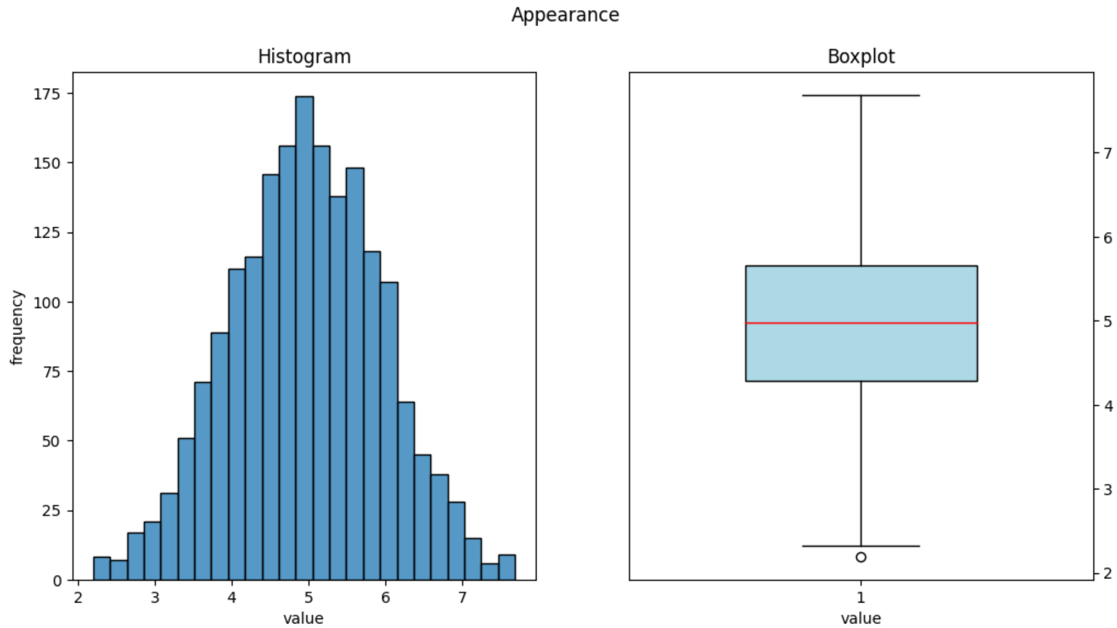
6.0.4 3. Length

[455]:
```
distribution_plot_numeric(cleaned_data['Length'])
```



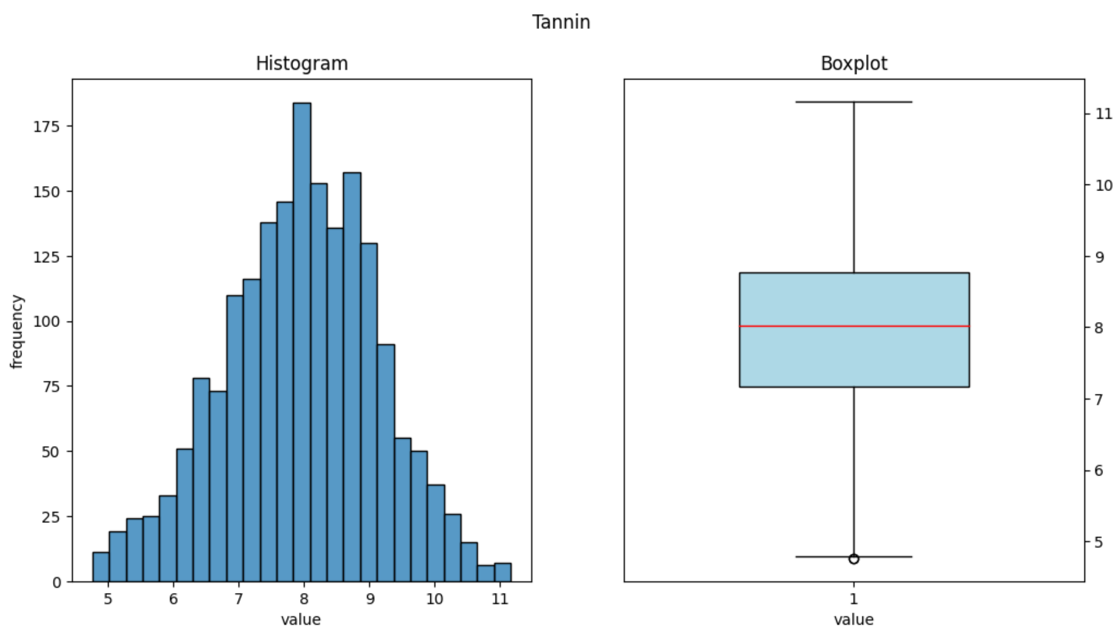
6.0.5 4. Appearance

[456]:
```
distribution_plot_numeric(cleaned_data['Appearance'])
```



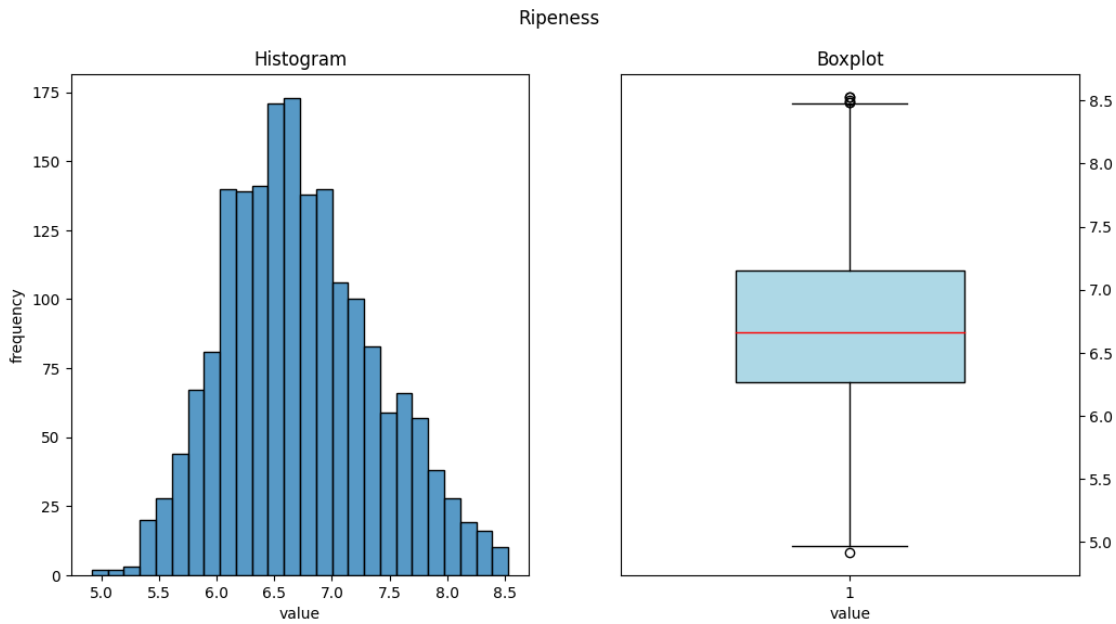
6.0.6 5. Tannin

[457]:
```
distribution_plot_numeric(cleaned_data['Tannin'])
```



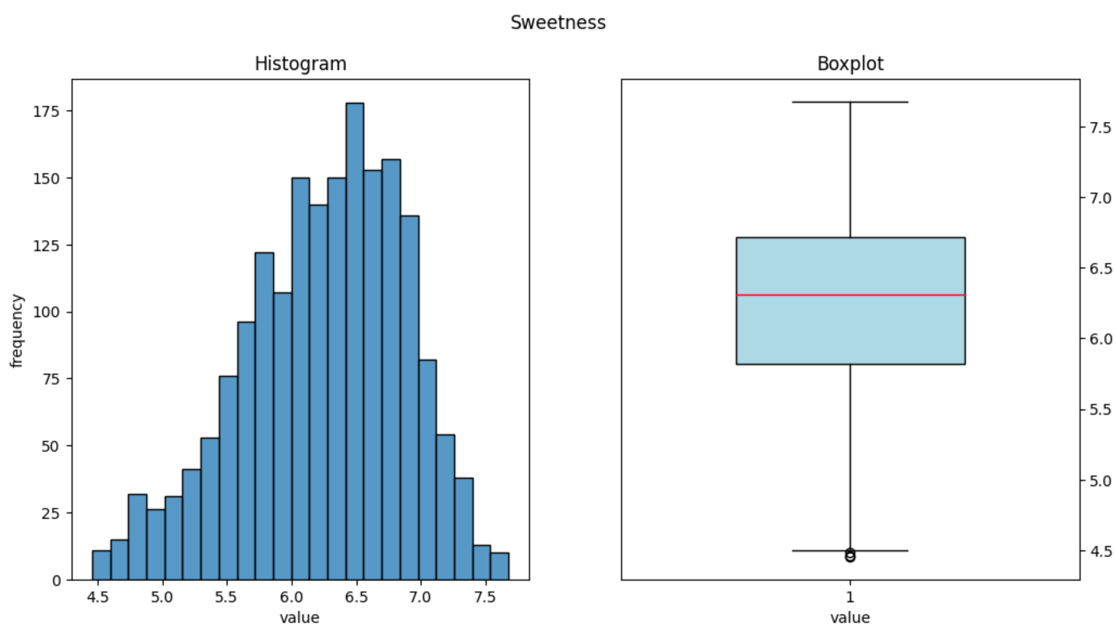
6.0.7 6. Ripeness

[458]:
```
distribution_plot_numeric(cleaned_data['Ripeness'])
```



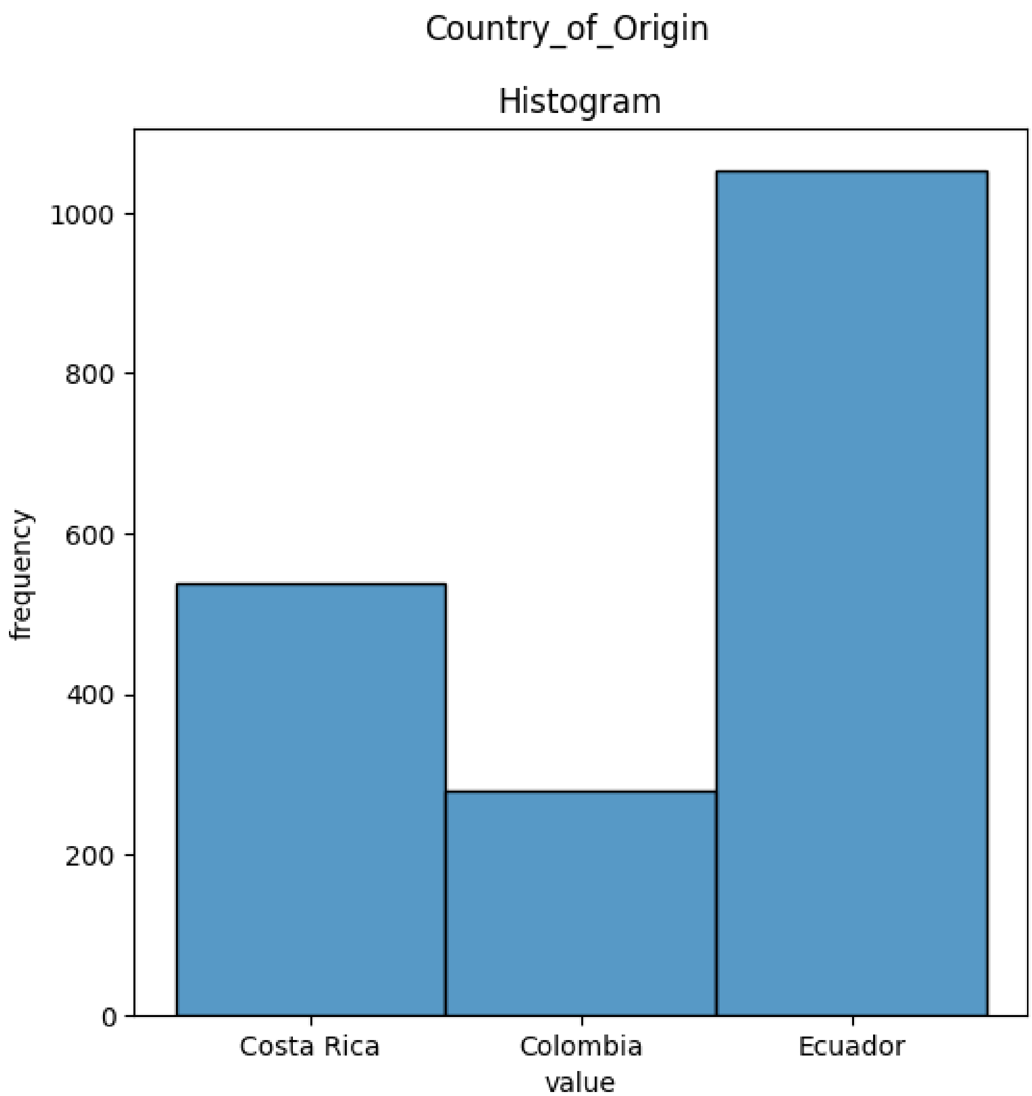
6.0.8 7. Sweetness

[459]:
```
distribution_plot_numeric(cleaned_data['Sweetness'])
```



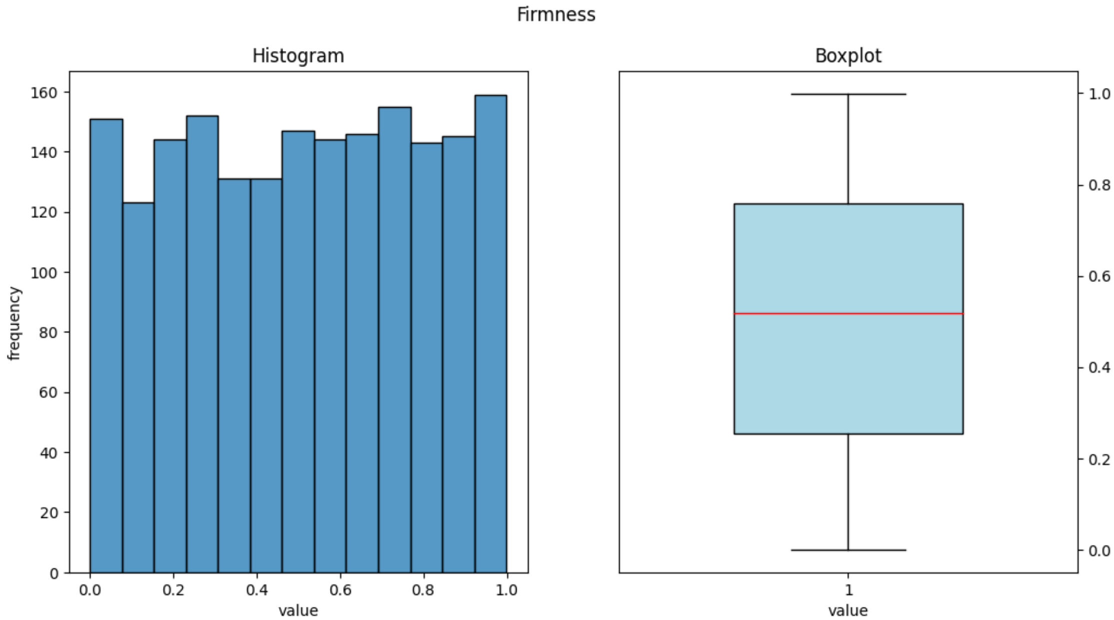
6.0.9 8. Country of Origin

[460]:
```
distribution_plot_string(cleaned_data['Country_of_Origin'])
```



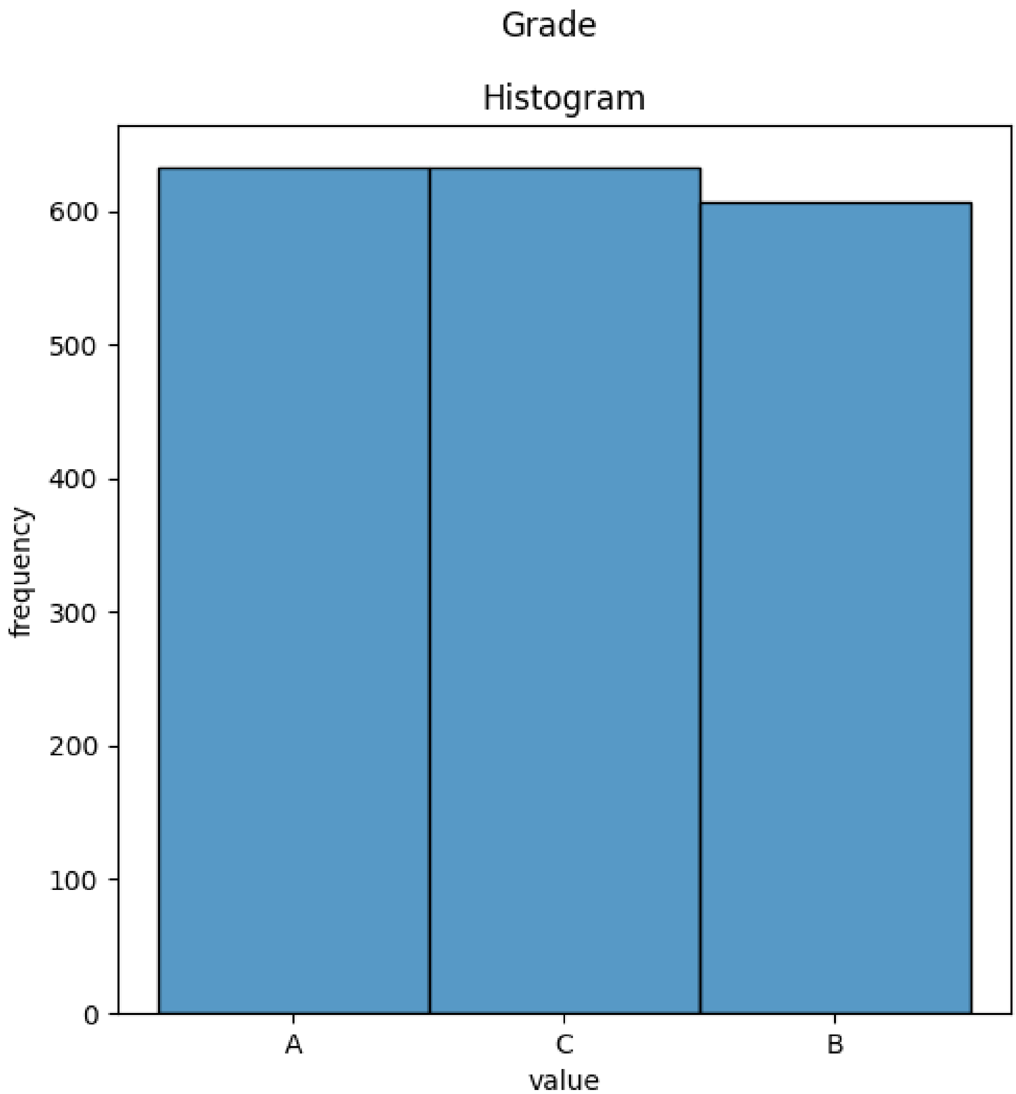
6.0.10 9. Firmness

[461]:
```
distribution_plot_numeric(cleaned_data['Firmness'])
```



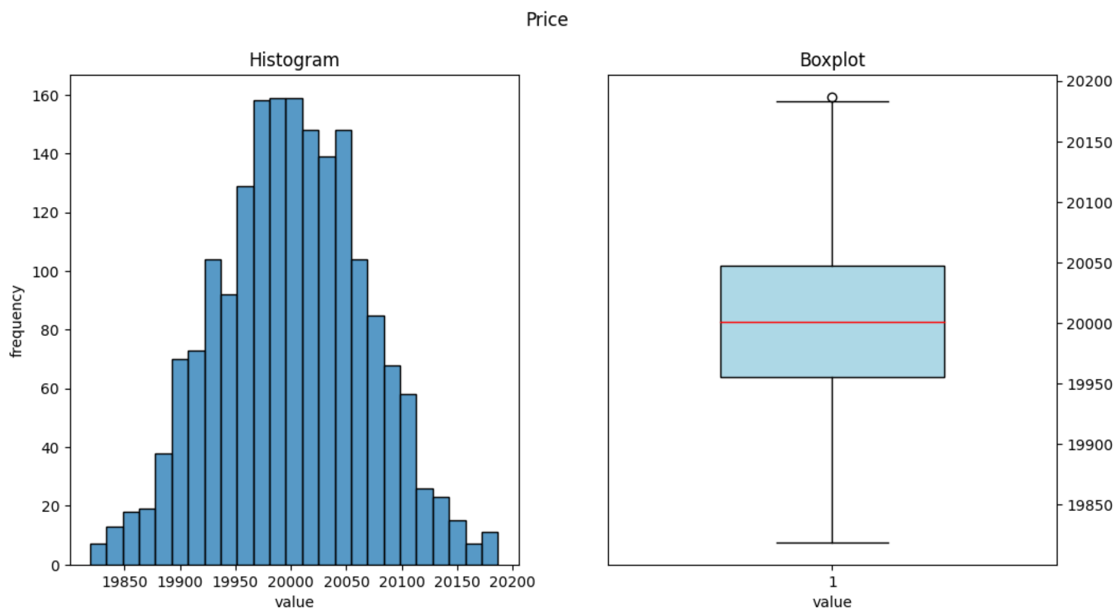
6.0.11 10. Grade

[462]:
```
distribution_plot_string(cleaned_data['Grade'])
```



6.0.12 11. Price

[463]:
```
distribution_plot_numeric(cleaned_data['Price'])
```



7 4. Menentukan setiap kolom numerik berdistribusi normal atau tidak

[464]:
```
def normal_distribution_test(atr, alpha=0.05):
    stat, p = shapiro(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'))
        test_distribution_type(atr, alpha )
```

[465]:
```
def test_distribution_type(atr, alpha=0.05):
    print()
    display(Markdown("***Cari Distribusi yang Tepat***"))
    f = Fitter(atr, distributions=get_common_distributions())
    f.fit()
    best_dist = f.get_best(method = 'sumsquare_error')
    f.summary()
    # Extract the name of the best distribution
    best_dist_name = list(best_dist.keys())[0]

    # Extract parameters of the best distribution
    best_dist_params = best_dist[best_dist_name]

    # Display the best distribution with Markdown
```

```
    display(Markdown(f"### Distribusi Terbaik"))
    display(Markdown(f"Distribusi terbaik berdasarkan Sum of Squares Error␣
↪adalah: **{best_dist_name}**"))
    display(Markdown(f"Dengan parameter: **{best_dist_params}**"))
```

[466]:
```
normal_distribution_test(cleaned_data['Acidity'])
```

Nilai p = ***0.003275780472904443***7

Data ***Acidity*** Tidak Terdistribusi Normal

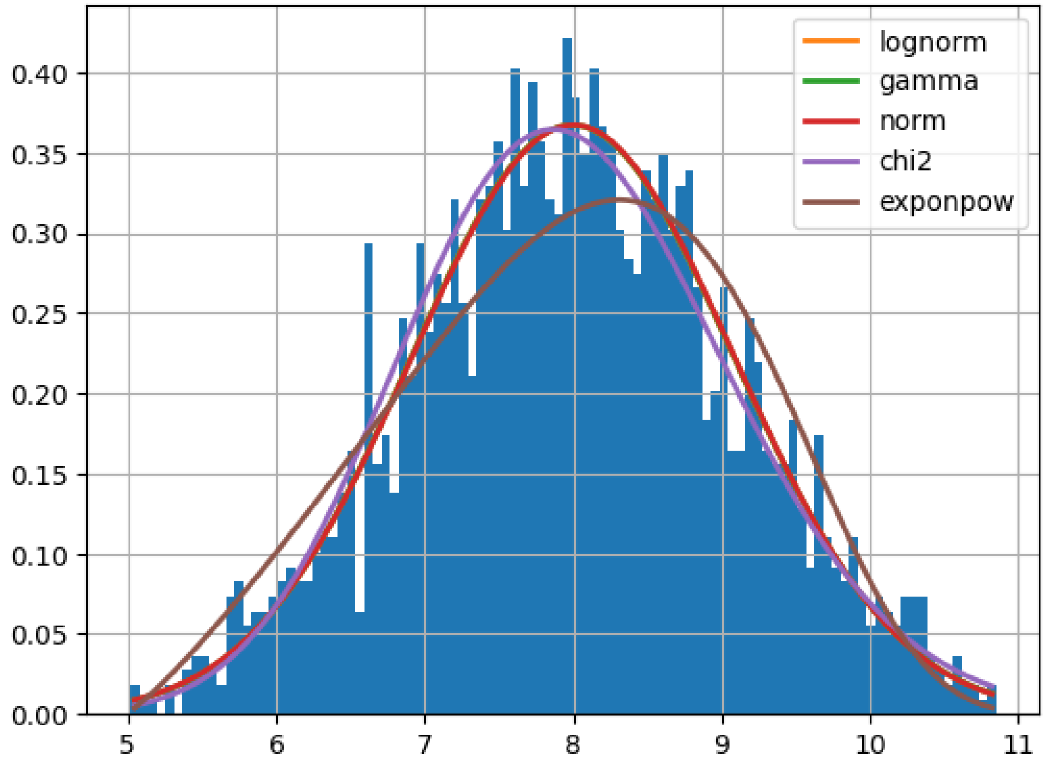
Cari Distribusi yang Tepat

```
2024-05-24 15:36:11.199 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.426439)
2024-05-24 15:36:11.239 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=2.789187)
2024-05-24 15:36:11.484 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.122012)
2024-05-24 15:36:11.878 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.11403)
2024-05-24 15:36:11.933 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.113982)
2024-05-24 15:36:11.966 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.114256)
2024-05-24 15:36:11.999 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=1.460763)
2024-05-24 15:36:12.017 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.202231)
2024-05-24 15:36:12.038 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.631215)
2024-05-24 15:36:12.048 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=1.551872)
```

7.0.1 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **lognorm**

Dengan parameter: **{'s': 0.004535062404587585, 'loc': -231.4940204467273, 'scale': 239.49380583103516}**



[467]:
```
normal_distribution_test(cleaned_data['Weight'])
```

Nilai p = ***0.015015353448688984***

Data ***Weight*** Tidak Terdistribusi Normal

Cari Distribusi yang Tepat

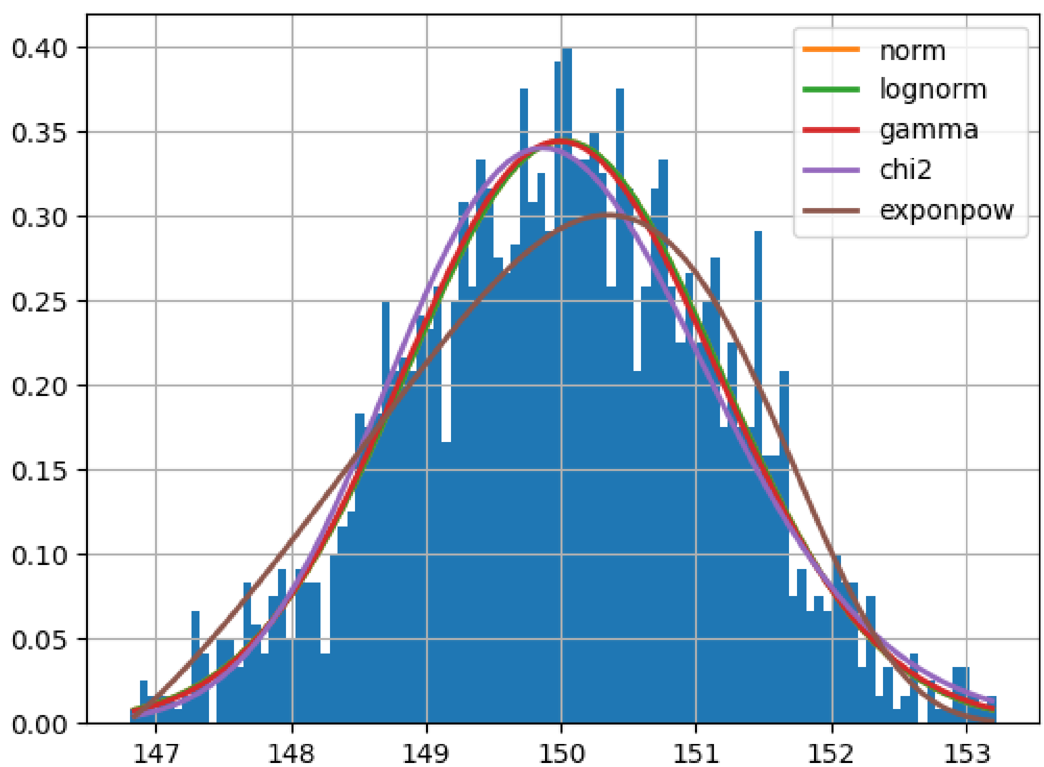
```
2024-05-24 15:36:14.005 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.410732)
2024-05-24 15:36:14.043 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=2.422189)
2024-05-24 15:36:14.389 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.134742)
2024-05-24 15:36:14.642 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.161306)
```

```
2024-05-24 15:36:14.689 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.115585)
2024-05-24 15:36:14.712 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.115583)
2024-05-24 15:36:14.743 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.117698)
2024-05-24 15:36:14.757 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=1.365499)
2024-05-24 15:36:14.770 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.596504)
2024-05-24 15:36:14.781 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=1.423539)
```

7.0.2 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **norm**

Dengan parameter: **{'loc': 150.01906265635472, 'scale': 1.157814497533124}**



```
[468]: normal_distribution_test(cleaned_data['Length'])
```

Nilai p = ***0.01219115499407053***

Data ***Length*** Tidak Terdistribusi Normal

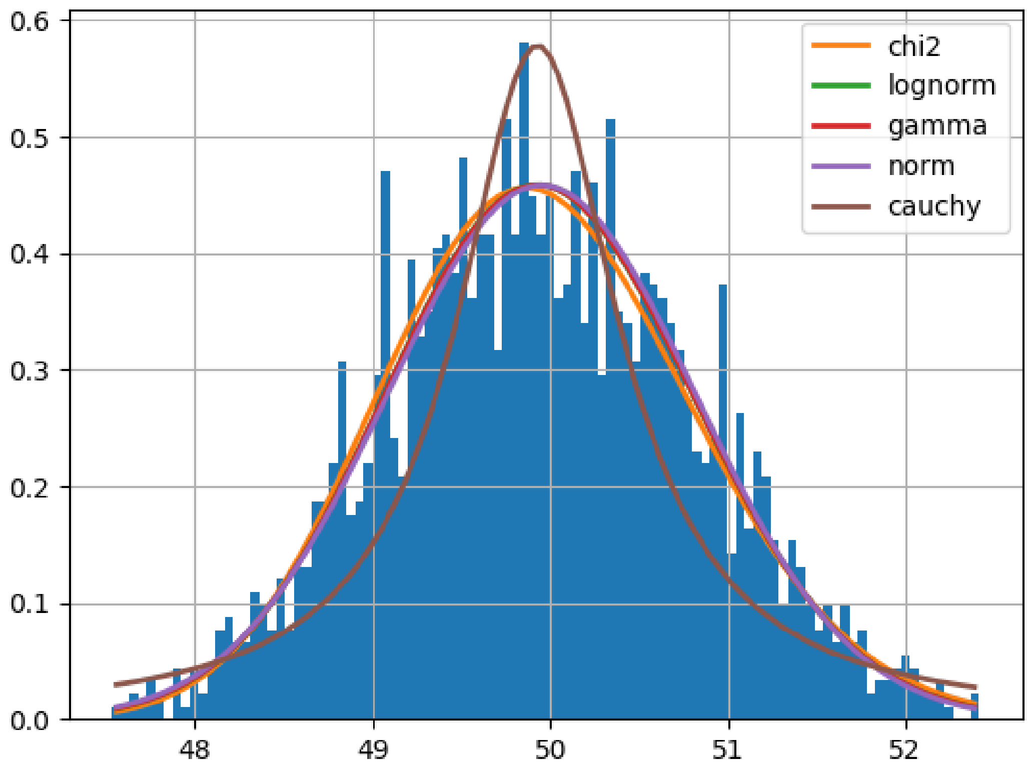
Cari Distribusi yang Tepat

```
2024-05-24 15:36:16.418 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.789795)
2024-05-24 15:36:16.466 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=4.250075)
2024-05-24 15:36:16.779 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.241409)
2024-05-24 15:36:17.071 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.243353)
2024-05-24 15:36:17.139 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.243103)
2024-05-24 15:36:17.172 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.246425)
2024-05-24 15:36:17.218 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=2.445602)
2024-05-24 15:36:17.249 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.997669)
2024-05-24 15:36:17.271 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=2.536461)
2024-05-24 15:36:17.339 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=17.784827)
```

7.0.3 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **chi2**

Dengan parameter: **{'df': 207.26130032526567, 'loc': 41.00882665182942, 'scale': 0.043141978976837}**



[469]:
```
normal_distribution_test(cleaned_data['Appearance'])
```

Nilai p = ***0.08680165559053421***

Data ***Appearance*** Terdistribusi Normal

[470]:
```
normal_distribution_test(cleaned_data['Tannin'])
```

Nilai p = ***0.0003684391558635 98***

Data ***Tannin*** Tidak Terdistribusi Normal

Cari Distribusi yang Tepat

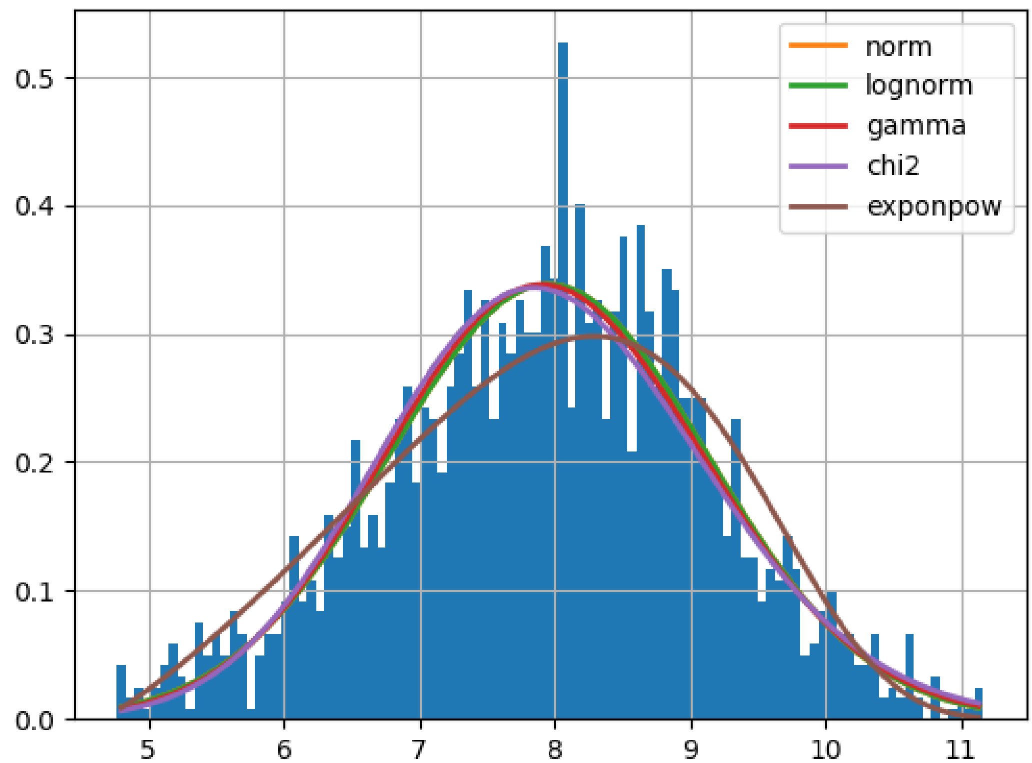
```
2024-05-24 15:36:19.171 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.422658)
2024-05-24 15:36:19.204 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=2.429854)
2024-05-24 15:36:19.845 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.19722)
```

```
2024-05-24 15:36:20.320 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.182535)
2024-05-24 15:36:20.372 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.172008)
2024-05-24 15:36:20.392 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.172006)
2024-05-24 15:36:20.431 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=1.392709)
2024-05-24 15:36:20.477 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.675937)
2024-05-24 15:36:20.497 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=1.44715)
2024-05-24 15:36:20.546 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.216746)
```

7.0.4 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **norm**

Dengan parameter: **{'loc': 7.954611113282607, 'scale': 1.1791332059834063}**



[471]:

```
normal_distribution_test(cleaned_data['Ripeness'])
```

Nilai p = ***1.1732893823168666e-10***

Data ***Ripeness*** Tidak Terdistribusi Normal

Cari Distribusi yang Tepat

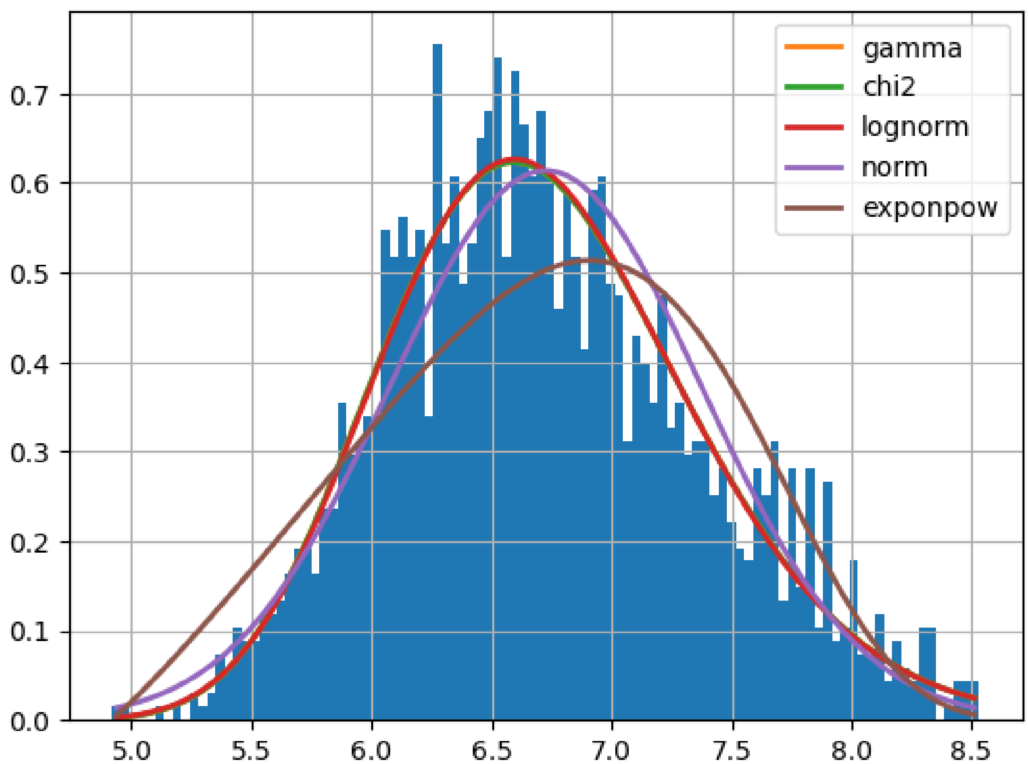
```
2024-05-24 15:36:22.332 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=1.266624)
2024-05-24 15:36:22.350 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=7.956471)
2024-05-24 15:36:22.545 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.411839)
2024-05-24 15:36:22.669 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.411839)
2024-05-24 15:36:22.699 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
```

```
Fitted lognorm distribution with error=0.414979)
2024-05-24 15:36:22.720 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.618173)
2024-05-24 15:36:22.742 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=4.580379)
2024-05-24 15:36:22.767 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=1.782286)
2024-05-24 15:36:22.776 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=4.765421)
2024-05-24 15:36:22.821 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=1.118505)
```

7.0.5 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **gamma**

Dengan parameter: **{'a': 23.81298258586093, 'loc': 3.5462480204780666, 'scale': 0.1335136376149216}**



```
[472]: normal_distribution_test(cleaned_data['Sweetness'])
```

Nilai p = ***3.1812714355178295e-14***

Data ***Sweetness*** Tidak Terdistribusi Normal

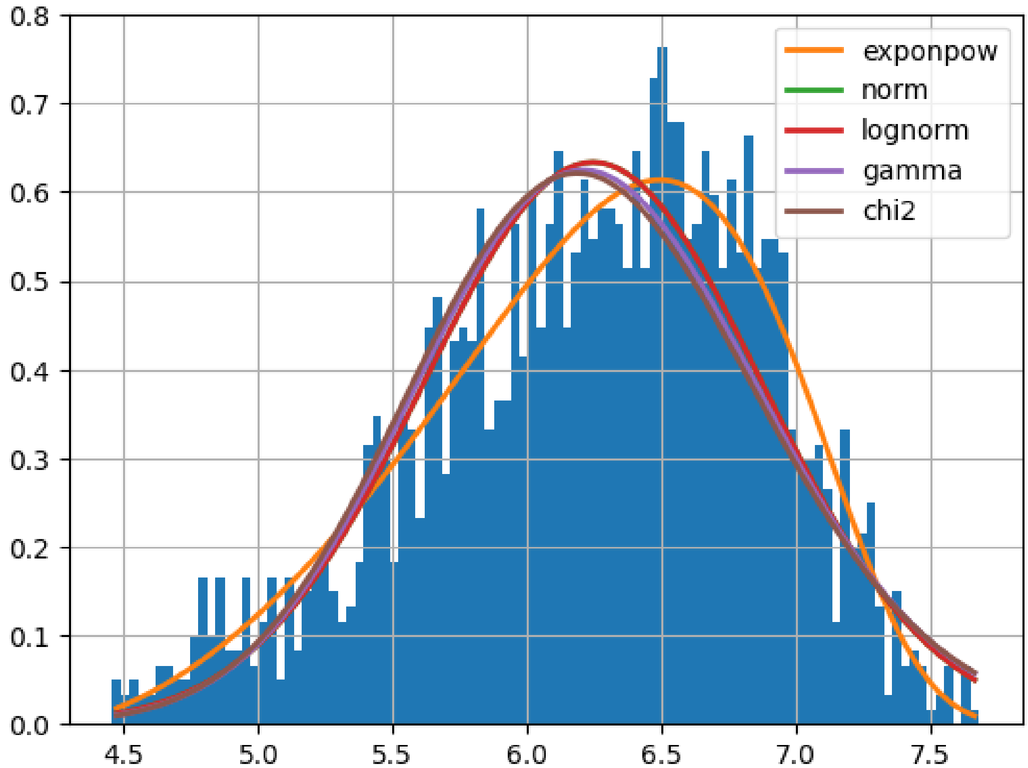
Cari Distribusi yang Tepat

```
2024-05-24 15:36:23.636 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=1.810537)
2024-05-24 15:36:23.648 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=9.243486)
2024-05-24 15:36:24.021 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.451674)
2024-05-24 15:36:24.159 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=0.982294)
2024-05-24 15:36:24.188 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=0.809488)
2024-05-24 15:36:24.202 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=0.809463)
2024-05-24 15:36:24.224 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=4.11175)
2024-05-24 15:36:24.239 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=2.960288)
2024-05-24 15:36:24.254 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=4.802295)
2024-05-24 15:36:24.305 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=0.908295)
```

7.0.6 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **exponpow**

Dengan parameter: **{'b': 2.787877740878511, 'loc': 4.20078255283327, 'scale': 2.644827170409579}**



[473]:
```
normal_distribution_test(cleaned_data['Firmness'])
```

Nilai p = ***7.2707984347899 89e-24***

Data ***Firmness*** Tidak Terdistribusi Normal

Cari Distribusi yang Tepat

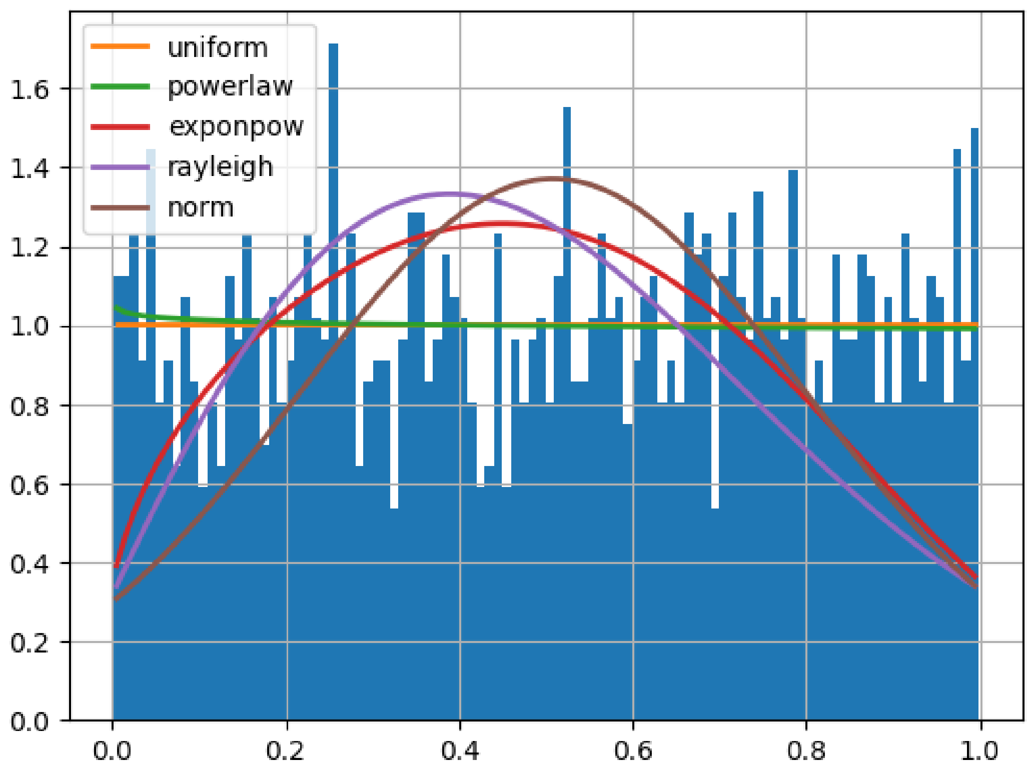
```
2024-05-24 15:36:24.947 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=29.87662)
2024-05-24 15:36:24.959 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=31.724905)
2024-05-24 15:36:25.300 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=14.260422)
2024-05-24 15:36:25.490 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=19.230011)
2024-05-24 15:36:25.519 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
```

```
Fitted lognorm distribution with error=18.915921)
2024-05-24 15:36:25.530 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=18.915876)
2024-05-24 15:36:25.560 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=5.124321)
2024-05-24 15:36:25.576 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=17.976315)
2024-05-24 15:36:25.587 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=5.094852)
2024-05-24 15:36:25.614 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=18.932221)
```

7.0.7 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **uniform**

Dengan parameter: **{'loc': 0.0002540323592254, 'scale': 0.9991769364603657}**



[474]:
```
normal_distribution_test(cleaned_data['Price'])
```

Nilai p = ***0.04199299588799 4766***

Data ***Price*** Tidak Terdistribusi Normal

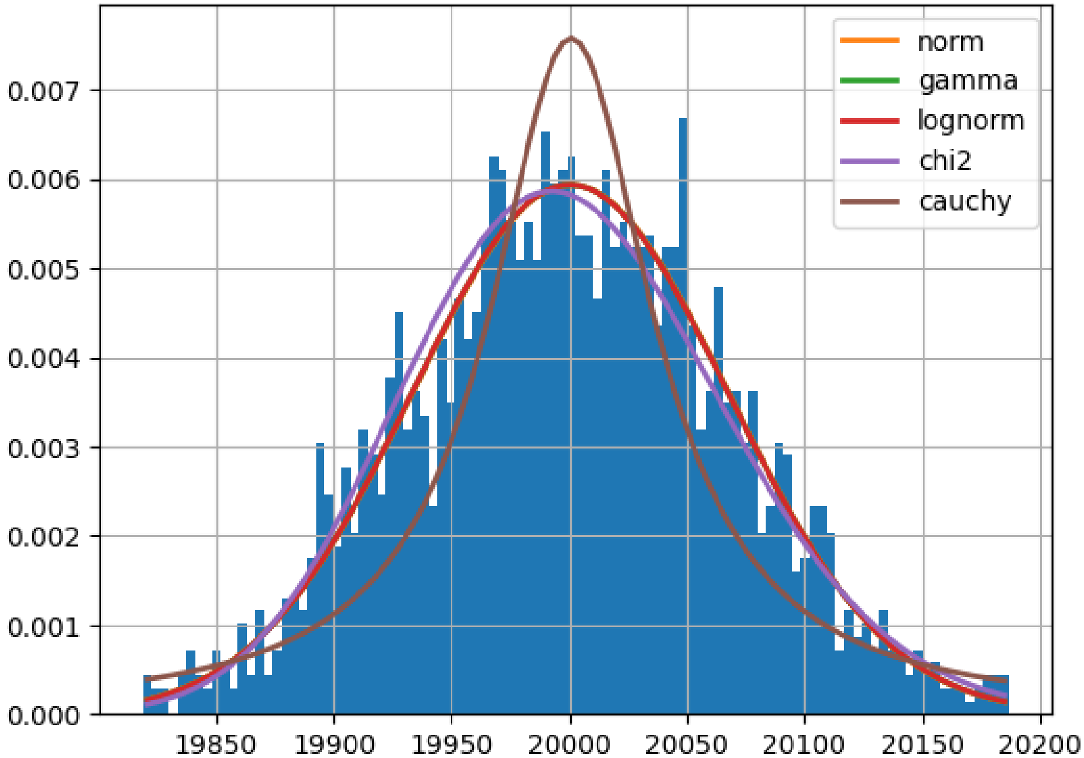
Cari Distribusi yang Tepat

```
2024-05-24 15:36:26.326 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted cauchy distribution with error=0.000116)
2024-05-24 15:36:26.347 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted expon distribution with error=0.000709)
2024-05-24 15:36:26.775 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted exponpow distribution with error=0.001151)
2024-05-24 15:36:26.961 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted chi2 distribution with error=3.2e-05)
2024-05-24 15:36:27.003 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted lognorm distribution with error=2.9e-05)
2024-05-24 15:36:27.015 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted norm distribution with error=2.8e-05)
2024-05-24 15:36:27.028 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted powerlaw distribution with error=0.000398)
2024-05-24 15:36:27.044 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted rayleigh distribution with error=0.000157)
2024-05-24 15:36:27.056 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted uniform distribution with error=0.000413)
2024-05-24 15:36:27.094 | INFO     |
fitter.fitter:_fit_single_distribution:337 -
Fitted gamma distribution with error=2.9e-05)
```

7.0.8 Distribusi Terbaik

Distribusi terbaik berdasarkan Sum of Squares Error adalah: **norm**

Dengan parameter: **{'loc': 20000.7133711723, 'scale': 67.22979588632606}**



8 5. Melakukan test hipotesis 1 sampel

8.1 Hipotesis 1 sampel

- 5.1. Perusahaan menerima beberapa keluhan bahwa buah pisang yang mereka terima akhir-akhir ini cukup asam. Dapatkah anda mengecek apakah rata-rata nilai Acidity di atas 6?
- 5.2. Supplier menjanjikan bahwa rata-rata berat buah pisang adalah 150 gram. Pemilik mencurigai kebenaran hal ini. Apakah rata-rata buah pisang yang mereka kirim tidak bernilai 150 gram?
- 5.3. Periksalah apakah rata-rata panjang buah pisang 10 baris terakhir tidak sama dengan 49!
- 5.4. Apakah proporsi nilai Tannin yang lebih besar dari 8 tidak sama dengan 55% dari total dataset?

8.2 Fungsi penghitung t-score

Untuk menghitung t untuk satu sampel:

$$t = \frac{x_i - \bar{x}}{\frac{s}{\sqrt{n}}}$$

dengan - x : nilai observasi, - $\bar{x}$: nilai rata-rata sampel, - s : standar deviasi sampel, dan - n : jumlah sampel

[475]:
```python
from statistics import pstdev

# Fungsi hitung t-score:
def t_score_1samp(h0, sample_mean, sample_stdev, n_samples):
    return (sample_mean - h0) / (sample_stdev/np.sqrt(n_samples))
```

8.2.1 1. Perusahaan menerima beberapa keluhan bahwa buah pisang yang mereka terima akhir-akhir ini cukup asam. Dapatkah anda mengecek apakah rata-rata nilai Acidity di atas 6?

8.2.2 Langkah-langkah pengujian:

1. $H_0 : \mu = 6$
2. $H_1 : \mu > 6$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji t untuk satu sampel

8.2.3 Langkah lanjutan:

[476]:
```python
# Variabel
mu_0 = 6
alpha = 0.05
acidity_mean = cleaned_data['Acidity'].mean()
acidity_std = cleaned_data['Acidity'].std()
n_samples = len(cleaned_data['Acidity'])

# Hitung derajat kebebasan
df = n_samples - 1

# Hitung critical region
critical_region = t.ppf(1 - alpha, df)

# Hitung t-score
t_score_test = t_score_1samp(mu_0, acidity_mean, acidity_std, n_samples)

# Hitung p-value
p_value = 2 * t.sf(t_score_test, df)

# Perbandingan dengan library scipy
t_score_lib, p_value_lib = stats.ttest_1samp(cleaned_data['Acidity'], 6,␣
 ↪alternative='greater')

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f't-score = ***{t_score_test}***'))
display(Markdown(f'Critical Region = ***{critical_region}***'))
display(Markdown(f'P-value = ***{p_value}***'))
print()
```

```python
# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f't-score (Library) = ***{t_score_lib}***'))
display(Markdown(f'P-value (Library) = ***{p_value_lib}***'))
print()

display(Markdown(f'***6. Kesimpulan***:'))
if t_score_test > critical_region:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $t$-score > critical␣
↪region, yakni {t_score_test:.2f} > {critical_region:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value < alpha, yakni␣
↪{p_value} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

t-score = ***79.71789415333593***

Critical Region = ***1.6456688830453665***

P-value = ***0.0***

Perbandingan dengan Library Scipy:

t-score (Library) = ***79.71789415333596***

P-value (Library) = ***0.0***

6. Kesimpulan:

Tolak H_0 karena nilai t-score > critical region, yakni 79.72 > 1.65

Tolak H_0 karena nilai P-value < alpha, yakni 0.0 < 0.05

Karena H_0 ditolak, maka dapat disimpulkan bahwa rata-rata nilai Acidity di atas 6

8.2.4 2. Supplier menjanjikan bahwa rata-rata berat buah pisang adalah 150 gram. Pemilik mencurigai kebenaran hal ini. Apakah rata-rata buah pisang yang mereka kirim tidak bernilai 150 gram?

8.2.5 Langkah-langkah pengujian:

1. $H_0 : =150 $
2. $H_1 :   150 $
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji t untuk satu sampel

8.2.6 Langkah lanjutan:

[477]:
```
# Variabel
mu_0 = 150
alpha = 0.05
weight_mean = cleaned_data['Weight'].mean()
weight_std = cleaned_data['Weight'].std()
n_samples = len(cleaned_data['Weight'])

# Hitung derajat kebebasan
df = n_samples - 1

# Hitung critical region
critical_region = t.ppf(1 - alpha, df)

# Hitung t-score
t_score_test = t_score_1samp(mu_0, weight_mean, weight_std, n_samples)

# Hitung p-value
p_value = 2 * t.sf(t_score_test, df)

# Perbandingan dengan library scipy
t_score_lib, p_value_lib = stats.ttest_1samp(cleaned_data['Weight'], 150)

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f't-score = ***{t_score_test}***'))
display(Markdown(f'Critical Region = ***{critical_region}***'))
display(Markdown(f'P-value = ***{p_value}***'))
print()

# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f't-score (Library) = ***{t_score_lib}***'))
display(Markdown(f'P-value (Library) = ***{p_value_lib}***'))
print()

display(Markdown(f'***6. Kesimpulan***:'))

if t_score_test > critical_region:
```

```
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $t$-score > critical␣
↪region, yakni {t_score_test:.2f} > {critical_region:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value < alpha, yakni␣
↪{p_value} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

t-score = ***0.7119758108200114***

Critical Region = ***1.6456688830453665***

P-value = ***0.47656856717219553***

Perbandingan dengan Library Scipy:

t-score (Library) = ***0.7119758108200115***

P-value (Library) = ***0.47656856717219553***

6. Kesimpulan:

Gagal Tolak H_0

Gagal Tolak H_0

Karena H_0 gagal ditolak, maka dapat disimpulkan bahwa rata-rata berat pisang yang dikirim adalah benar 150 gram

8.2.7 3. Periksalah apakah rata-rata panjang buah pisang 10 baris terakhir tidak sama dengan 49!

8.2.8 Langkah-langkah pengujian:

1. $H_0 : \bar{x} = 49$
2. $H_1 : \bar{x} \neq 49$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji t untuk satu sampel

8.2.9 Langkah lanjutan:

[478]:
```
# Variabel
mu_0 = 49
alpha = 0.05
```

```python
sample_mean = cleaned_data['Length'].tail(10).mean()
sample_std = cleaned_data['Length'].tail(10).std()
n_samples = 10

# Hitung derajat kebebasan
df = n_samples - 1

# Hitung critical region
critical_region = t.ppf(1 - alpha, df)

# Hitung t-score
t_score_test = t_score_1samp(mu_0, sample_mean, sample_std, n_samples)

# Hitung p-value
p_value = 2 * t.sf(t_score_test, df)

# Perbandingan dengan library scipy
t_score_lib, p_value_lib = stats.ttest_1samp(cleaned_data['Length'].tail(10),␣
↪49)

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f't-score = ***{t_score_test}***'))
display(Markdown(f'Critical Region = ***{critical_region}***'))
display(Markdown(f'P-value = ***{p_value}***'))
print()

# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f't-score (Library) = ***{t_score_lib}***'))
display(Markdown(f'P-value (Library) = ***{p_value_lib}***'))
print()

display(Markdown(f'***6. Kesimpulan***:'))

if t_score_test > critical_region:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $t$-score > critical␣
↪region, yakni {t_score_test:.2f} > {critical_region:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value < alpha, yakni␣
↪{p_value} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

t-score = ***1.2862564266596896***

Critical Region = ***1.8331129326536335***

P-value = ***0.23045572947089393***

Perbandingan dengan Library Scipy:

t-score (Library) = ***1.2862564266596894***

P-value (Library) = ***0.23045572947089393***

6. Kesimpulan:

Gagal Tolak H_0

Gagal Tolak H_0

Karena H_0 gagal ditolak, maka dapat disimpulkan bahwa panjang buah pisang 10 baris terakhir tidak sama dengan 49

8.2.10 4. Apakah proporsi nilai Tannin yang lebih besar dari 8 tidak sama dengan 55% dari total dataset?

1. $H_0 : p = 0.55$
2. $H_1 : p \neq 0.55$
3. $\alpha = 0.05$
4. Uji statistik menggunakan uji Z karena ukuran data > 30

$$Z = \frac{\bar{p} - p_0}{\sqrt{\frac{p_0 q_0}{n}}}$$

8.2.11 Langkah Lanjutan:

[479]:
```
# Variabel
alpha = 0.05
p0 = 55/100
q0 = 1 - p0

tannin = cleaned_data.loc[cleaned_data['Tannin'] > 8, 'Tannin']
tannin_mean = tannin.mean()
tannin_std = pstdev(tannin)
n_samples = len(tannin)
n_population = len(cleaned_data['Tannin'])

pbar = n_samples/n_population
```

```python
# Hitung critical region (Z alpha)
Z_alpha_l = norm.ppf(alpha/2)
Z_alpha_h = norm.ppf(1-(alpha/2))

# Hitung Z-score dengan binomial didekati normal
Z_score = (pbar - p0) / np.sqrt((p0*q0)/n_population)

# Hitung p-value
p_value = 2*norm.cdf(Z_score)

# Perbandingan dengan library scipy
p_value_lib = 2 * (1 - stats.norm.cdf(np.abs(Z_score)))

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f'Z-score = ***{Z_score}***'))
display(Markdown(f'Critical region ($Z_l$)= ***{Z_alpha_l}***'))
display(Markdown(f'Critical region ($Z_h$)= ***{Z_alpha_h}***'))
display(Markdown(f'P-value = ***{p_value}***'))
print()

# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f'P-value (Library) = ***{p_value_lib}***'))
print()

display(Markdown(f'***6. Kesimpulan***:'))
if Z_score > Z_alpha_h or Z_score < Z_alpha_l:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $Z$-score berada di␣
 ↪critical region, yakni {Z_score:.2f} > {Z_alpha_h:.2f} atau {Z_score:.2f} <␣
 ↪{Z_alpha_l:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value < alpha, yakni␣
 ↪{p_value} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

Z-score = ***-3.9058289707451572***

Critical region (Z_l)= ***-1.959963984540054 5***

Critical region (Z_h)= ***1.959963984540054***

P-value = ***9.39029125400427e-05***

Perbandingan dengan Library Scipy:

P-value (Library) = ***9.390291253996708e-05***

6. Kesimpulan:

Tolak H_0 karena nilai Z-score berada di critical region, yakni -3.91 > 1.96 atau -3.91 < -1.96

Tolak H_0 karena nilai P-value < alpha, yakni 9.39029125400427e-05 < 0.05

Karena H_0 ditolak, maka dapat disimpulkan bahwa proporsi nilai Tannin yang lebih besar dari 8 tidak sama dengan 55% dari total dataset

9 6. Melakukan test hipotesis 2 sampel

9.1 Perusahaan ingin membandingkan kualitas buah yang diterima pada paruh awal dan paruh akhir kerjasama. Anda dapat melakukan ini dengan membagi 1 dataset menjadi 2 bagian yang sama panjang.

- 6.1. Anda diminta untuk memeriksa apakah rata-rata acidity dari buah pisang yang disuplai bernilai sama pada kedua kurun waktu tersebut. Bandingkanlah rata-rata appearance pada bagian awal dan akhir.
- 6.2. Apakah rata-rata appearance pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.1 unit?
- 6.3. Apakah variansi dari panjang pisang yang dipasok suplier sama pada bagian awal dan akhir?
- 6.4. Apakah proporsi berat pisang yang lebih dari 150 pada dataset awal lebih besar daripada proporsi di bagian dataset akhir?

9.2 Fungsi-fungsi yang digunakan

t-score didefinisikan oleh rumus berikut:

$$t = \frac{(\bar{x_1} - \bar{x_2}) - d_0}{s_p\sqrt{1/n_1 + 1/n_2}}$$

t-accent:

$$t' = \frac{(\bar{x}_1 - \bar{x}_2) - d_0}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}$$

Varians gabungan (pooled varians):

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

Distribusi f:

$$f = \frac{s_1^2}{s_2^2}$$

z-score:

$$Z = \frac{\hat{p_1} - \hat{p_2}}{\sqrt{(\hat{p}\hat{q}(\frac{1}{n_1} + \frac{1}{n_2}))}}$$

Derajat kebebasan:

$$df = n_1 + n_2 - 2$$

[480]:
```
def t_score(mean1, mean2, d0, sp, n1, n2):
    return ((mean1 - mean2) - d0) / (sp * np.sqrt(1/n1 + 1/n2))

def taccent_score(mean1, mean2, d0, s1, n1, s2, n2):
    return ((mean1 - mean2) - d0) / (np.sqrt(s1**2/n1 + s2**2/n2))

def spsquared(n1, s12, n2, s22):
    return ((n1-1) *s12 + (n2-1) *s22) / (n1+n2-2)

def f_score(s12, s22):
    return s12/s22

def z_score_prop(p1, p2, p, n1, n2):
    return (p1 - p2) / np.sqrt(p * (1-p) * (1/n1 + 1/n2))
```

9.2.1 1. Anda diminta untuk memeriksa apakah rata-rata acidity dari buah pisang yang disuplai bernilai sama pada kedua kurun waktu tersebut.

9.2.2 Langkah-langkah pengujian:

1. $H_0 : *{1}* - {2} = 0 $

   (Rata-rata acidity pada paruh awal sama dengan rata-rata acidity pada paruh akhir)

2. $H_1 : *{1}* - {2} 0 $

   (Rata-rata acidity pada paruh awal berbeda dengan rata-rata acidity pada paruh akhir)

3. $\alpha = 0.05$

4. Uji statistik menggunakan uji t dua sampel karena keduanya berasal dari populasi data yang sama sehingga variansi populasi keduanya akan sama, tetapi tidak diketahui

   $ _1^2 = _2^2 = ^2 $

9.2.3 Langkah lanjutan:

[481]:
```
# Variabel
n = len(cleaned_data)
n_half = n // 2
first_half = cleaned_data['Acidity'][:n_half]
second_half = cleaned_data['Acidity'][n_half:]
mean1 = first_half.mean()
mean2 = second_half.mean()
```

```
std1 = first_half.std()
std2 = second_half.std()
n1 = len(first_half)
n2 = len(second_half)

# Hitung derajat kebebasan
df = n1 + n2 - 2

# Hitung ritical region
t_critical_l = -t.ppf(1 - alpha / 2, df)
t_critical_h = t.ppf(1 - alpha / 2, df)

# Hitung t-score
sp = np.sqrt(spsquared(n1, std1**2, n2, std2**2))
t_score_test = t_score(mean1, mean2, 0, sp, n1, n2)

# Hitung p-value
p_value_manual = 2 * (1 - t.cdf(abs(t_score_test), df))

# Hitung p-value menggunakan scipy
t_stat, p_value_scipy = ttest_ind(first_half, second_half, equal_var=True)

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f'T-score = ***{t_score_test}***'))
display(Markdown(f'Critical region ($t_l$)= ***{t_critical_l}***'))
display(Markdown(f'Critical region ($t_h$)= ***{t_critical_h}***'))
display(Markdown(f'P-value (manual) = ***{p_value}***'))
print()


# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f'P-value (Library) = ***{p_value_lib}***'))
print()


display(Markdown(f'***6. Kesimpulan***:'))
if t_score_test > t_critical_h or t_score_test < t_critical_l:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $t$-score berada di␣
↪critical region, yakni {t_score_test:.2f} > {t_critical_h:.2f} atau␣
↪{t_score_test:.2f} < {t_critical_l:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value_manual < alpha:
```

```
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value (manual) <␣
↪alpha, yakni {p_value_manual} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value_scipy < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value (scipy) <␣
↪alpha, yakni {p_value_scipy} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

T-score = ***-1.32524878130604***

Critical region (t_l)= ***-1.9612340659350982***

Critical region (t_h)= ***1.9612340659350982***

P-value (manual) = ***9.39029125400427e-05***

Perbandingan dengan Library Scipy:

P-value (Library) = ***9.390291253996708e-05***

6. Kesimpulan:

Gagal Tolak H_0

Gagal Tolak H_0

Gagal Tolak H_0

Karena H_0 gagal ditolak, dapat disimpulkan bahwa rata-rata acidity pada paruh awal sama dengan rata-rata acidity pada paruh akhir

9.2.4 2. Bandingkanlah rata-rata appearance pada bagian awal dan akhir. Apakah rata-rata appearance pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.1 unit?

9.2.5 Langkah-langkah pengujian:

1. $H_0 : *{1}* - {2} = 0.1 $

   (Rata-rata appearance pada dataset bagian awal lebih besar daripada bagian akhir sebesar 0.1 unit)

2. $H_1 : *{1}* - {2} 0.1 $

   (Rata-rata appearance pada dataset bagian awal tidak lebih besar daripada bagian akhir sebesar 0.1 unit)

3. $\alpha = 0.05$
4. Uji statistik menggunakan uji t dua sampel karena keduanya berasal dari populasi data yang sama sehingga variansi populasi keduanya akan sama, tetapi tidak diketahui

   $ _1^2 = _2^2 = ^2 $

9.2.6 Langkah lanjutan:

[482]:
```
# Variabel
n = len(cleaned_data)
n_half = n // 2
first_half = cleaned_data['Acidity'][:n_half]
second_half = cleaned_data['Acidity'][n_half:]
mean1 = first_half.mean()
mean2 = second_half.mean()
std1 = first_half.std()
std2 = second_half.std()
n1 = len(first_half)
n2 = len(second_half)

# Hitung derajat kebebasan
df = n1 + n2 - 2

# Hitung ritical region
t_critical_l = -t.ppf(1 - alpha / 2, df)
t_critical_h = t.ppf(1 - alpha / 2, df)

# Hitung t-score
sp = np.sqrt(spsquared(n1, std1**2, n2, std2**2))
t_score_test = t_score(mean1, mean2, 0.1, sp, n1, n2)

# Hitung p-value
p_value_manual = 2 * (1 - t.cdf(abs(t_score_test), df))

# Hitung p-value menggunakan scipy
t_stat, p_value_scipy = ttest_ind(first_half, second_half, equal_var=True)
t_stat_diff, p_value_scipy_diff = ttest_ind(first_half, second_half,
 ↪equal_var=True, alternative='two-sided')


# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f'T-score = ***{t_score_test}***'))
display(Markdown(f'Critical region ($t_l$)= ***{t_critical_l}***'))
display(Markdown(f'Critical region ($t_h$)= ***{t_critical_h}***'))
display(Markdown(f'P-value (manual)= ***{p_value_manual}***'))
print()
```

```python
# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f'P-value (scipy) = ***{p_value_lib}***'))
print()

display(Markdown(f'***6. Kesimpulan***:'))
if t_score_test > t_critical_h or t_score_test < t_critical_l:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $t$-score berada di␣
 ↪critical region, yakni {t_score_test:.2f} > {t_critical_h:.2f} atau␣
 ↪{t_score_test:.2f} < {t_critical_l:.2f}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value_manual < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value (manual) <␣
 ↪alpha, yakni {p_value_manual} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value_scipy < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value (scipy,␣
 ↪two-sided) < alpha, yakni {p_value_scipy} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))

if p_value_scipy_diff < alpha:
    display(Markdown(f'***Tolak*** $H_0$ karena nilai $P$-value (scipy, with␣
 ↪difference) < alpha, yakni {p_value_scipy_diff} < {alpha}'))
else:
    display(Markdown(f'***Gagal Tolak*** $H_0$'))
```

5. Nilai Uji Statistik:

T-score = ***-3.3163606016374785***

Critical region (t_l)= ***-1.9612340659350982***

Critical region (t_h)= ***1.9612340659350982***

P-value (manual)= ***0.0009294508661958911***

Perbandingan dengan Library Scipy:

P-value (scipy) = ***9.390291253996708e-05***

6. Kesimpulan:

Tolak H_0 karena nilai t-score berada di critical region, yakni -3.32 > 1.96 atau -3.32 < -1.96

Tolak H_0 karena nilai P-value (manual) < alpha, yakni 0.0009294508661958911 < 0.05

Gagal Tolak H_0

Gagal Tolak H_0

Karena H_0 ditolak, dapat disimpulkan bahwa rata-rata appearance pada bagian awal tidak lebih besar daripada bagian akhir sebesar 0.1 unit

9.2.7 3. Apakah variansi dari panjang pisang yang dipasok suplier sama pada bagian awal dan akhir?

9.2.8 Langkah Pengujian :

1. $H_0 : _1^2 - _2^2 = 0 $

(Varian dari panjang pisang yang dipasok supplier sama pada bagian awal dan akhir)

2. $H_1 : \sigma_1^2 - \sigma_1^2 \neq 0$

(Varian dari panjang pisang yang dipasok supplier tidak sama pada bagian awal dan akhir)

3. $\alpha = 0.05$
4. Uji statistik menggunakan uji distribusi f. Jika nilai f lebih besar dari nilai kritis, maka H_0 ditolak

9.2.9 Langkah lanjutan:

[483]:
```
# Inisiasi Variabel
n = len(cleaned_data)
n_half = n // 2
first_half = cleaned_data['Length'][:n_half]
second_half = cleaned_data['Length'][n_half:]

# Hitung variansi dari masing-masing setengah data
var1 = np.var(first_half, ddof=1)
var2 = np.var(second_half, ddof=1)

# Hitung statistik F
F = f_score(var1, var2) if var1 > var2 else f_score(var2, var1)

# Derajat kebebasan
df1 = len(first_half) - 1
df2 = len(second_half) - 1

# Nilai kritis untuk alpha = 0.05 (dua arah)
alpha = 0.05
F_critical_low = f.ppf(alpha / 2, df1, df2)
F_critical_high = f.ppf(1 - alpha / 2, df1, df2)
```

```python
# Tes dengan scipy
f_lib = np.var(first_half, ddof=1)/np.var(second_half, ddof=1)
p_value_lib = 1 - f.cdf(f_lib, df1, df2)


# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f'F (manual)= ***{F}***'))
display(Markdown(f'Critical region ($F_{{\\text{{awal}}}}$) =␣
↪**{F_critical_low}**'))
display(Markdown(f'Critical region ($F_{{\\text{{akhir}}}}$) =␣
↪**{F_critical_high}**'))
print()

# Perbandingan dengan library scipy
display(Markdown(f'***Perbandingan dengan Library Scipy***:'))
display(Markdown(f'F (scipy) = ***{f_lib}***'))
print()

# Ambil keputusan
display(Markdown(f'***6. Kesimpulan***:'))
if F < F_critical_low or F > F_critical_high:
    display(Markdown(f'***Tolak*** $H_0$: variansi setengah pertama dan␣
↪setengah kedua berbeda.'))
else:
    display(Markdown(f'***Gagal tolak*** $H_0$: variansi setengah pertama dan␣
↪setengah kedua sama.'))
```

5. Nilai Uji Statistik:

F (manual)= ***1.0584959287083215***

Critical region (F_{awal}) = **0.8795805289566965**

Critical region (F_{akhir}) = **1.1369004918310361**

Perbandingan dengan Library Scipy:

F (scipy) = ***0.9447367466214973***

6. Kesimpulan:

Gagal tolak H_0: variansi setengah pertama dan setengah kedua sama.

H_0 **GAGAL DITOLAK** , artinya variansi panjang pisang yang dipasok supplier tidak sama pada bagian awal dan akhir

9.2.10 4. Apakah proporsi berat pisang yang lebih dari 150 pada dataset awal lebih besar daripada proporsi di bagian dataset akhir?

9.2.11 Langkah Pengujian:

1. $H_0 : p_{awal} = p_{akhir} $

(Proporsi berat pisang yang lebih dari 150 pada dataset awal sama dengan proporsi di bagian dataset akhir)

2. $H_1 : p_{awal} > p_{akhir} $

(Proporsi berat pisang yang lebih dari 150 pada dataset awal lebih besar daripada proporsi di bagian dataset akhir)

3. $ = 0.05 $

4. Menggunakan uji proporsi dua sampel dengan z-score dan distribusi normal (karena ukuran data > 30)

9.2.12 Langkah lanjutan:

[484]:
```python
# Inisiasi Variabel
n = len(cleaned_data["Weight"])
n_half = n // 2

first_half = cleaned_data["Weight"][:n_half]
second_half = cleaned_data["Weight"][n_half:]

selected_first_half = first_half[first_half > 150]
selected_second_half = second_half[second_half > 150]

x1 = len(selected_first_half)
x2 = len(selected_second_half)
n1 = len(first_half)
n2 = len(second_half)

p1 = x1 / n1
p2 = x2 / n2

p = (x1 + x2) / (n1 + n2)
q = 1 - p

# Hitung statistik Z
Z = z_score_prop(p1, p2, p, n1, n2)

# Calculate p-value
p_value =  (1 - norm.cdf(Z))

# hitung critical value untuk alpha = 0.05
alpha = 0.05
```

```python
Z_critical = norm.ppf(alpha)

# Tampilkan hasil
display(Markdown(f'***5. Nilai Uji Statistik***:'))
display(Markdown(f'Z = ***{Z}***'))
display(Markdown(f'Critical region ($Z$) = ***{Z_critical}***'))
display(Markdown(f'P-value = ***{p_value}***'))
print()

Z_statsmodels, p_value_statsmodels = proportions_ztest([x1, x2], [n1, n2])

# Tampilkan hasil
display(Markdown(f'***Hasil menggunakan statsmodels:***'))
display(Markdown(f'Z-score (statistik Z): ***{Z_statsmodels}***'))
display(Markdown(f'P-value: ***{p_value_statsmodels}***'))
print()


# Ambil keputusan
display(Markdown(f'***6. Kesimpulan***:'))
if Z < Z_critical:
    display(Markdown(f'***Tolak*** hipotesis nol: proporsi panjang buah di setengah pertama lebih besar dari setengah kedua.'))
else:
    display(Markdown(f'***Gagal tolak*** hipotesis nol: proporsi panjang buah di setengah pertama sama dengan setengah kedua.'))

if p_value < alpha:
    display(Markdown(f'***Tolak*** hipotesis nol: proporsi panjang buah di setengah pertama lebih besar dari setengah kedua.'))
else:
    display(Markdown(f'***Gagal tolak*** hipotesis nol: proporsi panjang buah di setengah pertama sama dengan setengah kedua.'))
```

5. Nilai Uji Statistik:

Z = ***1.5037056842668968***

Critical region (Z) = ***-1.6448536269514729***

P-value = ***0.06632858248913354***

Hasil menggunakan statsmodels:

Z-score (statistik Z): ***1.5037056842668968***

P-value: ***0.13265716497826716***

6. Kesimpulan:

Gagal tolak hipotesis nol: proporsi panjang buah di setengah pertama sama dengan setengah kedua.

Gagal tolak hipotesis nol: proporsi panjang buah di setengah pertama sama dengan setengah kedua.

H_0 **GAGAL DITOLAK** , artinya proporsi panjang buah di setengah data pertama sama dengan proporsi buah di setengah data kedua