K01-T1-IF2220-13520130

April 16, 2022

1 Tugas Besar IF2220 Probabilitas dan Statistika

1.1 Penarikan Kesimpulan dan Pengujian Hipotesis

1.1.1 Kelas: K01

1.1.2 Dibuat oleh:

1. Nelsen Putra - 13520130
2. Willy Wilsen - 13520160

1.1.3 Tujuan

- Mahasiswa memahami dan dapat menyelesaikan persoalan distribusi peluang variabel random diskrit dan kontinu.
- Mahasiswa mampu menyelesaikan persoalan untuk menarik kesimpulan mengenai parameter populasi yang diperoleh dari data hasil eksperimen.
- Mahasiswa mampu menyelesaikan persoalan pengujian hipotesis. ### Deskripsi Tugas Diberikan sebuah data water_potability.csv yang dapat diakses pada Dataset Tugas Besar IF2220. water_potability.csv merupakan data metrik kualitas air yang mengandung 11 kolom sebagai berikut:

1. id
2. pH
3. Hardness
4. Solids
5. Chloramines
6. Sulfate
7. Conductivity
8. OrganicCarbon
9. Trihalomethanes
10. Turbidity
11. Potability

Kolom 2-10 adalah kolom atribut (non-target), sedangkan kolom 11 adalah kolom target. Kelompok diminta untuk melakukan analisis statistika sebagai berikut:

1.2 Tahap 0 - Persiapan

1.2.1 Import Library

[ ]:
```
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
import scipy
from scipy import stats as st
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2.2 Pembacaan Data water_potability.csv

Membaca file spesifikasi yang diberikan dan memberikan nama setiap kolom.

[ ]:
```
column_names = ["id", "pH", "Hardness", "Solids", "Chloramines", "Sulfate",␣
 ↪"Conductivity", "OrganicCarbon", "Trihalomethanes", "Turbidity",␣
 ↪"Potability"]
df = pd.read_csv("water_potability.csv", header = None, names = column_names)
df
```

[ ]:

	id	pH	Hardness	Solids	Chloramines	Sulfate	\
0	1	8.316766	214.373394	22018.417441	8.059332	356.886136	
1	2	9.092223	181.101509	17978.986339	6.546600	310.135738	
2	3	5.584087	188.313324	28748.687739	7.544869	326.678363	
3	4	10.223862	248.071735	28749.716544	7.513408	393.663396	
4	5	8.635849	203.361523	13672.091764	4.563009	303.309771	
…	…	…	…	…	…	…	
2005	2006	8.197353	203.105091	27701.794055	6.472914	328.886838	
2006	2007	8.989900	215.047358	15921.412018	6.297312	312.931022	
2007	2008	6.702547	207.321086	17246.920347	7.708117	304.510230	
2008	2009	11.491011	94.812545	37188.826022	9.263166	258.930600	
2009	2010	6.069616	186.659040	26138.780191	7.747547	345.700257	

	Conductivity	OrganicCarbon	Trihalomethanes	Turbidity	Potability
0	363.266516	18.436524	100.341674	4.628771	0
1	398.410813	11.558279	31.997993	4.075075	0
2	280.467916	8.399735	54.917862	2.559708	0
3	283.651634	13.789695	84.603556	2.672989	0
4	474.607645	12.363817	62.798309	4.401425	0
…	…	…	…	…	…
2005	444.612724	14.250875	62.906205	3.361833	1
2006	390.410231	9.899115	55.069304	4.613843	1
2007	329.266002	16.217303	28.878601	3.442983	1
2008	439.893618	16.172755	41.558501	4.369264	1
2009	415.886955	12.067620	60.419921	3.669712	1

```
[2010 rows x 11 columns]
```

1.3 Tahap 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. Ditambahkan juga perhitungan jumlah data untuk setiap kolom.

[ ]:
```python
# Prosedur untuk menulis deskripsi statistik dari kolom c pada tabel
def printStatistic(c):
    print("count\t\t\t: {}".format(len(c)))
    print("mean\t\t\t: {}".format(c.mean()))
    print("median\t\t\t: {}".format(c.median()))
    print("modus\t\t\t:", end = " ")
    if (len(c.mode() > 1)):
        print("seluruh data adalah modus")
    else:
        print("{}".format(c.mode()))
    print("standar deviasi\t\t: {}".format(c.std()))
    print("variansi\t\t: {}".format(c.std()**2))
    print("range\t\t\t: {}".format(c.max()-c.min()))
    print("nilai minimum\t\t: {}".format(c.min()))
    print("nilai maksimum\t\t: {}".format(c.max()))
    quartiles = c.quantile([0,0.25,0.5,0.75,1])
    for i,quartile in enumerate(quartiles):
        print("quartil-{}\t\t: {}".format(i,quartile))
    print("IQR\t\t\t: {}".format(quartiles[0.75]-quartiles[0.25]))
    print("skewness\t\t: {}".format(c.skew()))
    print("kurtosis\t\t: {}".format(st.kurtosis(c, fisher=False)))
```

1.3.1 1. id

Id tidak perlu ditulis deskripsi statistikanya karena id merepresentasikan identifier yang unik untuk tiap data.

1.3.2 2. pH

[ ]:
```python
pH = df["pH"]
printStatistic(pH)
```

```
count                   : 2010
mean                    : 7.0871927687138285
median                  : 7.029490455474185
modus                   : seluruh data adalah modus
standar deviasi         : 1.5728029470456655
variansi                : 2.4737091102355304
```

```
range                   : 13.7725009497978
nilai minimum           : 0.2274990502021987
nilai maksimum          : 13.999999999999998
quartil-0               : 0.2274990502021987
quartil-1               : 6.09078502142353
quartil-2               : 7.029490455474185
quartil-3               : 8.053006240791538
quartil-4               : 13.999999999999998
IQR                     : 1.9622212193680078
skewness                : 0.04853451405270669
kurtosis                : 3.622362158216349
```

1.3.3 3. Hardness

[ ]:
```
Hardness = df["Hardness"]
printStatistic(Hardness)
```

```
count                   : 2010
mean                    : 195.96920903783524
median                  : 197.20352491941043
modus                   : seluruh data adalah modus
standar deviasi         : 32.643165859429864
variansi                : 1065.5762773262475
range                   : 243.84589036652147
nilai minimum           : 73.4922336890611
nilai maksimum          : 317.33812405558257
quartil-0               : 73.4922336890611
quartil-1               : 176.74065667669896
quartil-2               : 197.20352491941043
quartil-3               : 216.44758866727156
quartil-4               : 317.33812405558257
IQR                     : 39.7069319905726
skewness                : -0.08532104172868622
kurtosis                : 3.521190648776977
```

1.3.4 4. Solids

[ ]:
```
Solids = df["Solids"]
printStatistic(Solids)
```

```
count                   : 2010
mean                    : 21904.673439053095
median                  : 20926.88215534375
modus                   : seluruh data adalah modus
standar deviasi         : 8625.397911190576
variansi                : 74397489.12637074
range                   : 56167.72980146483
nilai minimum           : 320.942611274359
```

```
nilai maksimum          : 56488.67241273919
quartil-0               : 320.942611274359
quartil-1               : 15614.412961614333
quartil-2               : 20926.88215534375
quartil-3               : 27170.534648603603
quartil-4               : 56488.67241273919
IQR                     : 11556.12168698927
skewness                : 0.5910113724580447
kurtosis                : 3.333498156306705
```

1.3.5 5. Chloramines

[ ]:
```
Chloramines = df["Chloramines"]
printStatistic(Chloramines)
```

```
count                   : 2010
mean                    : 7.134322344600104
median                  : 7.1420143046226645
modus                   : seluruh data adalah modus
standar deviasi         : 1.5852140982642102
variansi                : 2.512903737335613
range                   : 11.736129095114823
nilai minimum           : 1.3908709048851806
nilai maksimum          : 13.127000000000002
quartil-0               : 1.3908709048851806
quartil-1               : 6.138326387572855
quartil-2               : 7.1420143046226645
quartil-3               : 8.109933216133502
quartil-4               : 13.127000000000002
IQR                     : 1.9716068285606472
skewness                : 0.013003497779569528
kurtosis                : 3.5454318545555785
```

1.3.6 6. Sulfate

[ ]:
```
Sulfate = df["Sulfate"]
printStatistic(Sulfate)
```

```
count                   : 2010
mean                    : 333.211376415189
median                  : 332.2141128069568
modus                   : seluruh data adalah modus
standar deviasi         : 41.21111102560979
variansi                : 1698.3556719651367
range                   : 352.03064230599716
nilai minimum           : 129.00000000000003
nilai maksimum          : 481.0306423059972
quartil-0               : 129.00000000000003
```

```
quartil-1               : 307.6269864860709
quartil-2               : 332.2141128069568
quartil-3               : 359.26814739141554
quartil-4               : 481.0306423059972
IQR                     : 51.641160905344634
skewness                : -0.04572780443653543
kurtosis                : 3.7819149219038866
```

1.3.7 7. Conductivity

[ ]:
```
Conductivity = df["Conductivity"]
printStatistic(Conductivity)
```

```
count                   : 2010
mean                    : 426.47670835257907
median                  : 423.43837202443706
modus                   : seluruh data adalah modus
standar deviasi         : 80.70187180729437
variansi                : 6512.792113200973
range                   : 551.7228828031471
nilai minimum           : 201.6197367551575
nilai maksimum          : 753.3426195583046
quartil-0               : 201.6197367551575
quartil-1               : 366.61921929632433
quartil-2               : 423.43837202443706
quartil-3               : 482.2097724598859
quartil-4               : 753.3426195583046
IQR                     : 115.5905531635616
skewness                : 0.26801233302645316
kurtosis                : 2.760400057844864
```

1.3.8 8. OrganicCarbon

[ ]:
```
OrganicCarbon = df["OrganicCarbon"]
printStatistic(OrganicCarbon)
```

```
count                   : 2010
mean                    : 14.357939902048074
median                  : 14.323285610653329
modus                   : seluruh data adalah modus
standar deviasi         : 3.3257700016987197
variansi                : 11.060746104199103
range                   : 24.80670661116602
nilai minimum           : 2.1999999999999886
nilai maksimum          : 27.00670661116601
quartil-0               : 2.1999999999999886
quartil-1               : 12.122530374047727
quartil-2               : 14.323285610653329
```

```
quartil-3               : 16.683561746173808
quartil-4               : 27.00670661116601
IQR                     : 4.561031372126081
skewness                : -0.02021975629181238
kurtosis                : 3.027957691493332
```

1.3.9 9. Trihalomethanes

[ ]:
```
Trihalomethanes = df["Trihalomethanes"]
printStatistic(Trihalomethanes)
```

```
count                   : 2010
mean                    : 66.40071666307466
median                  : 66.48204080309809
modus                   : seluruh data adalah modus
standar deviasi         : 16.08110898232513
variansi                : 258.60206610141796
range                   : 115.4229870670162
nilai minimum           : 8.577012932983806
nilai maksimum          : 124.0
quartil-0               : 8.577012932983806
quartil-1               : 55.94999302803186
quartil-2               : 66.48204080309809
quartil-3               : 77.2946128060674
quartil-4               : 124.0
IQR                     : 21.344619778035543
skewness                : -0.05138268451619478
kurtosis                : 3.2194788089667044
```

1.3.10 10. Turbidity

[ ]:
```
Turbidity = df["Turbidity"]
printStatistic(Turbidity)
```

```
count                   : 2010
mean                    : 3.9694969126303676
median                  : 3.967373963531836
modus                   : seluruh data adalah modus
standar deviasi         : 0.7804710407083957
variansi                : 0.6091350453844462
range                   : 5.044748555990993
nilai minimum           : 1.45
nilai maksimum          : 6.494748555990993
quartil-0               : 1.45
quartil-1               : 3.442881623557439
quartil-2               : 3.967373963531836
quartil-3               : 4.5146627202018825
quartil-4               : 6.494748555990993
```

```
IQR                  : 1.0717810966444437
skewness             : -0.03226597968019271
kurtosis             : 2.9473094836957947
```

1.3.11 11. Potability

Potability tidak perlu ditulis deskripsi statistikanya karena potability adalah kolom yang merupakan target.

1.4 Tahap 2 - Visualisasi

Membuat visualisasi plot distribusi, dalam bentuk histogram dan boxplot untuk setiap kolom numerik.

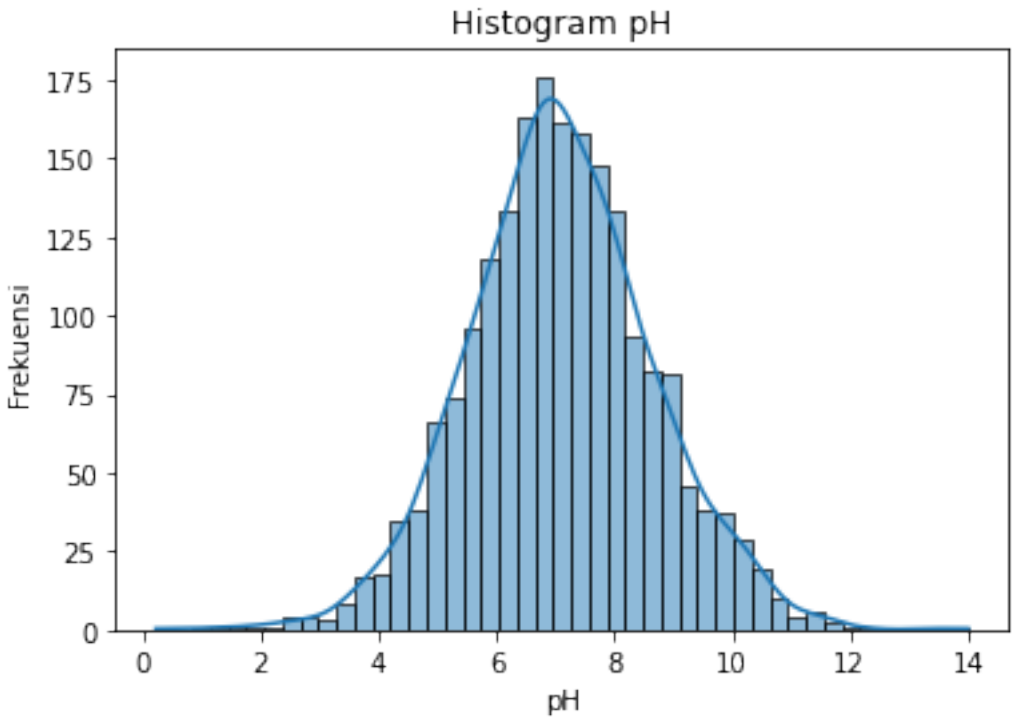
1.4.1 1. id

Kolom id tidak perlu divisualisasi karena selalu unik untuk setiap data.

1.4.2 2. pH

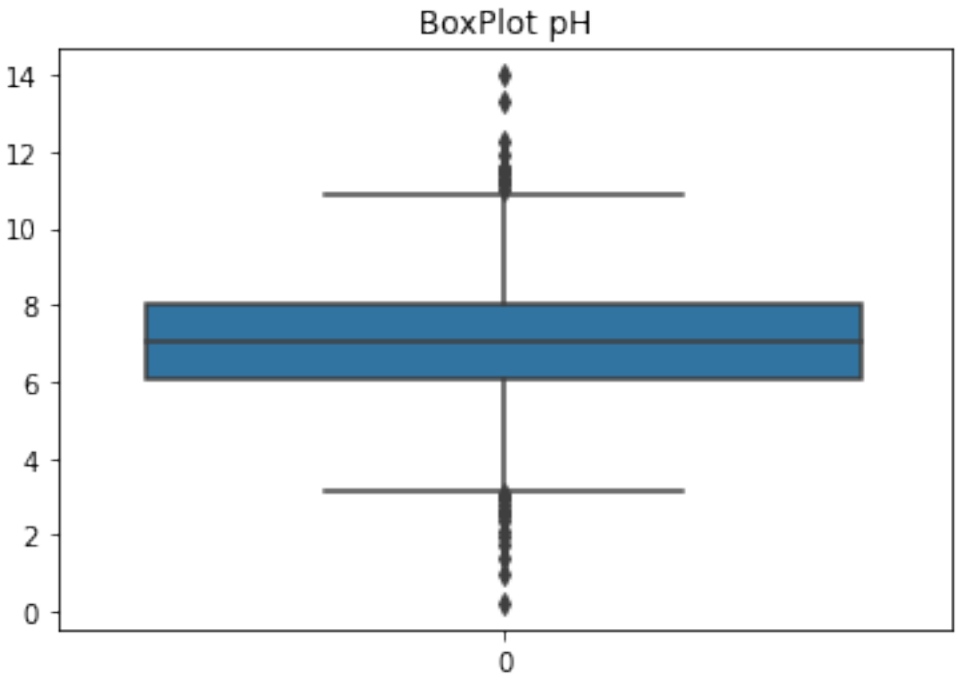
[ ]:
```
# Histogram kolom pH

HistpH = sns.histplot(pH, kde = True)
HistpH.set_title("Histogram pH")
HistpH.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom pH

BoxPlotpH= sns.boxplot(data = pH)
BoxPlotpH.set_title("BoxPlot pH")
```

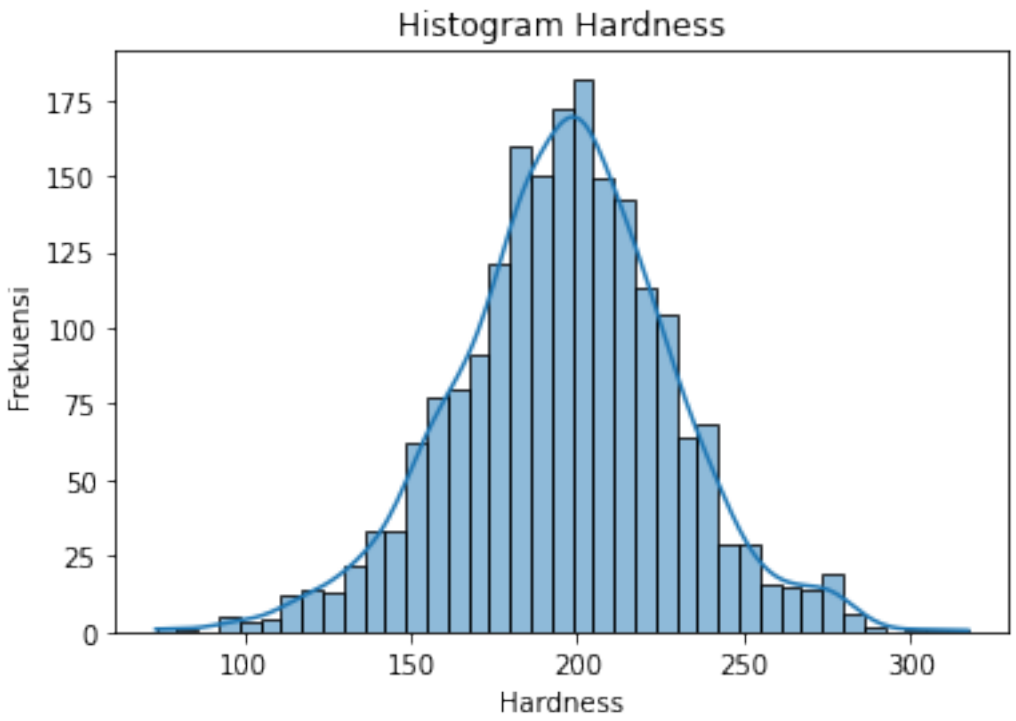
[ ]:
```
Text(0.5, 1.0, 'BoxPlot pH')
```



1.4.3 3. Hardness

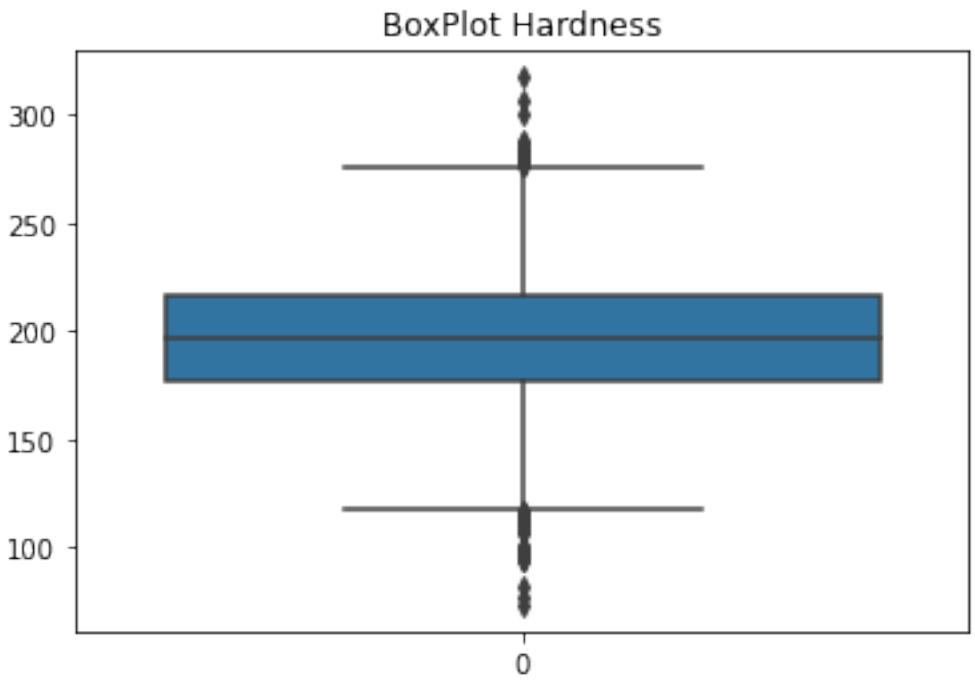
[ ]:
```
# Histogram kolom Hardness

HistHardness = sns.histplot(Hardness, kde = True)
HistHardness.set_title("Histogram Hardness")
HistHardness.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Hardness

BoxPlotHardness = sns.boxplot(data = Hardness)
BoxPlotHardness.set_title("BoxPlot Hardness")
```

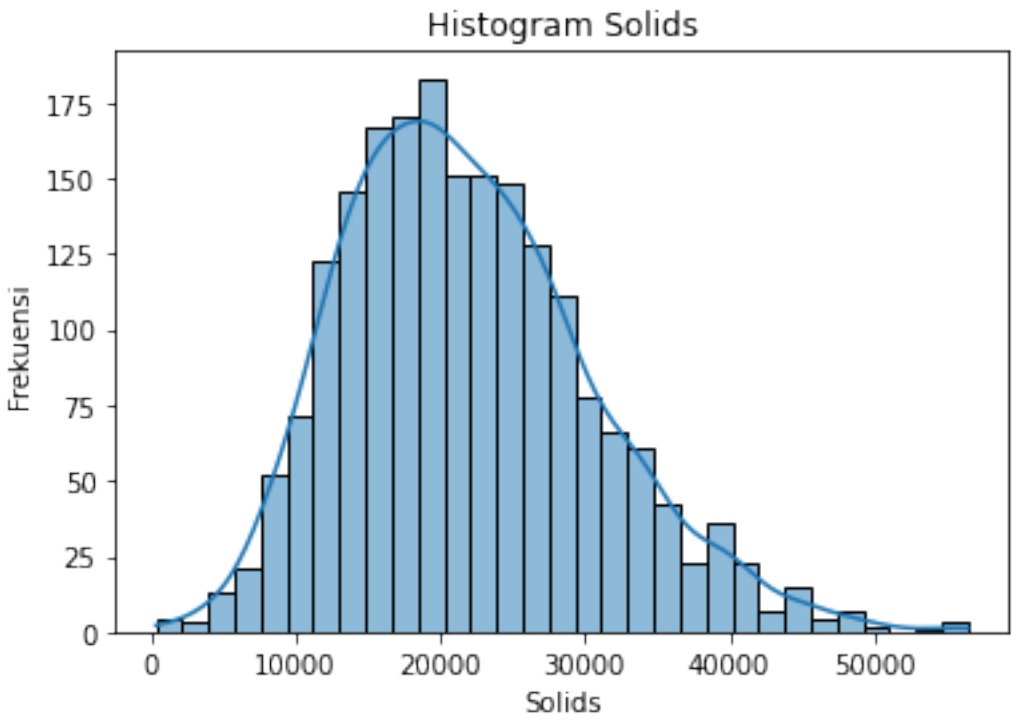
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Hardness')
```



1.4.4 4. Solids

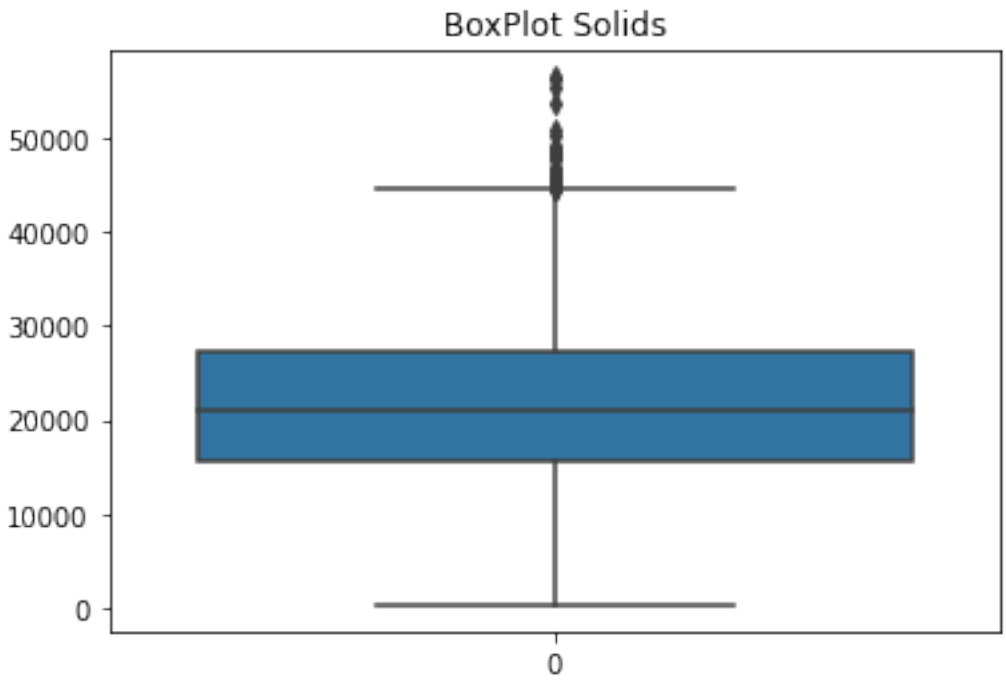
[ ]:
```
# Histogram kolom Solids

HistSolids = sns.histplot(Solids, kde = True)
HistSolids.set_title("Histogram Solids")
HistSolids.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Solids

BoxPlotSolids = sns.boxplot(data = Solids)
BoxPlotSolids.set_title("BoxPlot Solids")
```

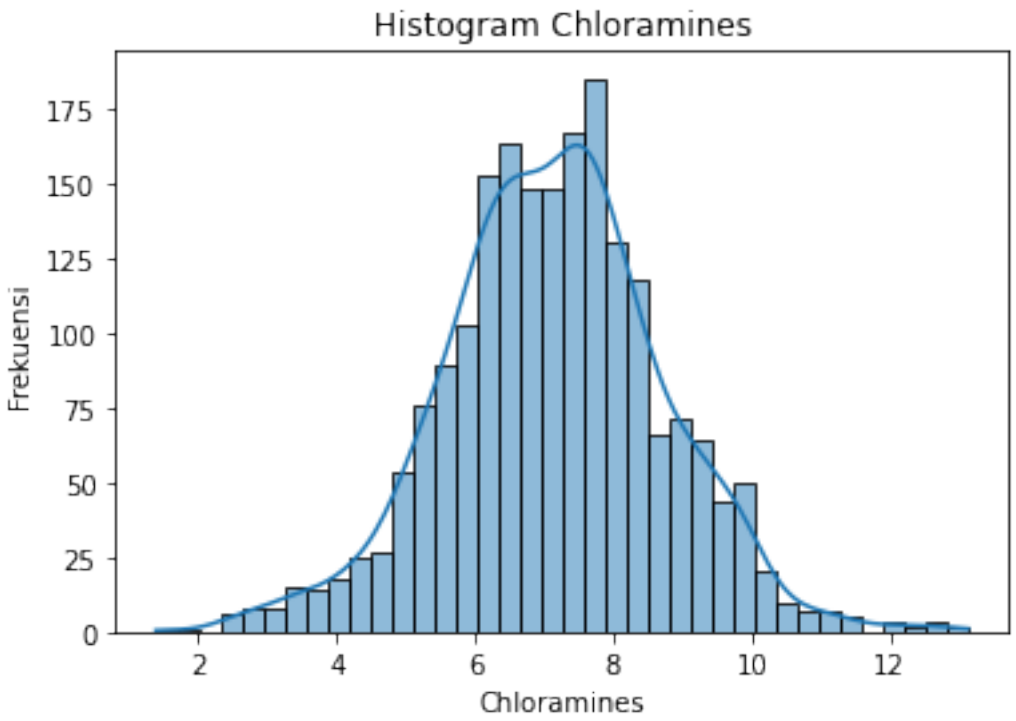
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Solids')
```



1.4.5 5. Chloramines

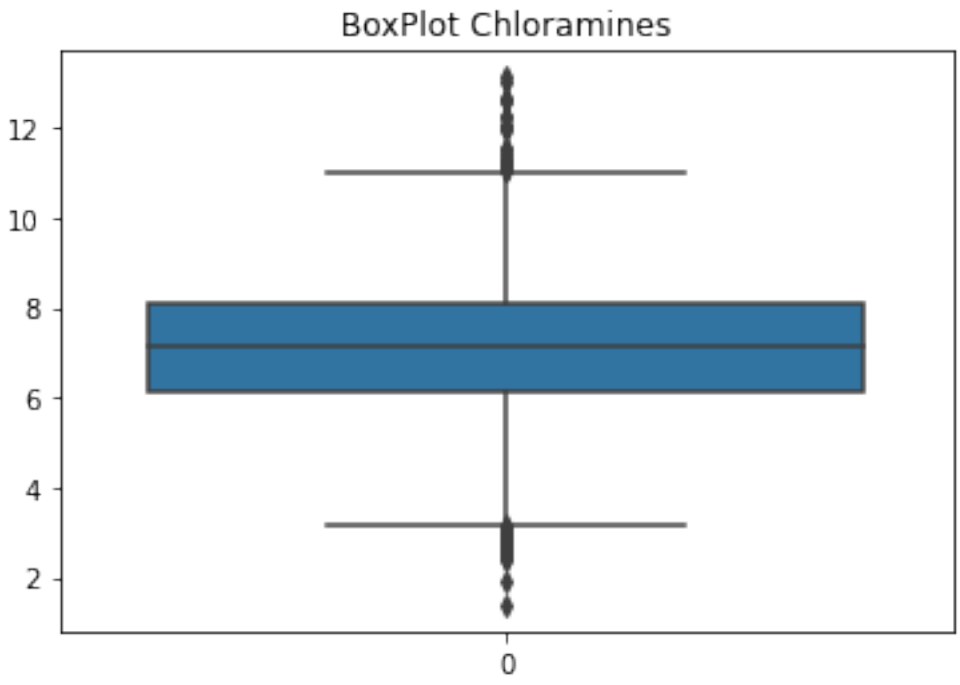
[ ]:
```
# Histogram kolom Chloramines

HistChloramines = sns.histplot(Chloramines, kde = True)
HistChloramines.set_title("Histogram Chloramines")
HistChloramines.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Chloramines

BoxPlotChloramines = sns.boxplot(data = Chloramines)
BoxPlotChloramines.set_title("BoxPlot Chloramines")
```

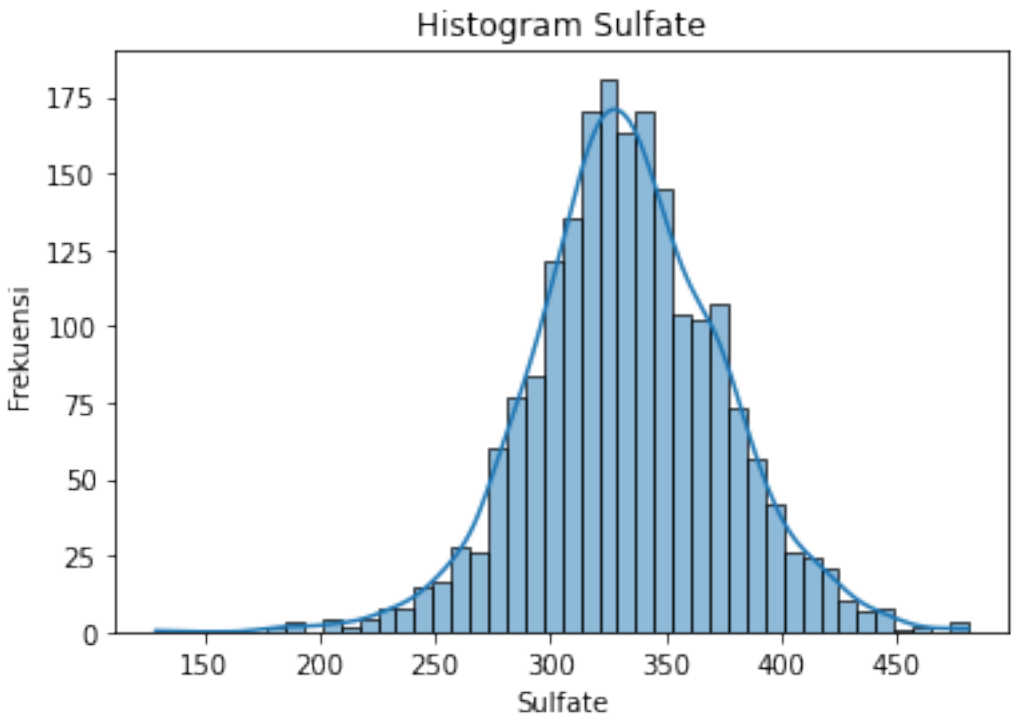
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Chloramines')
```



1.4.6 6. Sulfate

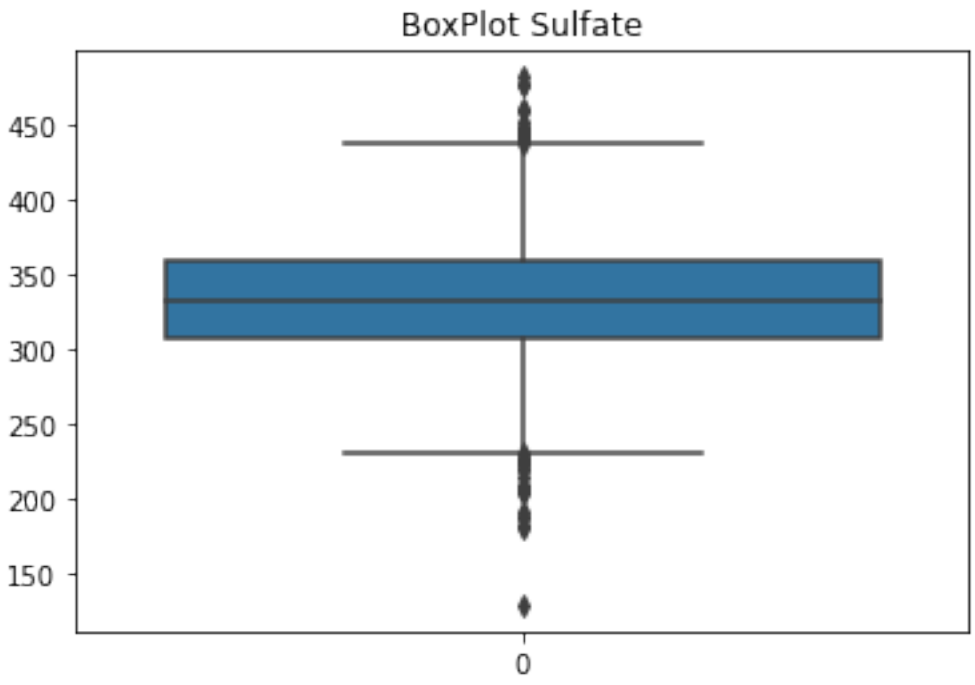
[ ]:
```
# Histogram kolom Sulfate

HistSulfate = sns.histplot(Sulfate, kde = True)
HistSulfate.set_title("Histogram Sulfate")
HistSulfate.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Sulfate

BoxPlotSulfate = sns.boxplot(data = Sulfate)
BoxPlotSulfate.set_title("BoxPlot Sulfate")
```

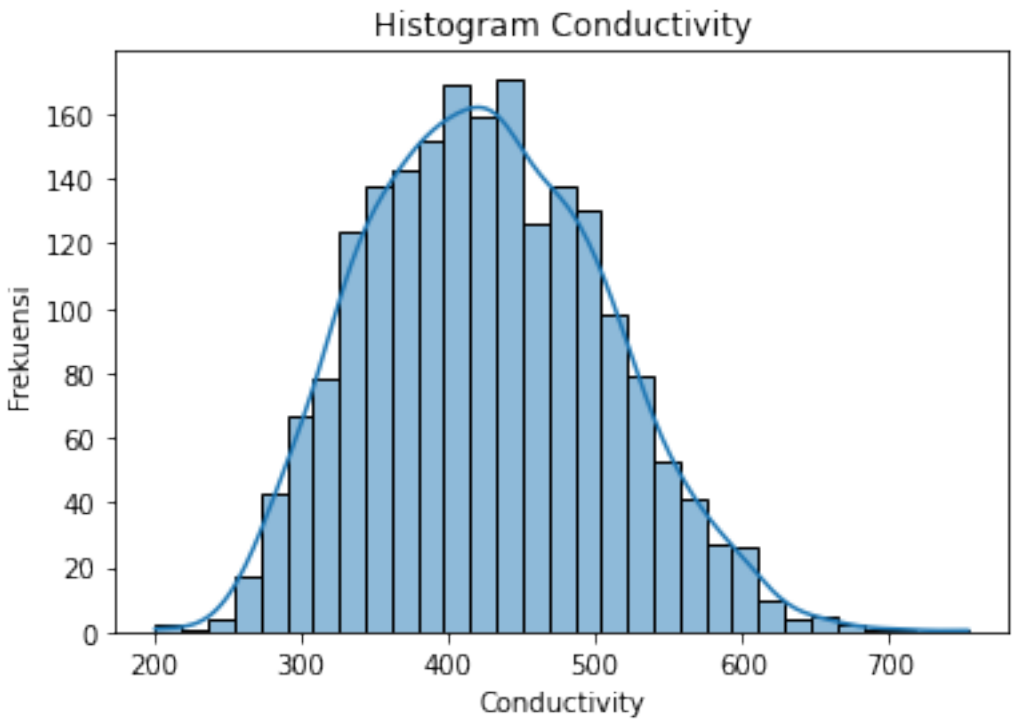
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Sulfate')
```



1.4.7 7. Conductivity

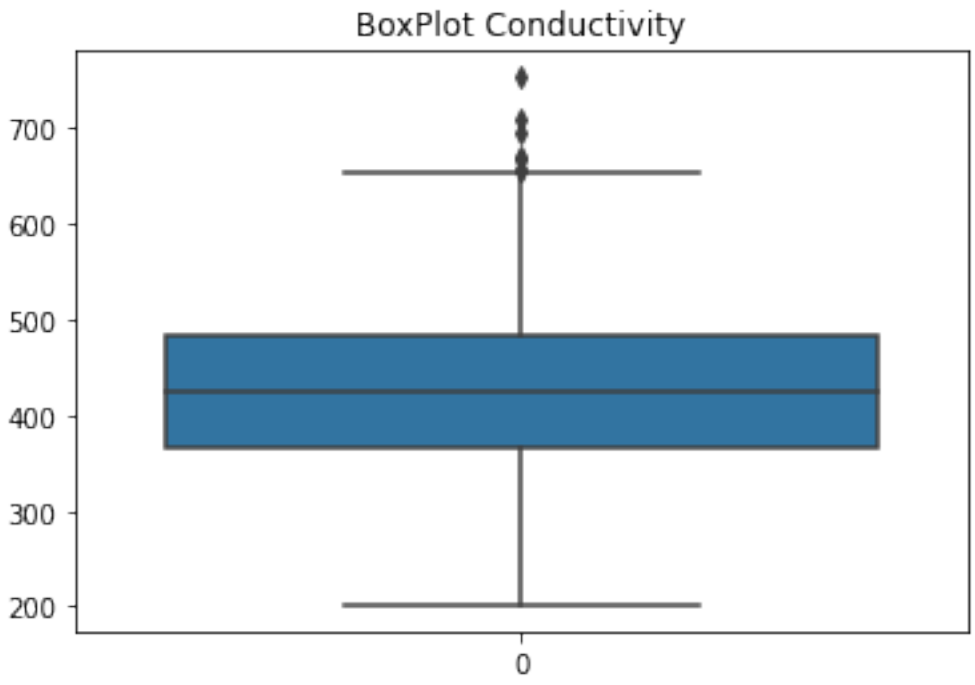
[ ]:
```
# Histogram kolom Conductivity

HistConductivity = sns.histplot(Conductivity, kde = True)
HistConductivity.set_title("Histogram Conductivity")
HistConductivity.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Conductivity

BoxPlotConductivity = sns.boxplot(data = Conductivity)
BoxPlotConductivity.set_title("BoxPlot Conductivity")
```

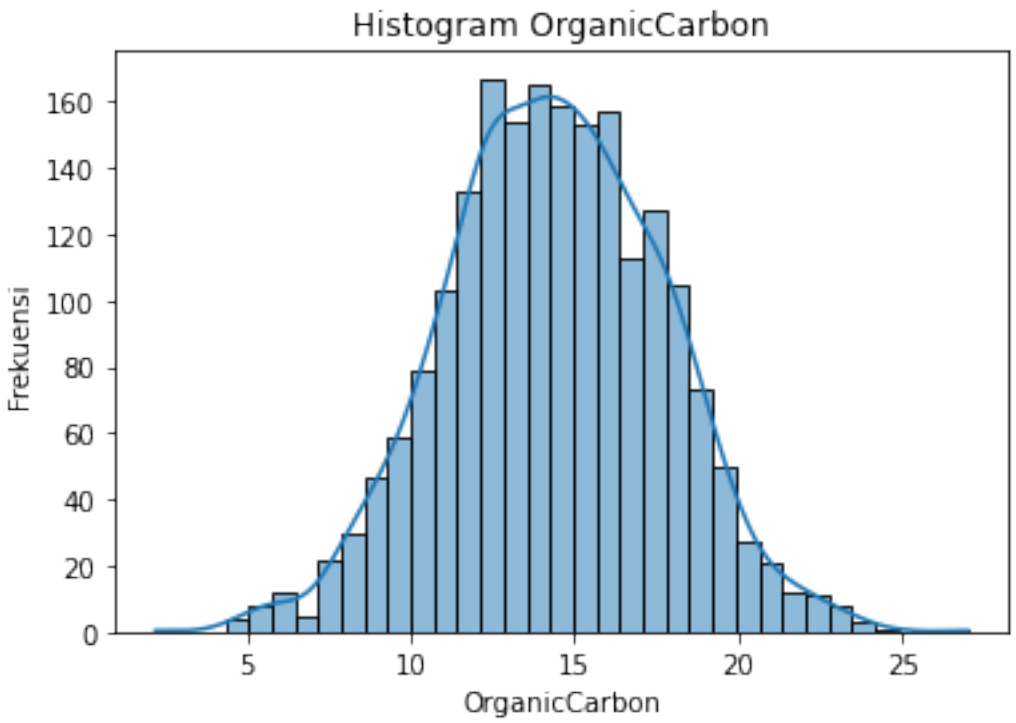
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Conductivity')
```



1.4.8 8. OrganicCarbon

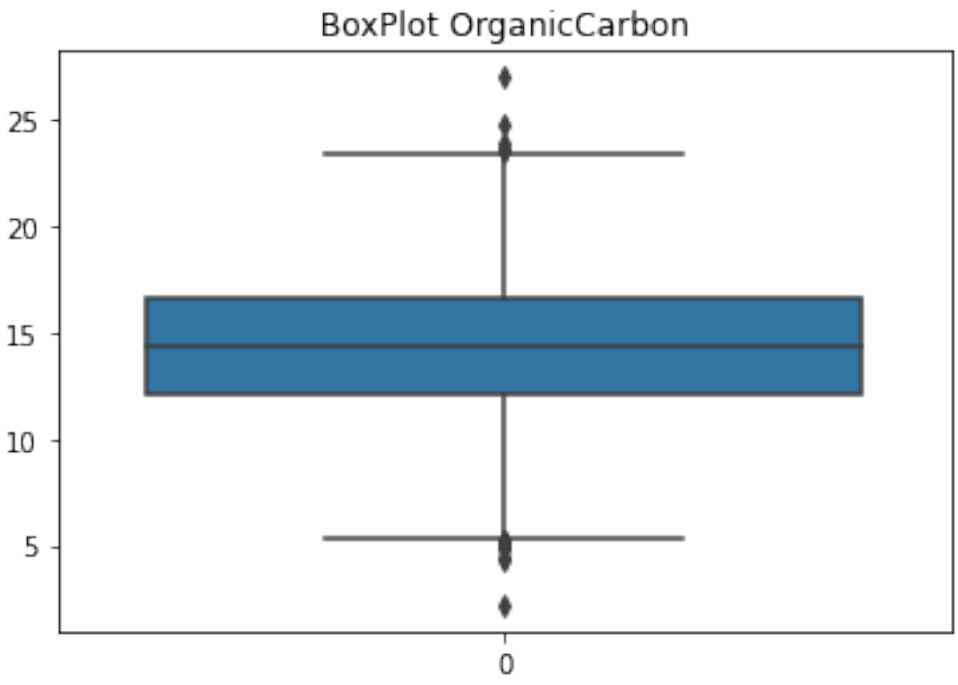
[ ]:
```
# Histogram kolom OrganicCarbon

HistOrganicCarbon = sns.histplot(OrganicCarbon, kde = True)
HistOrganicCarbon.set_title("Histogram OrganicCarbon")
HistOrganicCarbon.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom OrganicCarbon

BoxPlotOrganicCarbon = sns.boxplot(data = OrganicCarbon)
BoxPlotOrganicCarbon.set_title("BoxPlot OrganicCarbon")
```

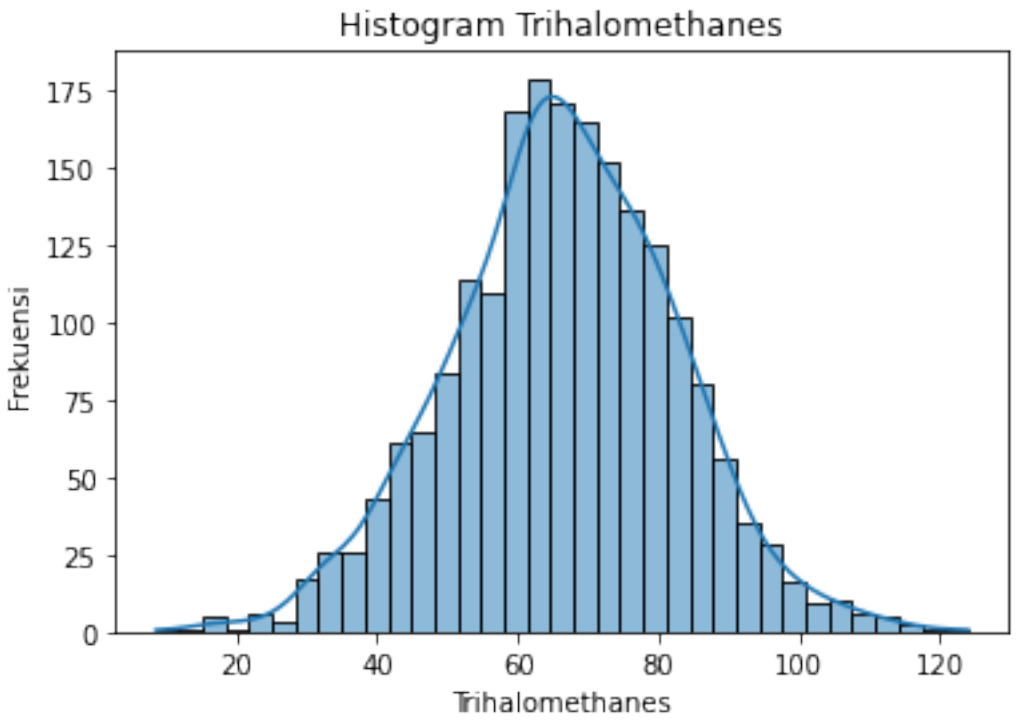
[ ]:
```
Text(0.5, 1.0, 'BoxPlot OrganicCarbon')
```



1.4.9 9. Trihalomethanes

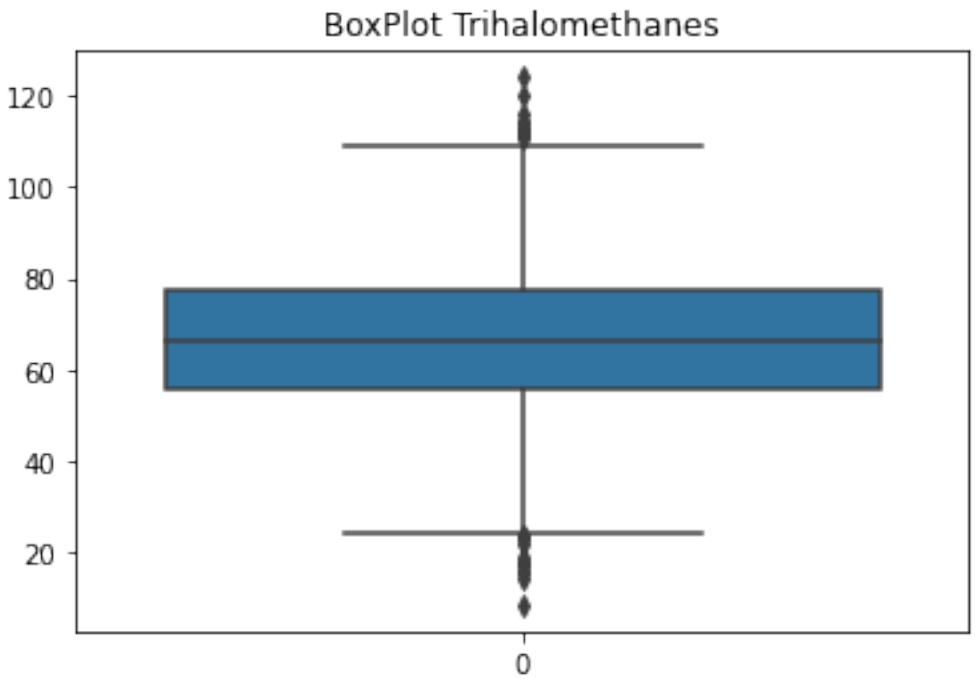
[ ]:
```
# Histogram kolom Trihalomethanes

HistTrihalomethanes = sns.histplot(Trihalomethanes, kde = True)
HistTrihalomethanes.set_title("Histogram Trihalomethanes")
HistTrihalomethanes.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Trihalomethanes

BoxPlotTrihalomethanes = sns.boxplot(data = Trihalomethanes)
BoxPlotTrihalomethanes.set_title("BoxPlot Trihalomethanes")
```

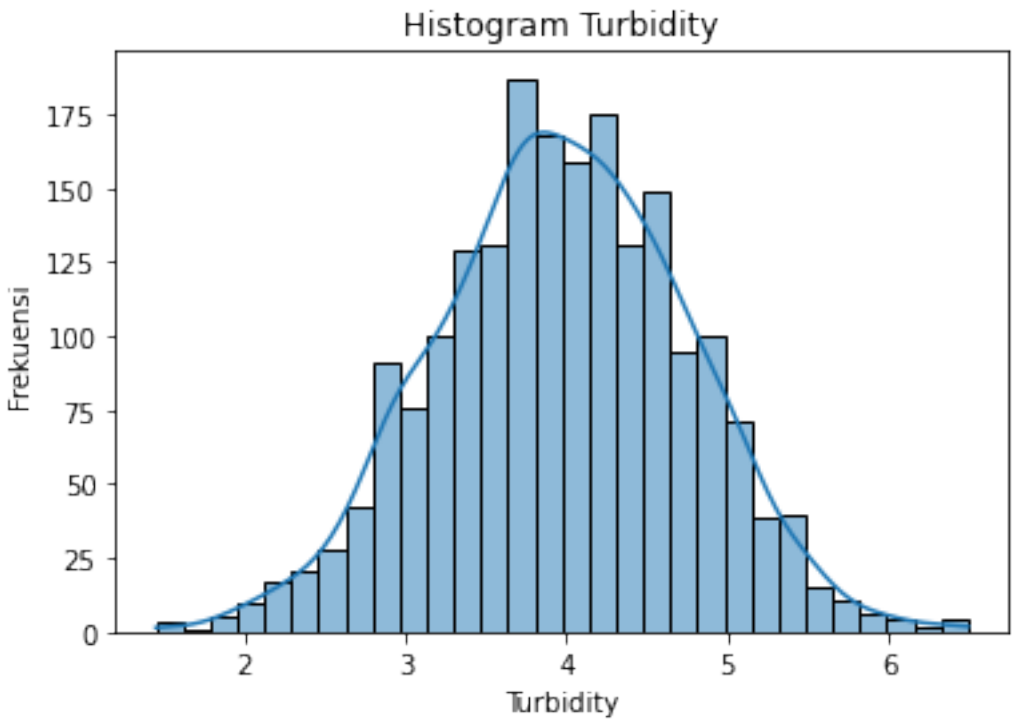
[ ]:
```
Text(0.5, 1.0, 'BoxPlot Trihalomethanes')
```



1.4.10 10. Turbidity

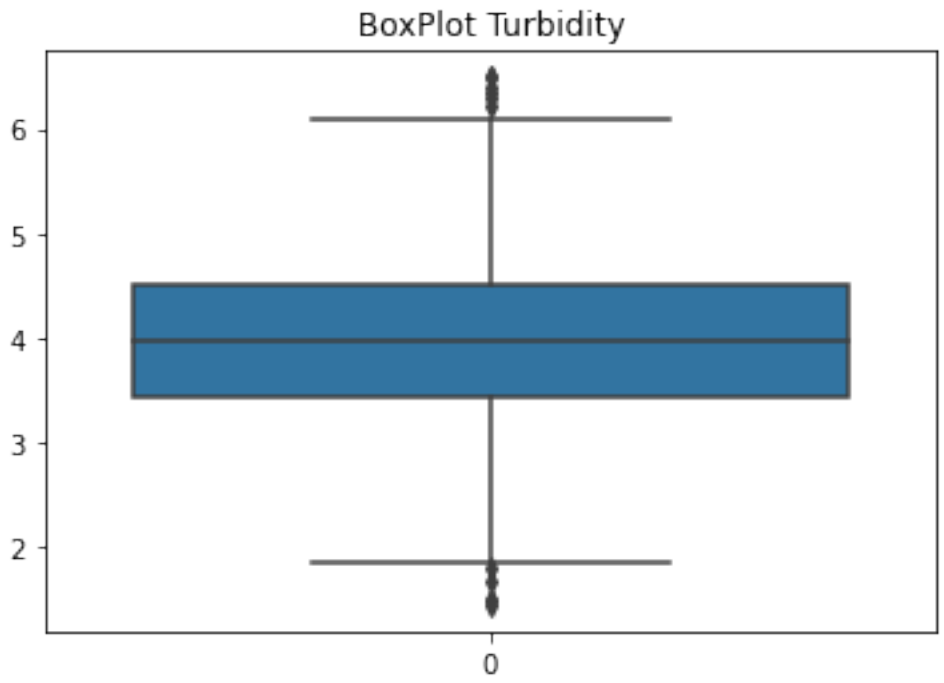
[ ]:
```
# Histogram kolom Turbidity

HistTurbidity = sns.histplot(Turbidity, kde = True)
HistTurbidity.set_title("Histogram Turbidity")
HistTurbidity.set_ylabel("Frekuensi")
```

[ ]:
```
Text(0, 0.5, 'Frekuensi')
```



[ ]:
```
# Boxplot kolom Turbidity

BoxPlotTurbidity = sns.boxplot(data = Turbidity)
BoxPlotTurbidity.set_title("BoxPlot Turbidity")
```

[ ]:
```
Text(0.5, 1.0, 'BoxPlot Turbidity')
```



1.4.11 11. Potability

Kolom potability tidak perlu divisualisasi karena merupakan kolom target.

1.5 Tahap 3 - Tes Normality

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

1.5.1 1. Id

Kolom id tidak perlu dilakukan normality test karena selalu unik untuk setiap data.

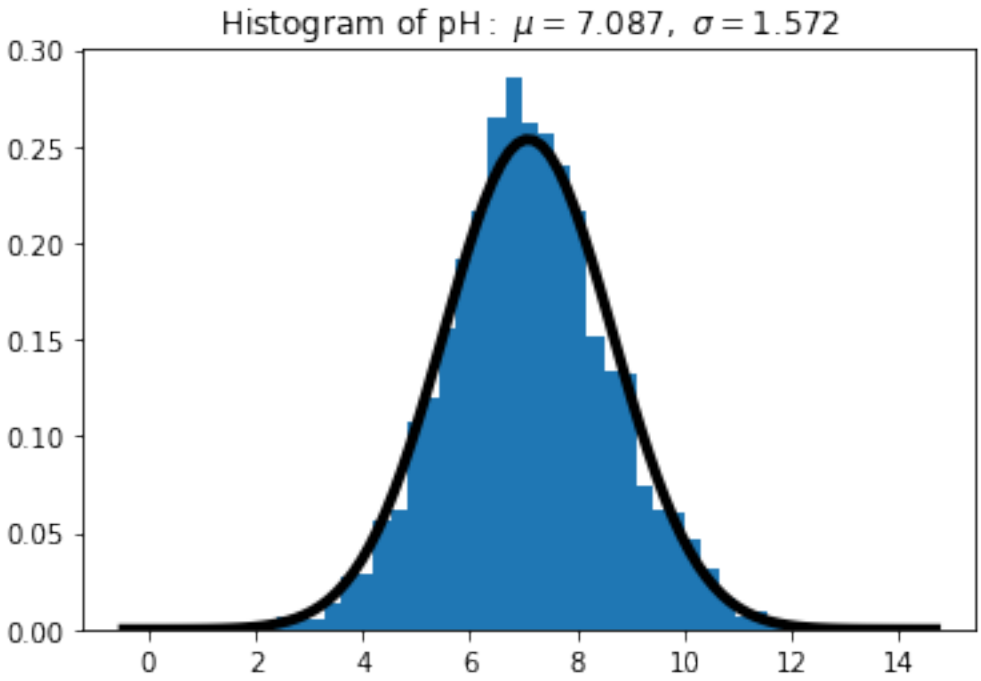
1.5.2 2. pH

[ ]:
```
k, p = st.normaltest(pH)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(pH)
plt.hist(pH, bins = 'auto', density = True)
```

```
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ pH:}\ \mu=%.3f,\ \sigma=%.3f$' %(mu, sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 2.65148e-05
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208effc69d0>]
```



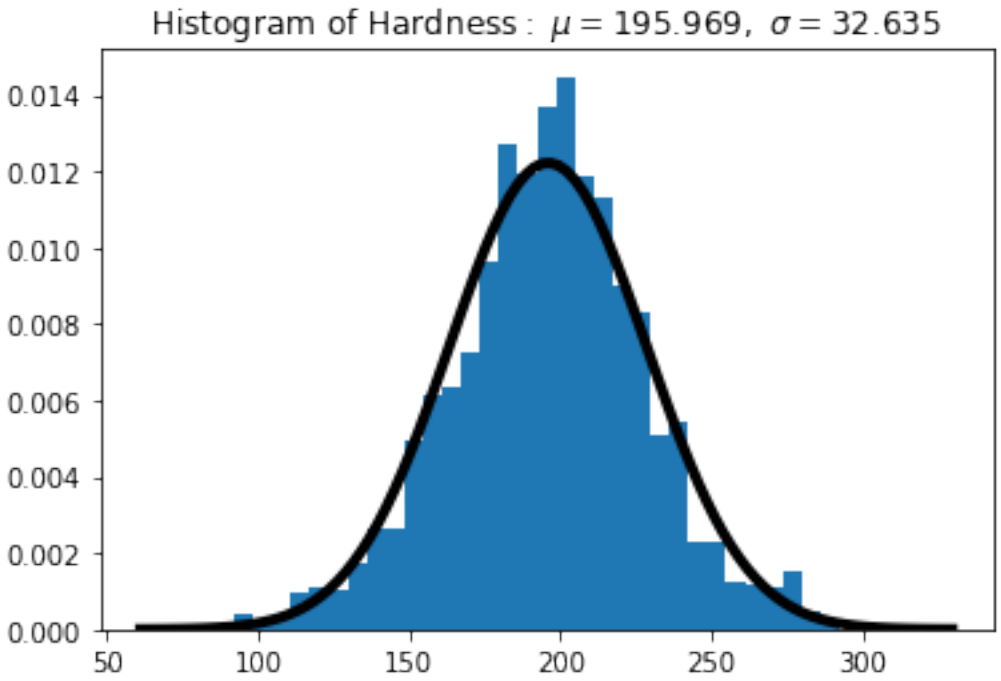
1.5.3 3. Hardness

[ ]:
```
k, p = st.normaltest(Hardness)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Hardness)
plt.hist(Hardness, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Hardness:}\ \mu=%.3f,\ \sigma=%.3f$' %(mu,␣
  ↪sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 0.000134424
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f01b5700>]
```



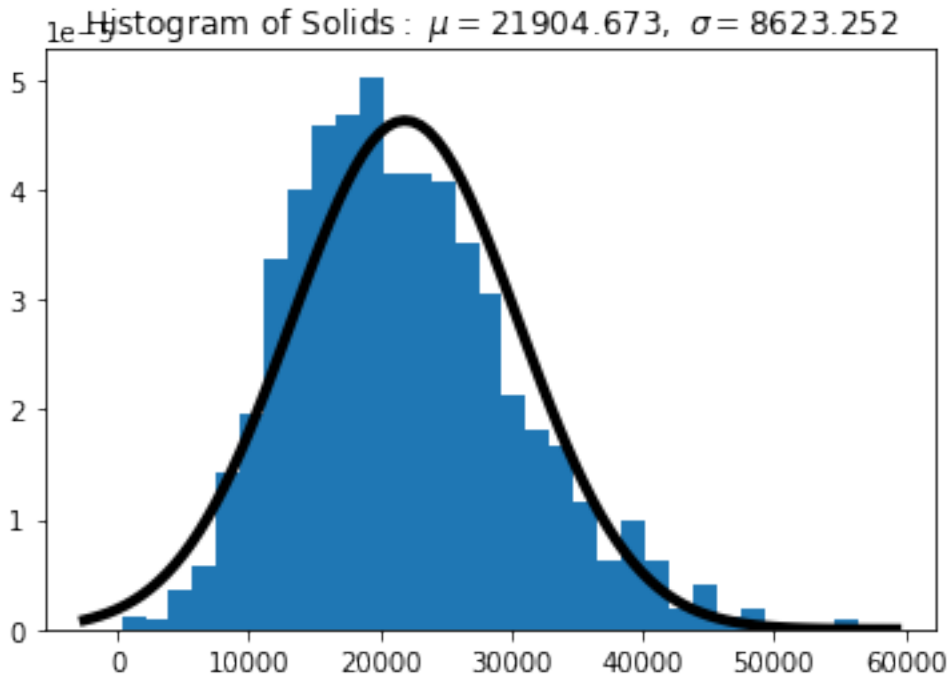
1.5.4 4. Solids

[ ]:
```
k, p = st.normaltest(Solids)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Solids)
plt.hist(Solids, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Solids:}\ \mu=%.3f,\ \sigma=%.3f$' %(mu,␣
  ↪sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 2.07966e-24
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f028feb0>]
```



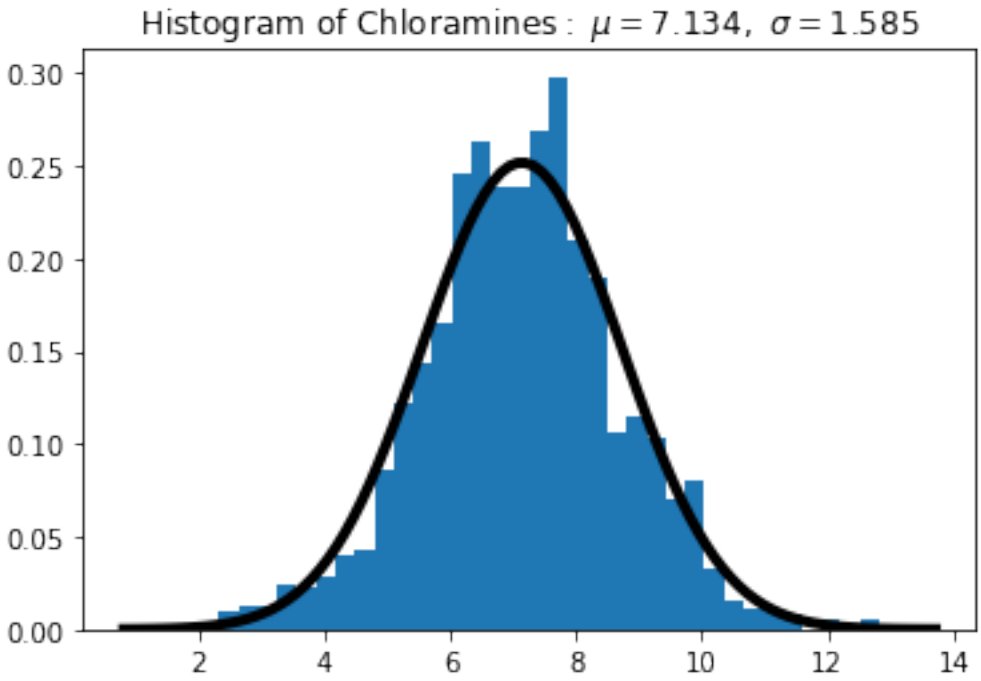
1.5.5 5. Chloramines

[ ]:
```
k, p = st.normaltest(Chloramines)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Chloramines)
plt.hist(Chloramines, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Chloramines:}\ \mu=%.3f,\ \sigma=%.3f$'␣
  ↪%(mu, sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 0.000250483
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f0392850>]
```



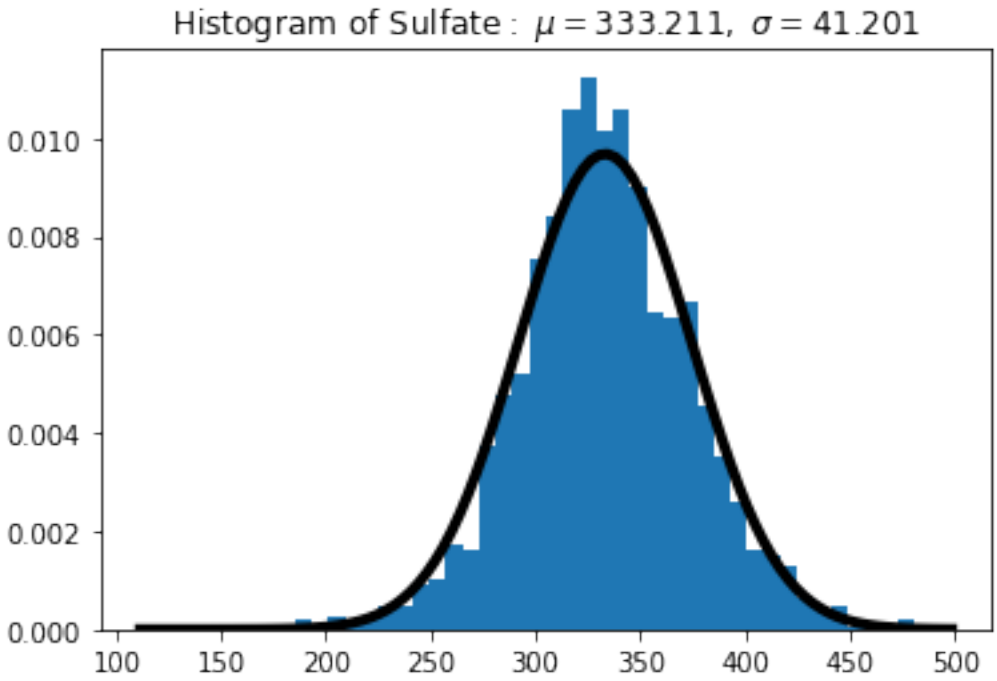
1.5.6 6. Sulfate

[ ]:
```
k, p = st.normaltest(Sulfate)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Sulfate)
plt.hist(Sulfate, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Sulfate:}\ \mu=%.3f,\ \sigma=%.3f$' %(mu,␣
  ↪sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 4.42559e-07
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f02614c0>]
```



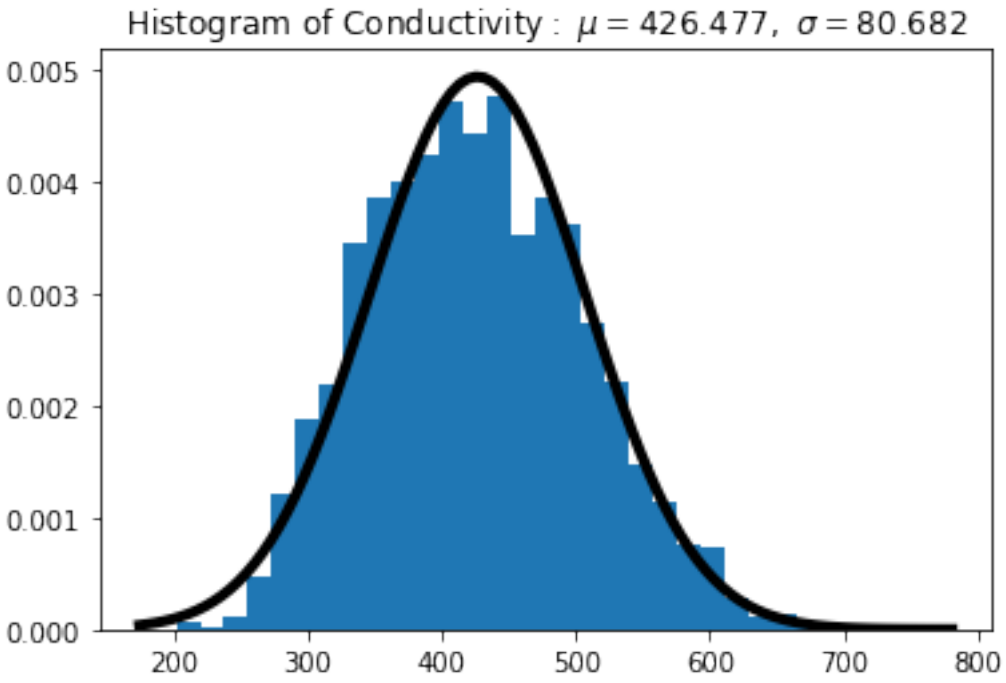
1.5.7 7. Conductivity

[ ]:
```
k, p = st.normaltest(Conductivity)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Conductivity)
plt.hist(Conductivity, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Conductivity:}\ \mu=%.3f,\ \sigma=%.3f$'␣
  ↪%(mu, sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 4.39018e-07
H0 ditolak, sehingga data tidak terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f04cfd00>]
```



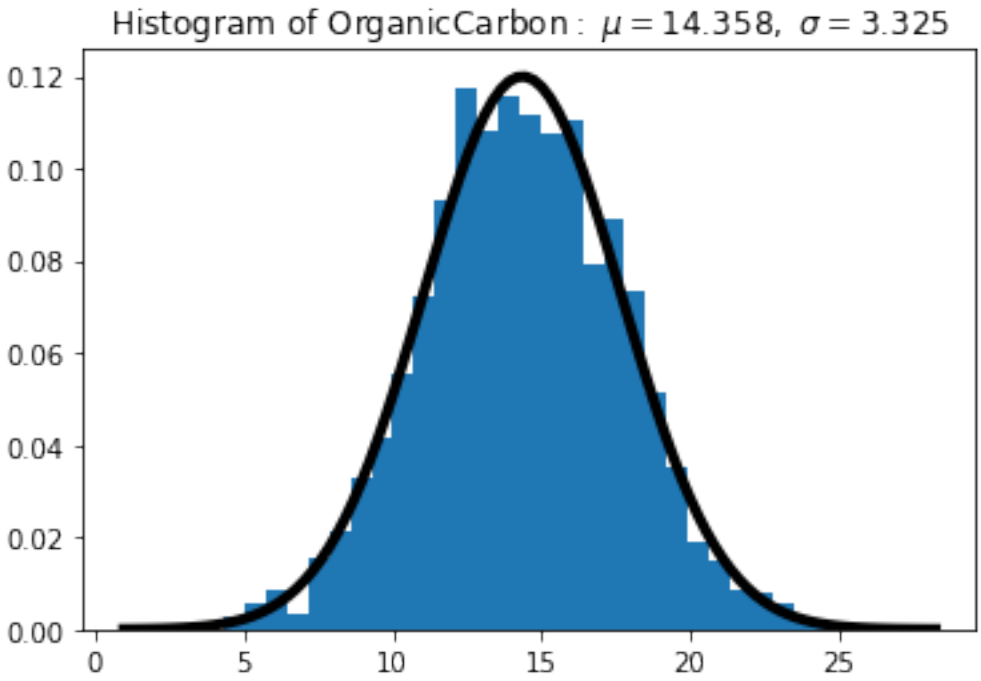
1.5.8 8. OrganicCarbon

[ ]:
```
k, p = st.normaltest(OrganicCarbon)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(OrganicCarbon)
plt.hist(OrganicCarbon, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ OrganicCarbon:}\ \mu=%.3f,\ \sigma=%.3f$'␣
  ↪%(mu, sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 0.88255
H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f05c3790>]
```



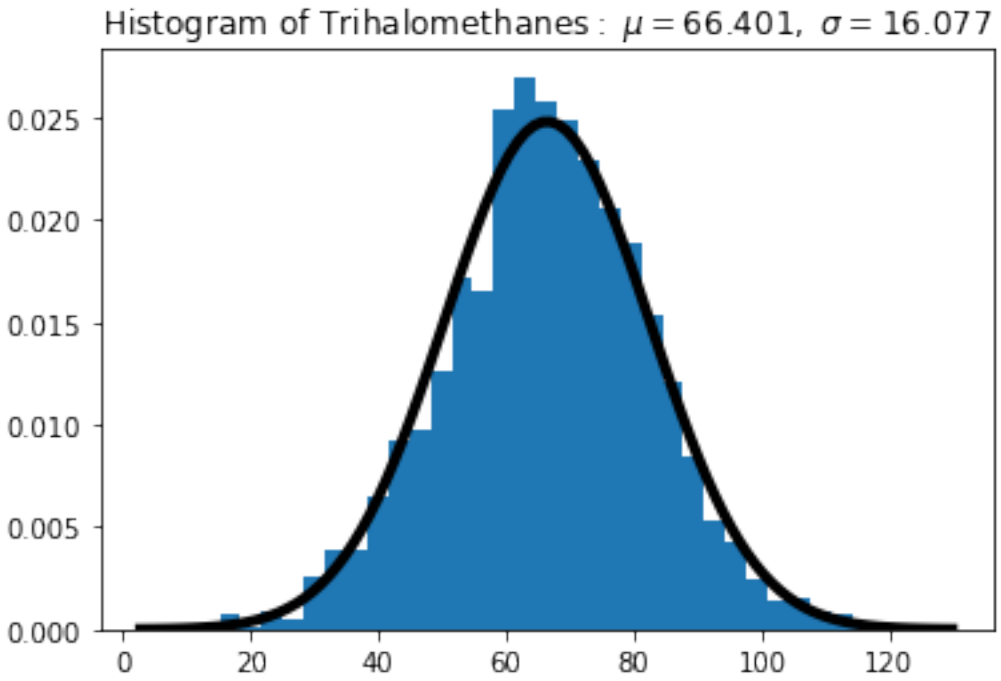
1.5.9 9. Trihalomethanes

[ ]:
```
k, p = st.normaltest(Trihalomethanes)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Trihalomethanes)
plt.hist(Trihalomethanes, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Trihalomethanes:}\ \mu=%.3f,\ \sigma=%.3f$'␣
  ↪%(mu, sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 0.10436
H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x208f06bd7f0>]
```



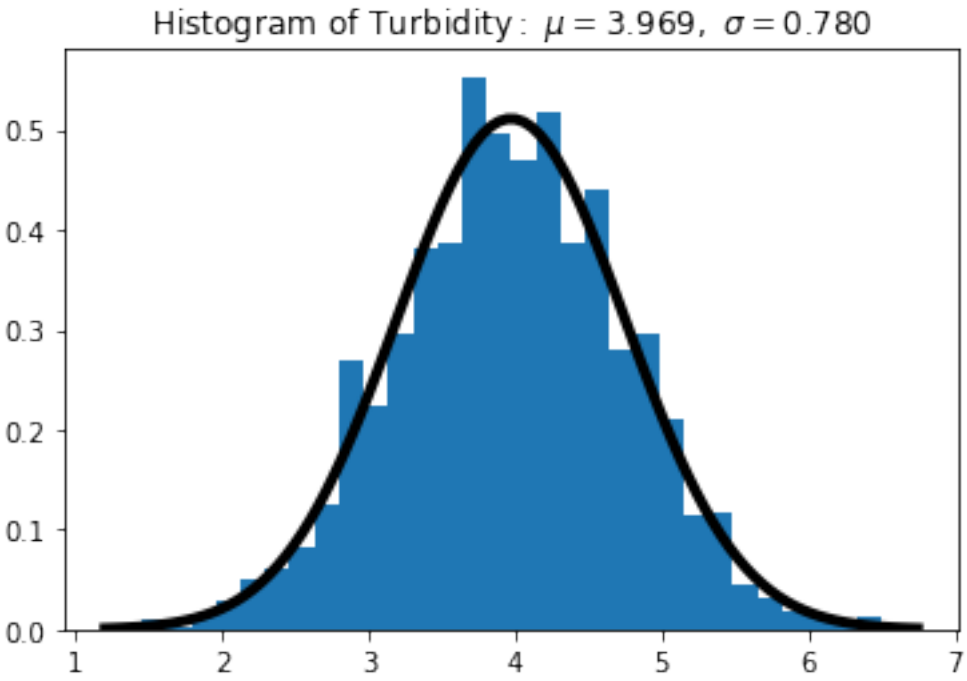
1.5.10 10. Turbidity

[ ]:
```
k, p = st.normaltest(Turbidity)
alpha = 0.05
print("p = {:g}".format(p))
if p < alpha:  # null hypothesis: x comes from a normal distribution
    print("H0 ditolak, sehingga data tidak terdistribusi dengan normal")
else:
    print("H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal")

mu, sigma = st.norm.fit(Turbidity)
plt.hist(Turbidity, bins = 'auto', density = True)
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
y = st.norm.pdf(x, mu, sigma)
plt.title(r'$\mathrm{Histogram\ of\ Turbidity:}\ \mu=%.3f,\ \sigma=%.3f$' %(mu,␣
  ↪sigma))
plt.plot(x, y,'k',lw=4)
```

```
p = 0.769472
H0 tidak bisa ditolak, sehingga data terdistribusi dengan normal
```

[ ]: [<matplotlib.lines.Line2D at 0x208f1783b20>]



1.5.11 11. Potability

Kolom potability tidak perlu dilakukan normality test karena merupakan kolom target.

1.6 Tahap 4 - Tes Hipotesis 1 Sampel

Melakukan test hipotesis 1 sampel dengan menuliskan 6 langkah testing dan menampilkan juga boxplotnya untuk kolom/bagian yang bersesuaian.

Enam Langkah Testing: 1. Tentukan Hipotesis nol (H0: = 0), dimana bisa berupa , 2, p, atau data lain berdistribusi tertentu (normal, binomial, dsc.). 2. Pilih hipotesis alternatif H1 salah dari dari > 0 , < 0 , atau 0. 3. Tentukan tingkat signifikan . 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 H0 jika nilai uji

terletak di daerah kritis atau dengan tes signifikan, TOLAK H0 jika p-value lebih kecil dibanding tingkat signifikansi yang diinginkan.

[ ]:
```
def Z_testStatistic(x̄, 0, ,root_n):
    return (float)(x̄- 0)/( /root_n)
def Z_testStatistic_bigN(p̂,p0,q0,n):
    return (float)(p̂-p0)/np.sqrt(p0*q0/n)
def T_testStatistic(x̄, 0,s,root_n):
    return (float)(x̄- 0)/(s/root_n)
```

1.6.1 a. Nilai rata-rata pH di atas 7?

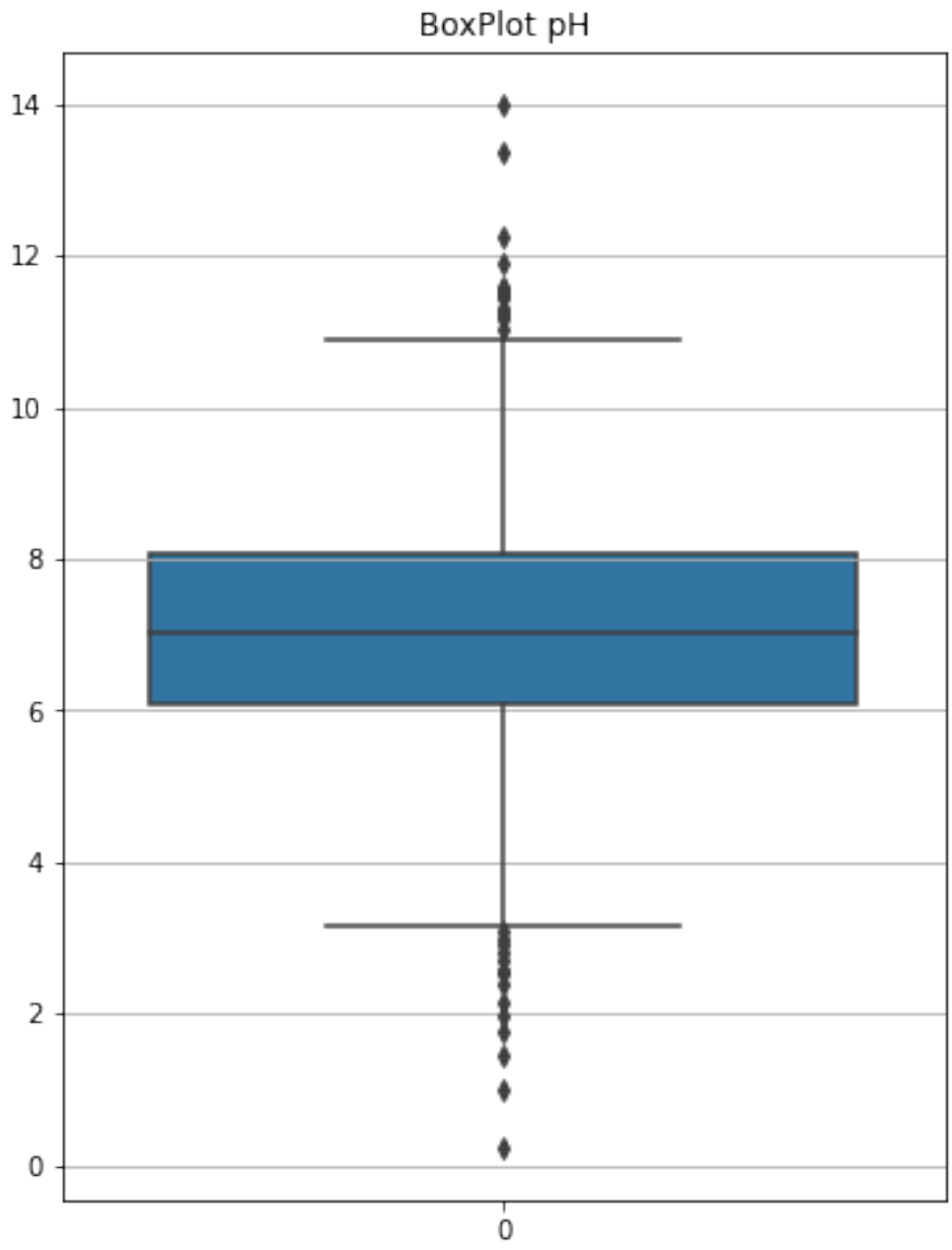
Asumsi sampel yang dicek sejumlah 2010 di mana 2010 data tersebut juga merupakan populasi (sampel yang dicek sekaligus populasi). Oleh karena itu, digunakan uji statistik distribusi Z untuk satu mean dengan standar deviasi populasi diketahui.

[ ]:
```
print("Nilai rata-rata pH di atas 7?")

# Langkah 1
H0 = " =7"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " >7"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z  = round(scipy.stats.norm.ppf(1- ),3)
print("4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui")
print("   Daerah Kritis: z>z  : z>{}".format(z ))

# Langkah 5
x̄ = pH.mean()
 0= 7
  = pH.std()
root_n = np.sqrt(len(pH))
z = round(Z_testStatistic(x̄, 0, ,root_n),3)
p_value = 1-scipy.stats.norm.cdf(z)
print("5. Komputasi")
print("   x̄: {} \n   root_n: {} \n    : {} \n    0: {}".format(x̄,root_n, , 0))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
```

```
if (z > z ):
    print("   Tolak H0 karena nilai uji = {}>{} (z>z )".format(str(z),str(z )))
    print("   Rata-Rata pH di atas 7")
else:
    print("   Terima H0 karena nilai uji = {}<={} (z<=z )".
 ↪format(str(z),str(z )))
    print("   Rata-Rata pH sama dengan 7")

# Menggambar Boxplot pH
# Konfigurasi Boxplot
plt.figure(figsize=(6,8))
plt.grid()
# Penggambaran Boxplot
BoxPlotpH= sns.boxplot(data=pH)
BoxPlotpH.set_title("BoxPlot pH")
```

```
Nilai rata-rata pH di atas 7?
1. H0:  =7
2. H1:  >7
3.   = 0.05
4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui
   Daerah Kritis: z>z  : z>1.645
5. Komputasi
   x̄: 7.0871927687138285
   root_n: 44.83302354291979
    : 1.5728029470456655
    0: 7
   p_value: 0.006477571731867804
   z: 2.485
6. Test Daerah Kritis
   Tolak H0 karena nilai uji = 2.485>1.645 (z>z )
   Rata-Rata pH di atas 7
```

[ ]: Text(0.5, 1.0, 'BoxPlot pH')



1.6.2 b. Nilai rata-rata Hardness tidak sama dengan 205?

Asumsi sampel yang dicek sejumlah 2010 di mana 2010 data tersebut juga merupakan populasi (sampel yang dicek sekaligus populasi). Oleh karena itu, digunakan uji statistik distribusi Z untuk satu mean dengan standar deviasi populasi diketahui.

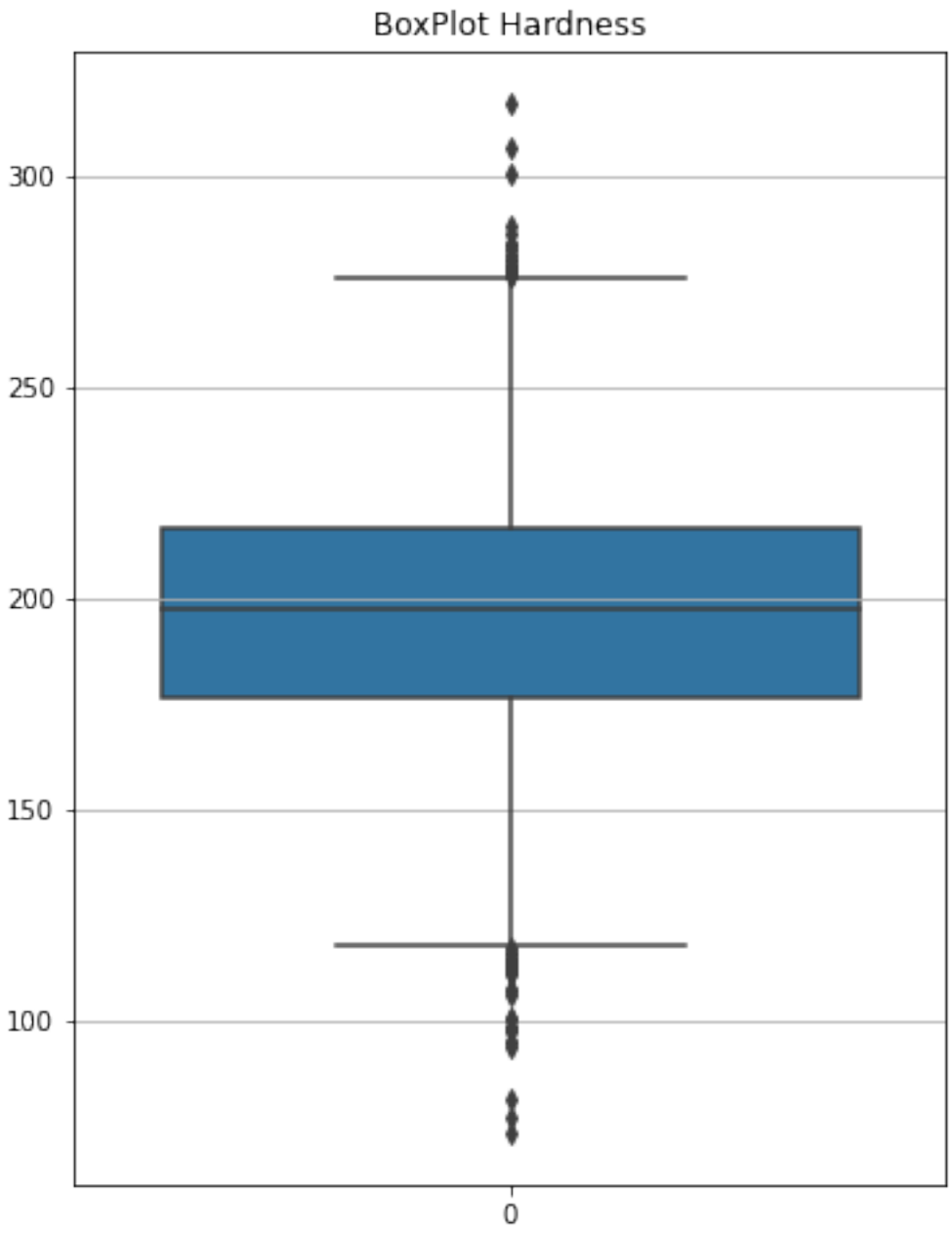
[ ]:
```
print("Nilai rata-rata Hardness tidak sama dengan 205?")

# Langkah 1
```

```
H0 = " =205"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " 205"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z _div2 = round(scipy.stats.norm.ppf(1-( /2)),3)
print("4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui")
print("   Daerah Kritis: z>z /2 atau z<-z /2 : z>{} atau z<{}".
 ↪format(z _div2,-1*z _div2))

# Langkah 5
x̄ = Hardness.mean()
 0= 205
  = Hardness.std()
n = len(Hardness)
root_n = np.sqrt(n)
z = round(Z_testStatistic(x̄, 0, ,root_n),3)
p_value = 1-abs(scipy.stats.norm.cdf(z)-scipy.stats.norm.cdf(-1*z))
print("5. Komputasi")
print("   x̄: {} \n   n: {} \n   root_n: {} \n    : {} \n    0: {}".
 ↪format(x̄,n,root_n, , 0))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z _div2 or z < -1*z _div2):
    if (z > z _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (z>z /2)".format(z,z _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (z<-z /2)".
 ↪format(z,-1*z _div2))
    print("   Rata-rata Hardness tidak sama dengan 205")
else:
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z /2<z<z /2)".
 ↪format(-1*z _div2,z,z _div2))
    print("   Rata-rata Hardness sama dengan 205")

# Menggambar Boxplot Hardness
# Konfigurasi Boxplot
plt.figure(figsize=(6,8))
```

```
plt.grid()
# Penggambaran Boxplot
BoxPlotHardness= sns.boxplot(data=Hardness)
BoxPlotHardness.set_title("BoxPlot Hardness")
```

```
Nilai rata-rata Hardness tidak sama dengan 205?
1. H0:  =205
2. H1:   205
3.    = 0.05
4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui
   Daerah Kritis: z>z /2 atau z<-z /2 : z>1.96 atau z<-1.96
5. Komputasi
   x̄: 195.96920903783524
   n: 2010
   root_n: 44.83302354291979
    : 32.643165859429864
    0: 205
   p_value: 0.0
   z: -12.403
6. Test Daerah Kritis
   Tolak H0 karena nilai uji = -12.403<-1.96 (z<-z /2)
   Rata-rata Hardness tidak sama dengan 205
```

[ ]: Text(0.5, 1.0, 'BoxPlot Hardness')



1.6.3 c. Nilai rata-rata 100 baris pertama kolom Solids bukan 21900?

Asumsi sampel yang dicek sejumlah 100, yakni 100 baris pertama dari populasi yang berjumlah 2010 baris. Oleh karena itu, digunakan uji statistik distribusi t untuk satu mean dengan standar deviasi populasi diketahui.

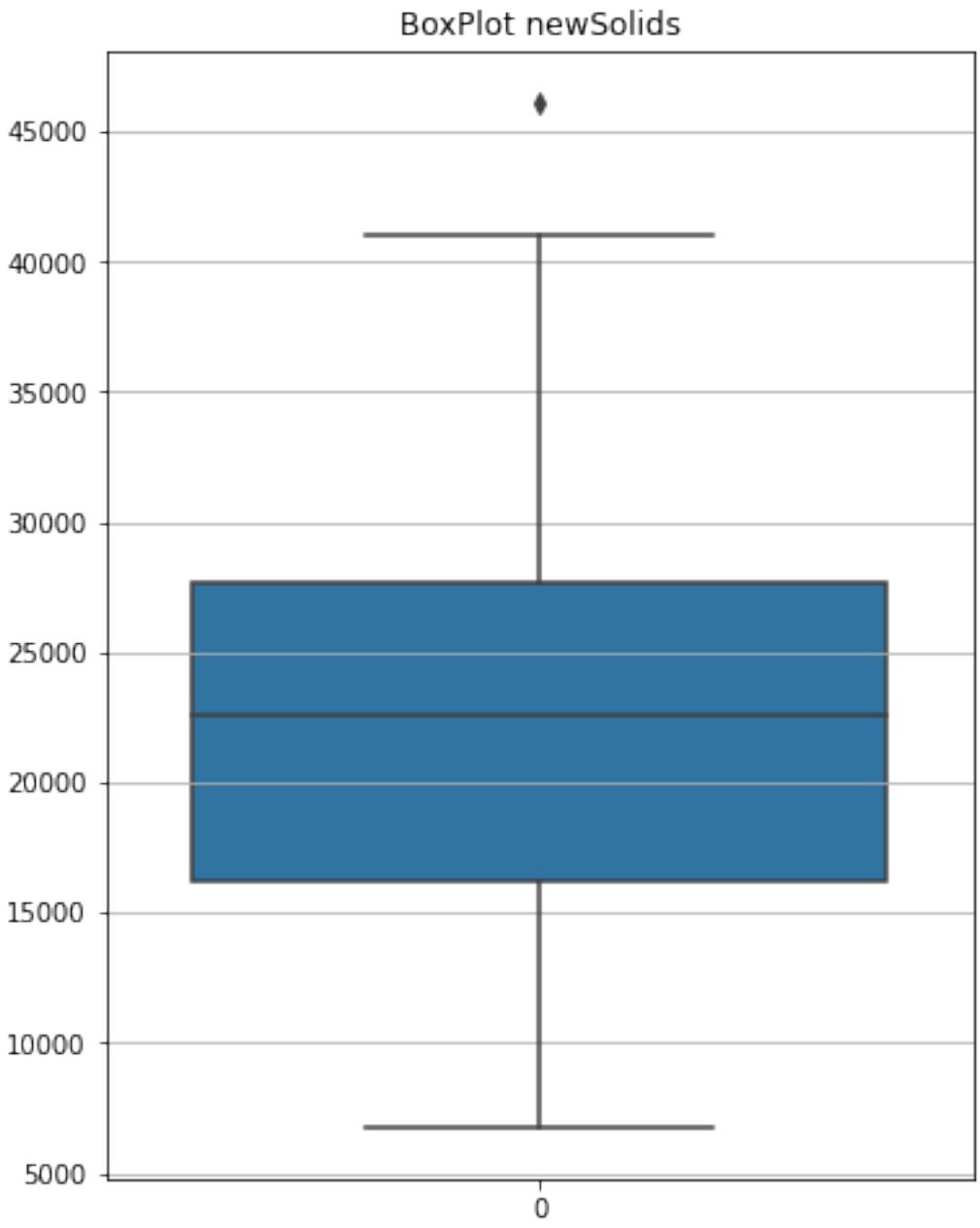
[ ]:
```
print("Nilai rata-rata 100 baris pertama kolom Solids bukan 21900?")

# Langkah 1
```

```
H0 = " =21900"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " 21900"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z _div2 = round(scipy.stats.norm.ppf(1-( /2)),3)
print("4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui")
print("   Daerah Kritis: z>z /2 atau z<-z /2 : z>{} atau z<{}".
 ↪format(z _div2,z _div2))

# Langkah 5
newSolids = Solids[:100]
x̄ = newSolids.mean()
 0= 21900
  = Solids.std()
n = len(newSolids)
root_n = np.sqrt(n)
z = round(Z_testStatistic(x̄, 0, ,root_n),3)
p_value = 1-abs(scipy.stats.norm.cdf(z)-scipy.stats.norm.cdf(-1*z))
print("5. Komputasi")
print("   x̄: {} \n   root_n: {} \n    : {} \n    0: {}".format(x̄,root_n, , 0))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z _div2 or z < -1*z _div2):
    if (z > z _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (z>z /2)".format(z,z _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (z<-z /2)".
 ↪format(z,-1*z _div2))
    print("   Nilai Rata-rata 100 baris pertama kolom Solids tidak sama dengan␣
 ↪21900")
else:
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z /2<z<z /2)".
 ↪format(-1*z _div2,z,z _div2))
    print("   Nilai Rata-rata 100 baris pertama kolom Solids sama dengan 21900")

# Menggambar Boxplot Solids
# Konfigurasi Boxplot
```

```
plt.figure(figsize=(6,8))
plt.grid()
# Penggambaran Boxplot
BoxPlotSolids= sns.boxplot(data=newSolids)
BoxPlotSolids.set_title("BoxPlot newSolids")
```

```
Nilai rata-rata 100 baris pertama kolom Solids bukan 21900?
1. H0:  =21900
2. H1:   21900
3.   = 0.05
4. Uji Statistik: z=(x̄- 0)/( /root_n),   diketahui
   Daerah Kritis: z>z /2 atau z<-z /2 : z>1.96 atau z<1.96
5. Komputasi
   x̄: 22347.334446383426
   root_n: 10.0
    : 8625.397911190576
    0: 21900
   p_value: 0.6037607412507624
   z: 0.519
6. Test Daerah Kritis
   Terima H0 karena nilai uji = -1.96<0.519<1.96 (-z /2<z<z /2)
   Nilai Rata-rata 100 baris pertama kolom Solids sama dengan 21900
```

[ ]: Text(0.5, 1.0, 'BoxPlot newSolids')



1.6.4 d. Proporsi nilai Conductivity yang lebih dari 450 adalah tidak sama dengan 10%?

Digunakan uji statistik berupa uji proporsi dengan n besar (binomial mendekati normal).

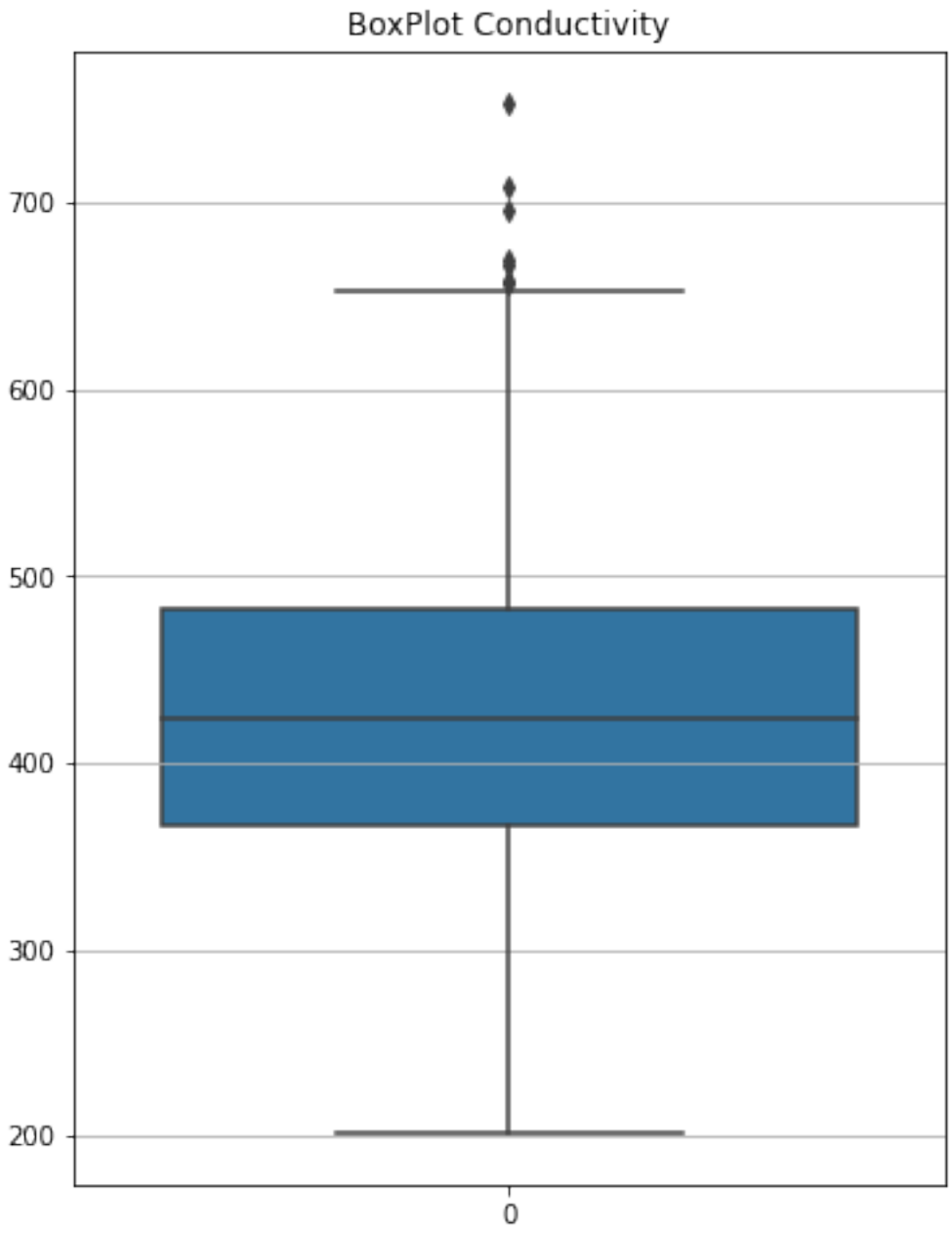
[ ]:
```
print("Proporsi nilai Conductivity yang lebih dari 450 adalah tidak sama dengan␣
↪10%?")
```

```
# Langkah 1
H0 = "p=0.10"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = "p 0.10"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4

z _div2 = round(scipy.stats.norm.ppf(1-( /2)),3)
print("4. Uji Statistik: z=(p̂-p0)/sqrt(p0*q0/n),   diketahui")
print("   Daerah Kritis: z>z /2 atau z<-z /2 : z>{} atau z<{}".
 ↪format(z _div2,z _div2))

# Langkah 5
newConductivity = [dia for dia in Conductivity if dia > 450]
p̂ = len(newConductivity)/len(Conductivity)
p0 = 0.10
q0 = 1-p0
n = len(Conductivity)
z = round(Z_testStatistic_bigN(p̂,p0,q0,n),3)
p_value = 1-abs(scipy.stats.norm.cdf(z)-scipy.stats.norm.cdf(-1*z))
print("5. Komputasi")
print("   p̂: {} \n   p0: {} \n   q0: {} \n   n: {}".format(p̂,p0,q0,n))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z _div2 or z < -1*z _div2):
    if (z > z _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (z>z /2)".format(z,z _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (z<-z /2)".
 ↪format(z,-1*z _div2))
    print("   Proporsi nilai Conductivity yang lebih dari 450 tidak sama dengan␣
 ↪10%")
else:
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z /2<z<z /2)".
 ↪format(-1*z _div2,z,z _div2))
    print("   Proporsi nilai Conductivity yang lebih dari 450 sama dengan 10%")

# Menggambar Boxplot Conductivity
```

```
# Konfigurasi Boxplot
plt.figure(figsize=(6,8))
plt.grid()
# Penggambaran Boxplot
BoxPlotConductivity= sns.boxplot(data=Conductivity)
BoxPlotConductivity.set_title("BoxPlot Conductivity")
```

```
Proporsi nilai Conductivity yang lebih dari 450 adalah tidak sama dengan 10%?
1. H0: p=0.10
2. H1: p 0.10
3.   = 0.05
4. Uji Statistik: z=(p̂-p0)/sqrt(p0*q0/n),   diketahui
   Daerah Kritis: z>z /2 atau z<-z /2 : z>1.96 atau z<1.96
5. Komputasi
   p̂: 0.3706467661691542
   p0: 0.1
   q0: 0.9
   n: 2010
   p_value: 0.0
   z: 40.446
6. Test Daerah Kritis
   Tolak H0 karena nilai uji = 40.446>1.96 (z>z /2)
   Proporsi nilai Conductivity yang lebih dari 450 tidak sama dengan 10%
```

[ ]: Text(0.5, 1.0, 'BoxPlot Conductivity')



1.6.5 e. Proporsi nilai Trihalomethanes yang kurang dari 40 adalah kurang dari 5%?

Digunakan uji statistik berupa uji proporsi dengan n besar (binomial mendekati normal).

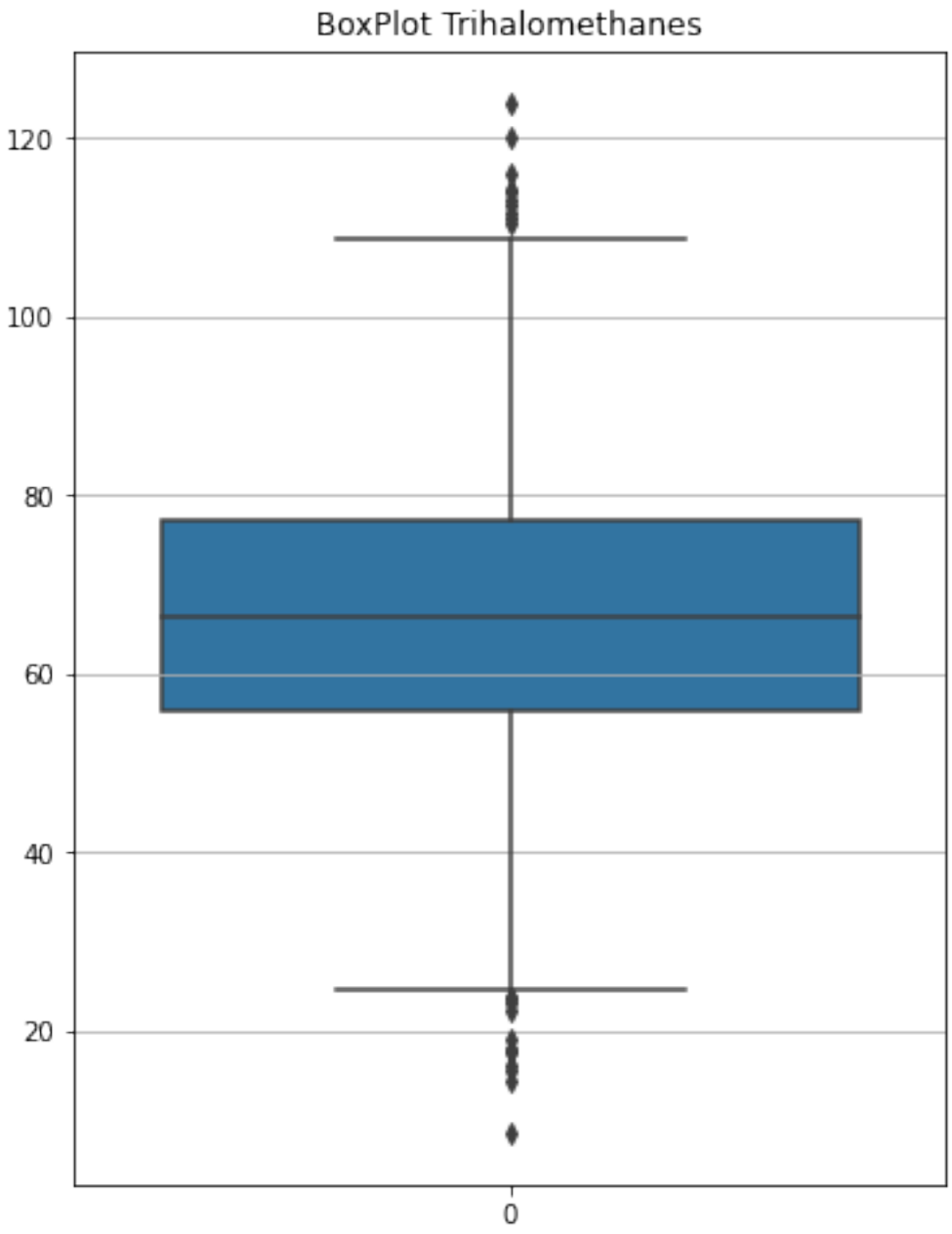
```
[ ]: print("Proporsi nilai Trihalomethanes yang kurang dari 40 adalah kurang dari 5%?
     ↪")

     # Langkah 1
```

```
H0 = "p=0.05"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = "p<0.05"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z  = round(scipy.stats.norm.ppf(1- ),3)
print("4. Uji Statistik: z=(p̂-p0)/sqrt(p0*q0/n),   diketahui")
print("   Daerah Kritis: z<-z  : z<{}".format(-1*z ))

# Langkah 5
newKel = [kel for kel in Trihalomethanes if kel < 40]
p̂ = len(newKel)/len(Trihalomethanes)
p0 = 0.05
q0 = 1-p0
n = len(Trihalomethanes)
z = round(Z_testStatistic_bigN(p̂,p0,q0,n),3)
p_value = scipy.stats.norm.cdf(z)
print("5. Komputasi")
print("   p̂: {} \n   p0: {} \n   q0: {} \n   n: {}".format(p̂,p0,q0,n))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z < -1*z ):
    print("   Tolak H0 karena nilai uji = {}<{} (z<-z )".
 ↪format(str(z),str(-1*z )))
    print("   Proporsi nilai Trihalomethanes yang kurang dari 40 adalah kurang␣
 ↪dari 5%")
else:
    print("   Terima H0 karena nilai uji = {}>={} (z>=-z )".
 ↪format(str(z),str(-z )))
    print("   Proporsi nilai Trihalomethanes yang kurang dari 40 adalah sama␣
 ↪dengan 5%")

# Menggambar Boxplot Trihalomethanes
# Konfigurasi Boxplot
plt.figure(figsize=(6,8))
plt.grid()
# Penggambaran Boxplot
BoxPlotTrihalomethanes= sns.boxplot(data=Trihalomethanes)
```

```
BoxPlotTrihalomethanes.set_title("BoxPlot Trihalomethanes")
```

```
Proporsi nilai Trihalomethanes yang kurang dari 40 adalah kurang dari 5%?
1. H0: p=0.05
2. H1: p<0.05
3.   = 0.05
4. Uji Statistik: z=(p̂-p0)/sqrt(p0*q0/n),   diketahui
   Daerah Kritis: z<-z  : z<-1.645
5. Komputasi
   p̂: 0.0527363184079602
   p0: 0.05
   q0: 0.95
   n: 2010
   p_value: 0.7132825580297869
   z: 0.563
6. Test Daerah Kritis
   Terima H0 karena nilai uji = 0.563>=-1.645 (z>=-z )
   Proporsi nilai Trihalomethanes yang kurang dari 40 adalah sama dengan 5%
```

[ ]: Text(0.5, 1.0, 'BoxPlot Trihalomethanes')



1.7 Tahap 5 - Tes Hipotesis 2 Sampel

Melakukan test hipotesis 2 sampel dengan menuliskan 6 langkah testing dan menampilkan juga boxplotnya untuk kolom/bagian yang bersesuaian.

Enam Langkah Testing: 1. Tentukan Hipotesis nol (H0: = 0), dimana bisa berupa , 2, p, atau data lain berdistribusi tertentu (normal, binomial, dsc.). 2. Pilih hipotesis alternatif H1 salah dari dari > 0 , < 0 , atau 0. 3. Tentukan tingkat signifikan . 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 H0 jika nilai uji terletak di daerah kritis atau dengan tes signifikan, TOLAK H0 jika p-value lebih kecil dibanding tingkat signifikansi yang diinginkan.

[ ]:
```
def Z_testStatistic_dual(x̄1,x̄2,d0, 1_pow2, 2_pow2,n1,n2):
    return (float)((x̄1-x̄2)-d0)/np.sqrt( 1_pow2/n1+ 2_pow2/n2)
def Z_testStatistic_dualNormal(p̂1,p̂2,p̂,q̂,n1,n2):
    return (float)(p̂1-p̂2)/np.sqrt(p̂*q̂*(1/n1+1/n2))
def X_testStatistic(n,s, 0):
    return (float)((n-1)*(s**2))/ 0**2
def X_testStatisticDual(s1,s2):
    return (float)(s1**2/s2**2)
def SP(n1,n2,s1_pow2,s2_pow2):
    return (float)(np.sqrt(((n1-1)*s1_pow2+(n2-1)*s2_pow2)/(n1+n2-2)))
def T_testStatistic_dual(x̄1,x̄2,d0,sp,n1,n2):
    return (float)((x̄1-x̄2)-d0)/(sp*np.sqrt(1/n1+1/n2))
```

1.7.1 a. Data kolom Sulfate dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata kedua bagian tersebut sama?

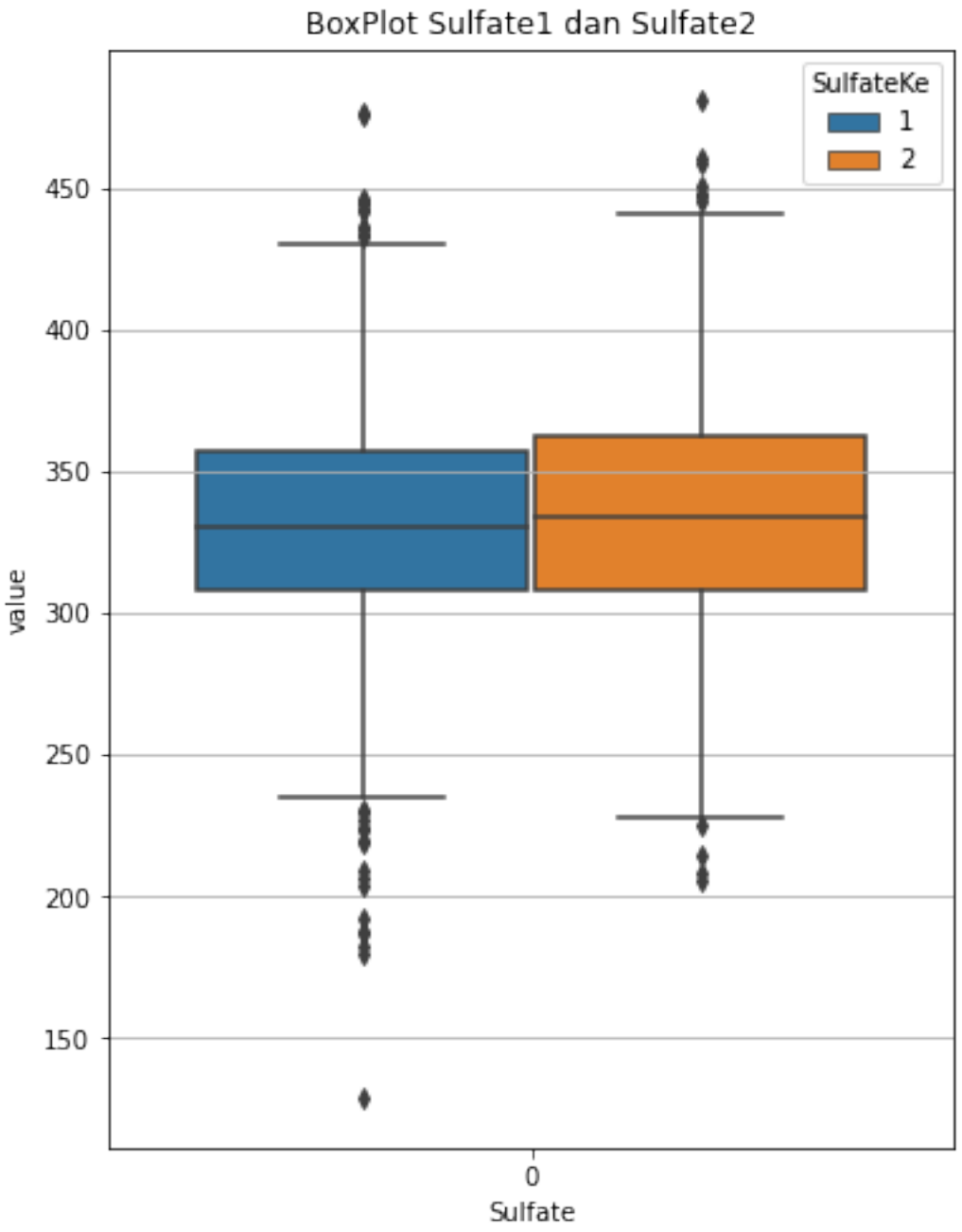
Asumsi pembagian kolom Sulfate menjadi 2 akan menghasilkan dua sampel berbeda dengan tiap sampel berjumlah 1005 baris dari populasi sejumlah 2010 baris. Oleh karena itu, digunakan uji statistik distribusi z untuk dua mean dengan standar deviasi populasi diketahui.

[ ]:
```
print("Data kolom Sulfate dibagi 2 sama rata: bagian awal dan bagian akhir␣
 ↪kolom. Benarkah rata-rata kedua bagian tersebut sama?")

# Langkah 1
H0 = " 1= 2"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " 1  2 ->  1- 2 0"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z _div2 = round(st.norm.ppf(1-( /2)),3)
print("4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui")
print("   Daerah Kritis: z>z /2 atau z<-z /2 : z>{} atau z<{}".
 ↪format(z _div2,z _div2))

# Langkah 5
d0 = 0
Sulfate1 = Sulfate[:len(Sulfate)//2]
```

```
x̄1 = Sulfate1.mean()
 1 = Sulfate1.std()
 1_pow2 =  1**2
n1 = len(Sulfate1)
Sulfate2 = Sulfate[len(Sulfate)//2:]
Sulfate2.reset_index(inplace=True, drop=True)
x̄2 = Sulfate2.mean()
 2 = Sulfate2.std()
 2_pow2 =  2**2
n2 = len(Sulfate2)
z = round(Z_testStatistic_dual(x̄1,x̄2,d0, 1_pow2, 2_pow2,n1,n2),3)
p_value = 1-abs(st.norm.cdf(z)-st.norm.cdf(-1*z))
print("5. Komputasi")
print("   x̄1: {} \n    1: {} \n    1²: {} \n   n1: {} \n   x̄2: {} \n    2: {} \n␣
 ↪   2²: {} \n   n2: {}".format(x̄1, 1, 1_pow2,n1,x̄2, 2, 2_pow2,n2))
print("   d0: {} \n   p_value: {} \n   z: {}".format(d0,str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z _div2 or z < -1*z _div2):
    if (z > z _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (z>z )".format(z,z _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (z<z )".format(z,-1*z _div2))
    print("   Rata-rata kedua bagian Sulfate tidak sama")
else:
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z <z<z )".
 ↪format(-1*z _div2,z,z _div2))
    print("   Rata-rata kedua bagian Sulfate sama")

# Menggambar Boxplot Sulfate1 dan Sulfate2
# Mengubah ke dalam Bentuk Dataframe
dfSulfate1 = pd.DataFrame(data=Sulfate1.tolist()).assign(SulfateKe=1)
dfSulfate2 = pd.DataFrame(data=Sulfate2.tolist()).assign(SulfateKe=2)
# Mengombinasikan Dataframe
combine = pd.concat([dfSulfate1,dfSulfate2],sort=False)
merge = pd.melt(combine, id_vars=['SulfateKe'], var_name=['Sulfate'])
merge
# Konfigurasi Boxplot
plt.figure(figsize=(6,8))
plt.grid()
# Penggambaran Boxplot Gabungan
BoxPlotSulfatem= sns.boxplot(x="Sulfate",y="value",hue="SulfateKe",data=merge)
BoxPlotSulfatem.set_title("BoxPlot Sulfate1 dan Sulfate2")
BoxPlotSulfatem
```

```
Data kolom Sulfate dibagi 2 sama rata: bagian awal dan bagian akhir kolom.
Benarkah rata-rata kedua bagian tersebut sama?
1. H0:  1= 2
2. H1:  1  2 ->  1- 2 0
3.   = 0.05
4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui
   Daerah Kritis: z>z /2 atau z<-z /2 : z>1.96 atau z<1.96
5. Komputasi
   x̄1: 331.30532950549565
    1: 41.332754590968776
    1²: 1708.3966020772505
   n1: 1005
   x̄2: 335.11742332488245
    2: 41.02112948764952
    2²: 1682.7330644425087
   n2: 1005
   d0: 0
   p_value: 0.037986534771476954
   z: -2.075
6. Test Daerah Kritis
   Tolak H0 karena nilai uji = -2.075<-1.96 (z<z )
   Rata-rata kedua bagian Sulfate tidak sama
```

[ ]: <AxesSubplot:title={'center':'BoxPlot Sulfate1 dan Sulfate2'}, xlabel='Sulfate', ylabel='value'>



1.7.2 b. Data kolom OrganicCarbon dibagi 2 sama rata: bagian awal dan bagian akhir kolom. Benarkah rata-rata bagian awal lebih besar dari pada bagian akhir sebesar 0.15?

Asumsi pembagian kolom OrganicCarbon menjadi 2 akan menghasilkan dua sampel berbeda dengan tiap sampel berjumlah 1005 baris dari populasi sejumlah 2010 baris. Oleh karena itu, digunakan uji statistik distribusi z untuk dua mean dengan standar deviasi populasi diketahui.

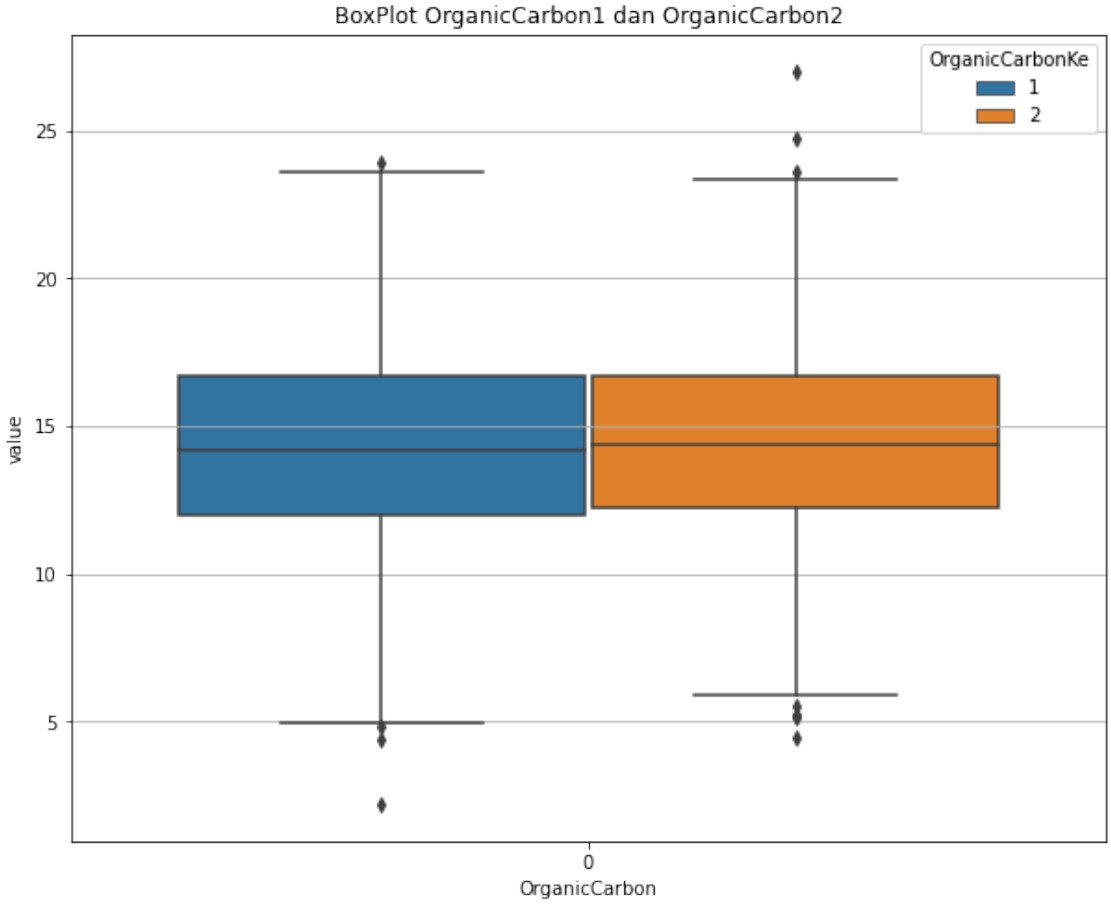
```
[ ]: print("Data kolom OrganicCarbon dibagi 2 sama rata: bagian awal dan bagian␣
      ↪akhir kolom. Benarkah rata-rata bagian awal lebih besar dari pada bagian␣
      ↪akhir sebesar 0.15?")

     # Langkah 1
     H0 = " 1- 2=0.15"
     print("1. H0: {}".format(H0))

     # Langkah 2
     H1 = " 1- 2<0.15"
     print("2. H1: {}".format(H1))

     # Langkah 3
      = 5e-2
     print("3.   = {}".format( ))

     # Langkah 4
     z  = round(st.norm.ppf(1- ),3)
     print("4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui")
     print("   Daerah Kritis: z<-z  : z<{} ".format(-1*z ))

     # Langkah 5
     d0 = 0.15
     OrganicCarbon1 = OrganicCarbon[:len(OrganicCarbon)//2]
     x̄1 = OrganicCarbon1.mean()
      1 = OrganicCarbon1.std()
      1_pow2 =  1**2
     n1 = len(OrganicCarbon1)
     OrganicCarbon2 = OrganicCarbon[len(OrganicCarbon)//2:]
     OrganicCarbon2.reset_index(inplace=True, drop=True)
     x̄2 = OrganicCarbon2.mean()
      2 = OrganicCarbon2.std()
      2_pow2 = 2**2
     n2 = len(OrganicCarbon2)
     z = round(Z_testStatistic_dual(x̄1,x̄2,d0, 1_pow2, 2_pow2,n1,n2),3)
     p_value = round(st.norm.cdf(z),3)
     print("5. Komputasi")
     print("   x̄1: {} \n    1: {} \n    1²: {} \n   n1: {} \n   x̄2: {} \n    2: {} \n␣
      ↪   2²: {} \n   n2: {}".format(x̄1, 1, 1_pow2,n1,x̄2, 2, 2_pow2,n2))
     print("   d0: {} \n   p_value: {} \n   z: {}".format(d0,str(p_value),str(z)))

     # Langkah 6
     print("6. Test Daerah Kritis")
     if (z < -1*z ):
         print("   Tolak H0 karena nilai uji = {}<{} (z<-z )".
      ↪format(str(z),str(-1*z )))
```

```
    print("   Selisih rata-rata bagian awal OrganicCarbon dengan bagian akhir␣
 ↪OrganicCarbon sebesar kurang dari 0.15")
else:
    print("   Terima H0 karena nilai uji = {}<={} (z<=z )".
 ↪format(str(z),str(z _div2)))
    print("   Rata-rata bagian awal OrganicCarbon lebih besar dari pada bagian␣
 ↪akhir OrganicCarbon sebesar 0.15")

# Menggambar Boxplot OrganicCarbon1 dan OrganicCarbon2
# Mengubah ke dalam Bentuk Dataframe
dfOrganicCarbon1 = pd.DataFrame(data=OrganicCarbon1.tolist()).
 ↪assign(OrganicCarbonKe=1)
dfOrganicCarbon2 = pd.DataFrame(data=OrganicCarbon2.tolist()).
 ↪assign(OrganicCarbonKe=2)
# Mengombinasikan Dataframe
combine = pd.concat([dfOrganicCarbon1,dfOrganicCarbon2],sort=False)
merge = pd.melt(combine, id_vars=['OrganicCarbonKe'],␣
 ↪var_name=['OrganicCarbon'])
# Konfigurasi Boxplot
plt.figure(figsize=(10,8))
plt.grid()
# Penggambaran Boxplot Gabungan
BoxPlotOrganicCarbonm= sns.
 ↪boxplot(x="OrganicCarbon",y="value",hue="OrganicCarbonKe",data=merge)
BoxPlotOrganicCarbonm.set_title("BoxPlot OrganicCarbon1 dan OrganicCarbon2")
BoxPlotOrganicCarbonm
```

```
Data kolom OrganicCarbon dibagi 2 sama rata: bagian awal dan bagian akhir kolom.
Benarkah rata-rata bagian awal lebih besar dari pada bagian akhir sebesar 0.15?
1. H0:  1- 2=0.15
2. H1:  1- 2<0.15
3.   = 0.05
4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui
   Daerah Kritis: z<-z  : z<-1.645
5. Komputasi
   x̄1: 14.253972723723393
    1: 3.3511620707420766
    1²: 11.230287224380323
   n1: 1005
   x̄2: 14.461907080372756
    2: 3.2985726887318214
    2²: 10.880581782847477
   n2: 1005
   d0: 0.15
   p_value: 0.008
   z: -2.413
6. Test Daerah Kritis
```

```
   Tolak H0 karena nilai uji = -2.413<-1.645 (z<-z )
   Selisih rata-rata bagian awal OrganicCarbon dengan bagian akhir OrganicCarbon
sebesar kurang dari 0.15
```

[ ]:
```
<AxesSubplot:title={'center':'BoxPlot OrganicCarbon1 dan OrganicCarbon2'},
xlabel='OrganicCarbon', ylabel='value'>
```



1.7.3 c. Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris terakhirnya?

Asumsi pembagian kolom Chloramines menjadi dua, yaitu 100 baris pertama dan 100 baris terakhir akan menghasilkan dua sampel berbeda dari populasi yang sama sejumlah 2010 baris. Oleh karena itu, digunakan uji statistik distribusi z untuk dua mean dengan standar deviasi populasi diketahui.

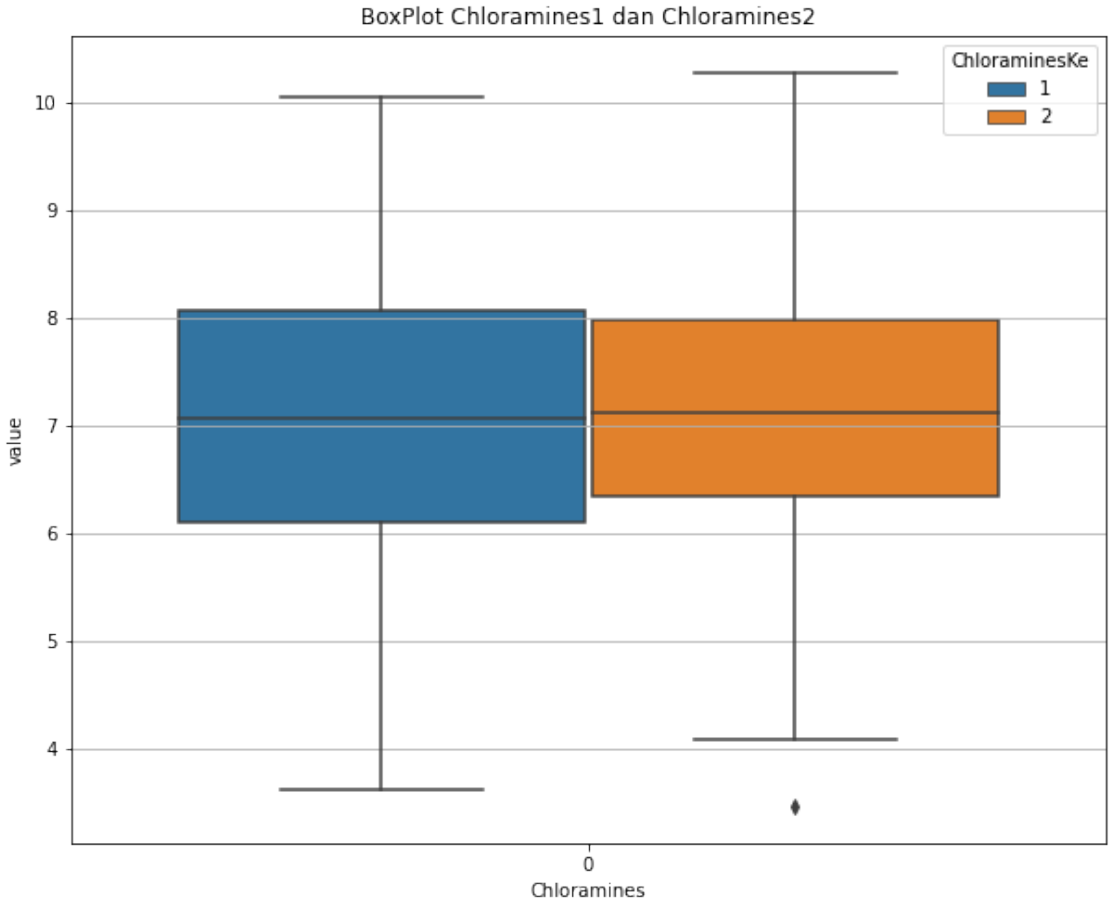
[ ]:
```
print("Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris␣
↪terakhirnya?")

# Langkah 1
H0 = " 1= 2"
```

```
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " 1  2 ->  1- 2 0"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))

# Langkah 4
z _div2 = round(st.norm.ppf(1-( /2)),3)
print("4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui")
print("   Daerah Kritis: z>z /2 atau z<-z /2 : z>{} atau z<{}".
 ↪format(z _div2,z _div2))

# Langkah 5
d0 = 0
Chloramines1 = Chloramines[:100]
x̄1 = Chloramines1.mean()
 1 = Chloramines1.std()
 1_pow2 =  1**2
n1 = len(Chloramines1)
Chloramines2 = Chloramines[len(Chloramines)-100:]
Chloramines2.reset_index(inplace=True, drop=True)
x̄2 = Chloramines2.mean()
 2 = Chloramines2.std()
 2_pow2 =  2**2
n2 = len(Chloramines2)
z = round(Z_testStatistic_dual(x̄1,x̄2,d0, 1_pow2, 2_pow2,n1,n2),3)
p_value = 1-abs(st.norm.cdf(z)-st.norm.cdf(-1*z))
print("5. Komputasi")
print("   x̄1: {} \n    1: {} \n    1²: {} \n   n1: {} \n   x̄2: {} \n    2: {} \n␣
 ↪   2²: {} \n   n2: {}".format(x̄1, 1, 1_pow2,n1,x̄2, 2, 2_pow2,n2))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z _div2 or z < -1*z _div2):
    if (z > z _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (z>z )".format(z,z _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (z<-z )".
 ↪format(z,-1*z _div2))
    print("   Rata-rata 100 baris pertama kolom Chloramines tidak sama dengan␣
 ↪100 baris terakhirnya")
else:
```

```
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z <z<z )".
 ↪format(-1*z _div2,z,z _div2))
    print("   Rata-rata 100 baris pertama kolom Chloramines sama dengan 100␣
 ↪baris terakhirnya")

# Menggambar Boxplot Chloramines1 dan Chloramines2
# Mengubah ke dalam Bentuk Dataframe
dfChloramines1 = pd.DataFrame(data=Chloramines1.tolist()).
 ↪assign(ChloraminesKe=1)
dfChloramines2 = pd.DataFrame(data=Chloramines2.tolist()).
 ↪assign(ChloraminesKe=2)
# Mengombinasikan Dataframe
combine = pd.concat([dfChloramines1,dfChloramines2],sort=False)
merge = pd.melt(combine, id_vars=['ChloraminesKe'], var_name=['Chloramines'])
# Konfigurasi Boxplot
plt.figure(figsize=(10,8))
plt.grid()
# Penggambaran Boxplot Gabungan
BoxPlotChloraminesm= sns.
 ↪boxplot(x="Chloramines",y="value",hue="ChloraminesKe",data=merge)
BoxPlotChloraminesm.set_title("BoxPlot Chloramines1 dan Chloramines2")
BoxPlotChloraminesm
```

```
Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris terakhirnya?
1. H0:  1= 2
2. H1:  1  2 ->  1- 2 0
3.   = 0.05
4. Uji Statistik: z=((x̄1-x̄2)-d0)/sqrt( 1²/n1+ 2²/n2),   diketahui
   Daerah Kritis: z>z /2 atau z<-z /2 : z>1.96 atau z<1.96
5. Komputasi
   x̄1: 7.007771140423921
    1: 1.4808922939392726
    1²: 2.193041986248721
   n1: 100
   x̄2: 7.147197636249925
    2: 1.306806325954166
    2²: 1.707742773553826
   n2: 100
   p_value: 0.4801881374515662
   z: -0.706
6. Test Daerah Kritis
   Terima H0 karena nilai uji = -1.96<-0.706<1.96 (-z <z<z )
   Rata-rata 100 baris pertama kolom Chloramines sama dengan 100 baris
terakhirnya
```

[ ]: <AxesSubplot:title={'center':'BoxPlot Chloramines1 dan Chloramines2'},
xlabel='Chloramines', ylabel='value'>



1.7.4 d. Proporsi nilai bagian awal Turbidity yang lebih dari 4 adalah lebih besar daripada proporsi nilai yang sama di bagian akhir Turbidity?

Digunakan tes dua proporsi.

[ ]:
```
print("Proporsi nilai bagian awal Turbidity yang lebih dari 4 adalah lebih␣
  ↪besar daripada proporsi nilai yang sama di bagian akhir Turbidity?")

# Langkah 1
H0 = "p1=p2 -> p1-p2=0"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = "p1>p2 -> p1-p2>0"
print("2. H1: {}".format(H1))

# Langkah 3
  = 5e-2
print("3.   = {}".format( ))
```

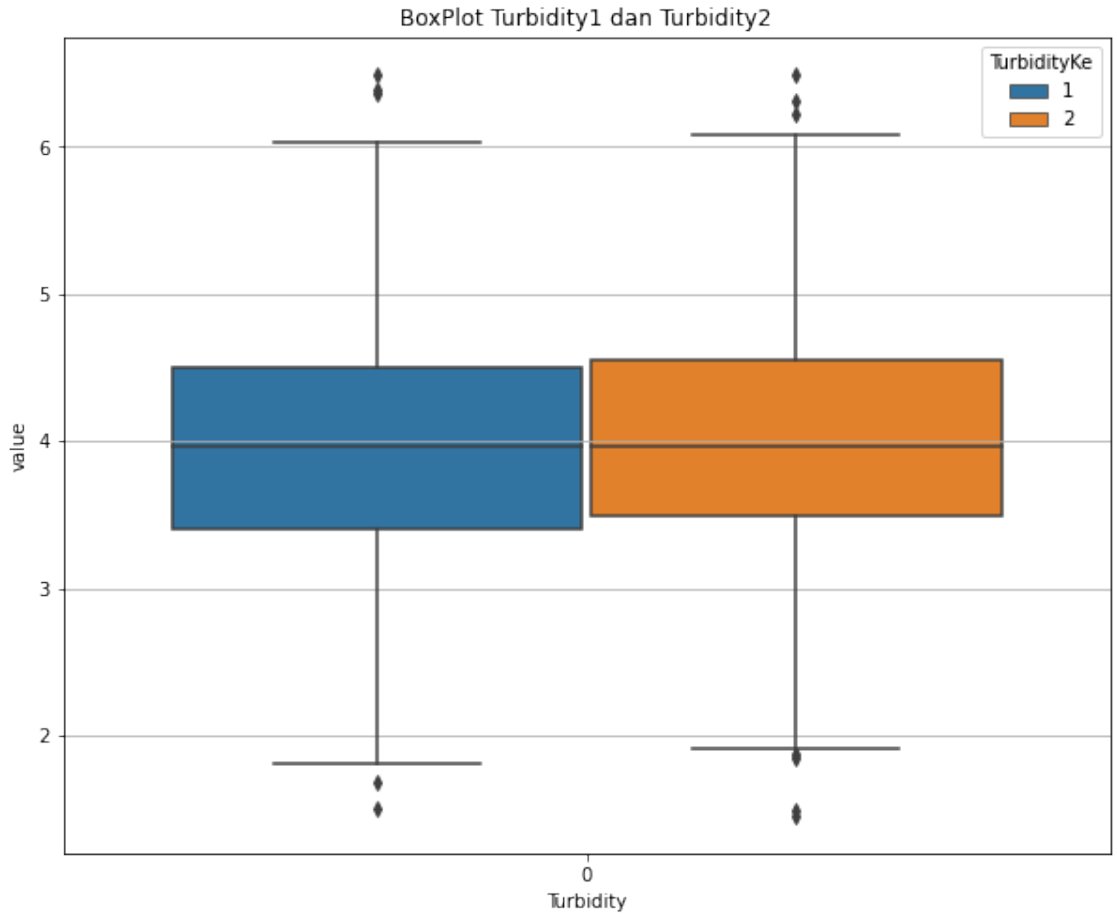
```
# Langkah 4
z  = round(st.norm.ppf(1- ),3)
print("4. Uji Statistik: z=(p̂1-p̂2)/sqrt(p̂*q̂*(1/n1+1/n2)),  diketahui")
print("   Daerah Kritis: z>z  : z>{}".format(z ))

# Langkah 5
Turbidity1 = Turbidity[:len(Turbidity)//2]
Turbidity1_moreThan4 = [turbidity for turbidity in Turbidity1 if turbidity > 4]
n1 = len(Turbidity1)
x1 = len(Turbidity1_moreThan4)
p̂1 = round(float(x1/n1),3)
Turbidity2 = Turbidity[len(Turbidity)//2:]
Turbidity2.reset_index(inplace=True, drop=True)
Turbidity2_moreThan4 = [turbidity for turbidity in Turbidity2 if turbidity > 4]
n2 = len(Turbidity2)
x2 = len(Turbidity2_moreThan4)
p̂2 = round(float(x2/n2),3)
p̂ = round(float((x1+x2)/(n1+n2)),3)
q̂ = 1-p̂
z = round(Z_testStatistic_dualNormal(p̂1,p̂2,p̂,q̂,n1,n2),3)
p_value = 1-st.norm.cdf(z)
print("5. Komputasi")
print("   n1: {} \n   x1: {} \n   p̂1: {} \n   n2: {} \n   x2: {} \n   p̂2: {} \n␣
 ↪  p̂: {} \n   q̂: {}".format(n1,x1,p̂1,n2,x2,p̂2,p̂,q̂))
print("   p_value: {} \n   z: {}".format(str(p_value),str(z)))

# Langkah 6
print("6. Test Daerah Kritis")
if (z > z ):
    print("   Tolak H0 karena nilai uji = {}>{} (z>z )".format(str(z),str(z )))
    print("   Proporsi bagian awal Turbidity yang lebih dari 4 lebih besar dari␣
 ↪proporsi nilai yang sama di bagian akhir Turbidity")
else:
    print("   Terima H0 karena nilai uji = {}<={} (z<=z )".
 ↪format(str(z),str(z )))
    print("   Proporsi bagian awal Turbidity yang lebih dari 4 kurang dari atau␣
 ↪sama dengan proporsi nilai yang sama di bagian akhir Turbidity")

# Menggambar Boxplot Turbidity1 dan Turbidity2
# Mengubah ke dalam Bentuk Dataframe
dfTurbidity1 = pd.DataFrame(data=Turbidity1.tolist()).assign(TurbidityKe=1)
dfTurbidity2 = pd.DataFrame(data=Turbidity2.tolist()).assign(TurbidityKe=2)
# Mengombinasikan Dataframe
combine = pd.concat([dfTurbidity1,dfTurbidity2],sort=False)
merge = pd.melt(combine, id_vars=['TurbidityKe'], var_name=['Turbidity'])
# Konfigurasi Boxplot
```

```
plt.figure(figsize=(10,8))
plt.grid()
# Penggambaran Boxplot Gabungan
BoxPlotTurbiditym= sns.
 ↪boxplot(x="Turbidity",y="value",hue="TurbidityKe",data=merge)
BoxPlotTurbiditym.set_title("BoxPlot Turbidity1 dan Turbidity2")
BoxPlotTurbiditym
```

Proporsi nilai bagian awal Turbidity yang lebih dari 4 adalah lebih besar daripada proporsi nilai yang sama di bagian akhir Turbidity?

1. H0: p1=p2 -> p1-p2=0
2. H1: p1>p2 -> p1-p2>0
3. = 0.05
4. Uji Statistik: z=(p̂1-p̂2)/sqrt(p̂*q̂*(1/n1+1/n2)), diketahui
   Daerah Kritis: z>z : z>1.645
5. Komputasi
   n1: 1005
   x1: 486
   p̂1: 0.484
   n2: 1005
   x2: 489
   p̂2: 0.487
   p̂: 0.485
   q̂: 0.515
   p_value: 0.5536940628343817
   z: -0.135
6. Test Daerah Kritis
   Terima H0 karena nilai uji = -0.135<=1.645 (z<=z)
   Proporsi bagian awal Turbidity yang lebih dari 4 kurang dari atau sama dengan proporsi nilai yang sama di bagian akhir Turbidity

[ ]: <AxesSubplot:title={'center':'BoxPlot Turbidity1 dan Turbidity2'}, xlabel='Turbidity', ylabel='value'>



1.7.5 e. Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian akhirnya?

Digunakan tes variansi. Pada kasus ini digunakan alpha sebesar 0.01 agar alpha/2 yang didapatkan ialah 0.005 dan terdapat pada tabel di buku referensi Walpole.

[ ]:
```
print("Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian␣
↪akhirnya?")

# Langkah 1
H0 = " 1²= 2²"
print("1. H0: {}".format(H0))

# Langkah 2
H1 = " 1² 2² -> 1²- 2² 0"
print("2. H1: {}".format(H1))

# Langkah 3
```

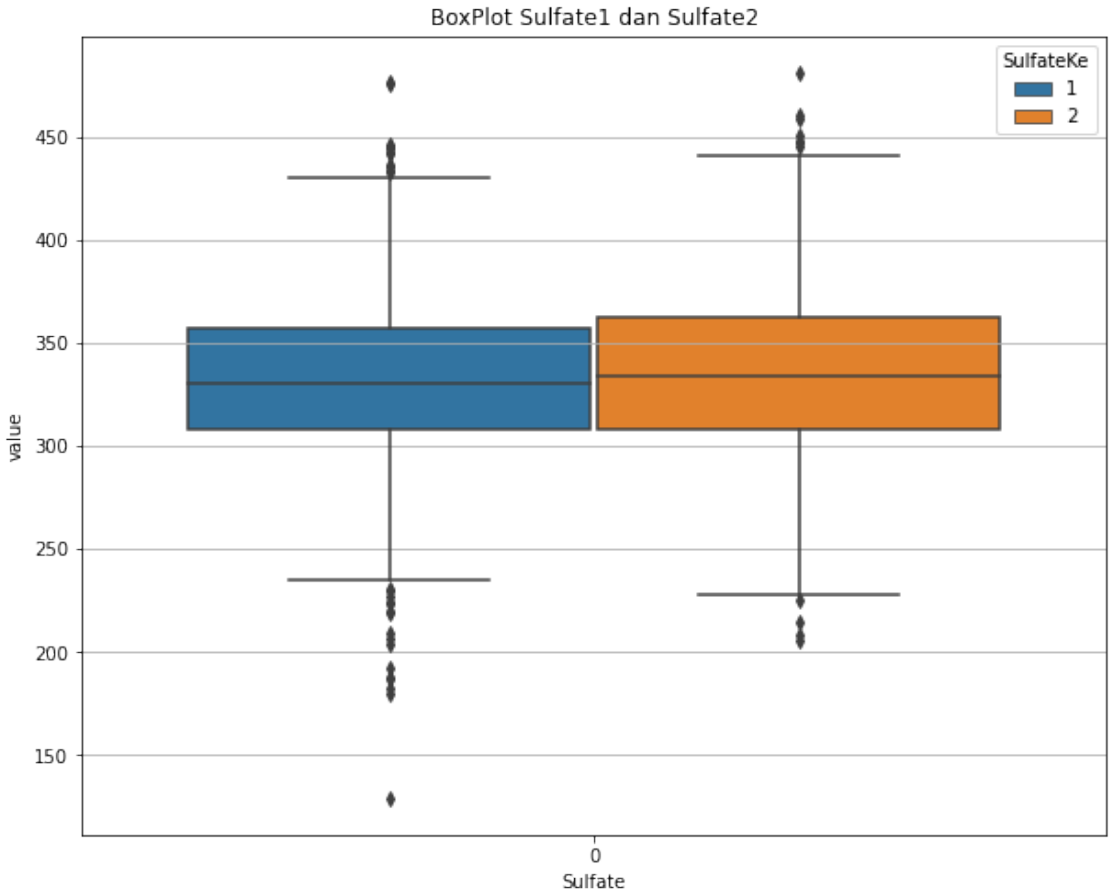
```
 = 1e-1
print("3.   = {}".format( ))

# Langkah 4
Sulfate1 = Sulfate[:len(Sulfate)//2]
Sulfate2 = Sulfate[len(Sulfate)//2:]
Sulfate2.reset_index(inplace=True, drop=True)
n1 = len(Sulfate1)
n2 = len(Sulfate2)
v1 = n1-1
v2 = n2-1
f _div2 = round(st.f.ppf(q=1- /2, dfn=v1, dfd=v2),3)
f1_min_ div2 = round((1.0/round(st.f.ppf(q=1- /2, dfn=v2, dfd=v1),3)),3)
print("4. Uji Statistik: z=((x̄1-x̄2)-d0)/np.sqrt( 1²/n1+ 2²/n2),   diketahui")
print("   Daerah Kritis: f>f /2(v1,v2) atau f<f1- /2(v1,v2) : f>{} atau f<{}".
 ↪format(f _div2,f1_min_ div2))

# Langkah 5
s1 = Sulfate1.std()
s2 = Sulfate2.std()
f = round(X_testStatisticDual(s1,s2),3)
print("5. Komputasi")
print("   n1: {} \n    ²1: {} \n   v1: {} \n   n2: {} \n    ²2: {} \n   v2: {}".
 ↪format(n1,s1**2,v1,n2,s2**2,v2))


# # Langkah 6
print("6. Test Daerah Kritis")
if (f > f _div2 or f < f1_min_ div2):
    if (f > f _div2):
        print("   Tolak H0 karena nilai uji = {}>{} (f > f /2)".
 ↪format(f,f _div2))
    else:
        print("   Tolak H0 karena nilai uji = {}<{} (f < f1- /2)".
 ↪format(f,f1_min_ div2))
    print("   Bagian awal kolom Sulfate memiliki variansi yang tidak sama␣
 ↪dengan bagian akhirnya")
else:
    print("   Terima H0 karena nilai uji = {}<{}<{} (-z <z<z )".
 ↪format(f1_min_ div2,f,f _div2))
    print("   Bagian awal kolom Sulfate memiliki variansi yang sama dengan␣
 ↪bagian akhirnya")

# Menggambar BoxPlot Sulfate1 dan Sulfate2
# Mengubah ke dalam Bentuk Dataframe
dfSulfate1 = pd.DataFrame(data=Sulfate1.tolist()).assign(SulfateKe=1)
```

```
dfSulfate2 = pd.DataFrame(data=Sulfate2.tolist()).assign(SulfateKe=2)
# Mengombinasikan Dataframe
combine = pd.concat([dfSulfate1,dfSulfate2],sort=False)
merge = pd.melt(combine, id_vars=['SulfateKe'], var_name=['Sulfate'])
# Konfigurasi Boxplot
plt.figure(figsize=(10,8))
plt.grid()
# Penggambaran Boxplot Gabungan
BoxPlotSulfatem= sns.boxplot(x="Sulfate",y="value",hue="SulfateKe",data=merge)
BoxPlotSulfatem.set_title("BoxPlot Sulfate1 dan Sulfate2")
BoxPlotSulfatem
```

Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian akhirnya?

1. H0: $\sigma_1^2 = \sigma_2^2$
2. H1: $\sigma_1^2 \neq \sigma_2^2$ -> $\sigma_1^2 - \sigma_2^2 \neq 0$
3. $\alpha = 0.1$
4. Uji Statistik: $z=((\bar{x}1-\bar{x}2)-d0)/np.sqrt(\sigma_1^2/n1+\sigma_2^2/n2)$, diketahui
   Daerah Kritis: f>f $_{\alpha/2}$(v1,v2) atau f<f1- $_{\alpha/2}$(v1,v2) : f>1.109 atau f<0.902
5. Komputasi
   n1: 1005
   σ_1^2: 1708.3966020772505
   v1: 1004
   n2: 1005
   σ_2^2: 1682.7330644425087
   v2: 1004
6. Test Daerah Kritis
   Terima H0 karena nilai uji = 0.902<1.015<1.109 ($-z_{\alpha} < z < z_{\alpha}$)
   Bagian awal kolom Sulfate memiliki variansi yang sama dengan bagian akhirnya

[ ]: <AxesSubplot:title={'center':'BoxPlot Sulfate1 dan Sulfate2'}, xlabel='Sulfate', ylabel='value'>



1.8 Tahap 6 - Tes Korelasi

Mentukan apakah setiap kolom non-target berkorelasi dengan kolom target dengan menggambarkan juga scatter plot nya. Tes korelasi dilakukan dengan menggunakan correlation test.

Daftar kolom non-target adalah sebagai berikut: 1. pH 2. Hardness 3. Solids 4. Chloramines 5. Sulfate 6. Conductivity 7. OrganicCarbon 8. Trihalomethanes 9. Turbidity

Di samping itu, kolom target adalah Potability.

Metode yang digunakan adalah sebagai berikut:

1. Melakukan tes korelasi dengan metode Pearson dengan alpha = 0.05
   - H0 : p = 0 (Tidak ada korelasi diantara kolom non-target dengan kolom target)
   - H1 : p 0 (Ada korelasi antara kolom non-target dan kolom target)
2. Jika terdapat korelasi di antara kolom non-target dan kolom target, cari berapa koefisien korelasi di antara kedua kolom tersebut. Beberapa kesimpulan yang dapat diambil adalah sebagai berikut:
   - Koefisien Korelasi = 1 (Strong Positive Correlation)
   - 0 < Koefisien Korelasi < 1 (Positive Correlation)
   - Koefisien Korelasi = 0 (No Correlation)

- -1 < Koefisien Korelasi < 0 (Negative Correlation)
- Koefisien Korelasi = -1 (Strong Negative Correlation)

3. Tampilkan Scatter Plot diantara dua kolom tersebut

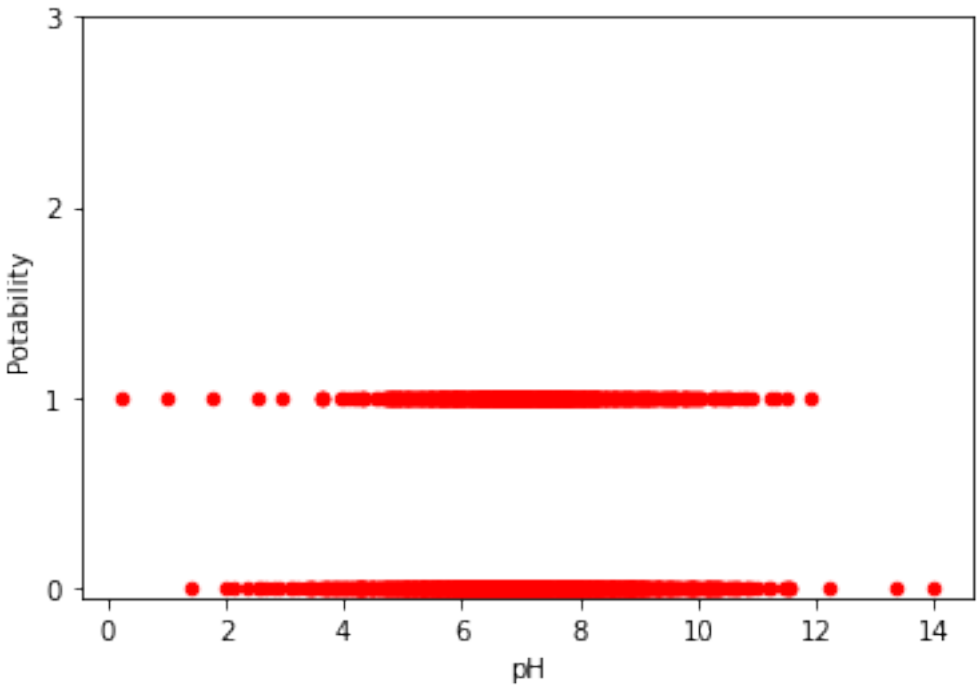
[ ]:
```
# Deklarasi kolom potability dan nilai alpha

Potability = df["Potability"]
alpha = 0.05
```

1.9 1. Tes Korelasi pH dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(pH, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(pH.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='pH', y='Potability', yticks=[0.0, 1.0, 2.0, 3.0],␣
 ↪color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah 0.015475094408433492
```

[ ]:
```
<AxesSubplot:xlabel='pH', ylabel='Potability'>
```

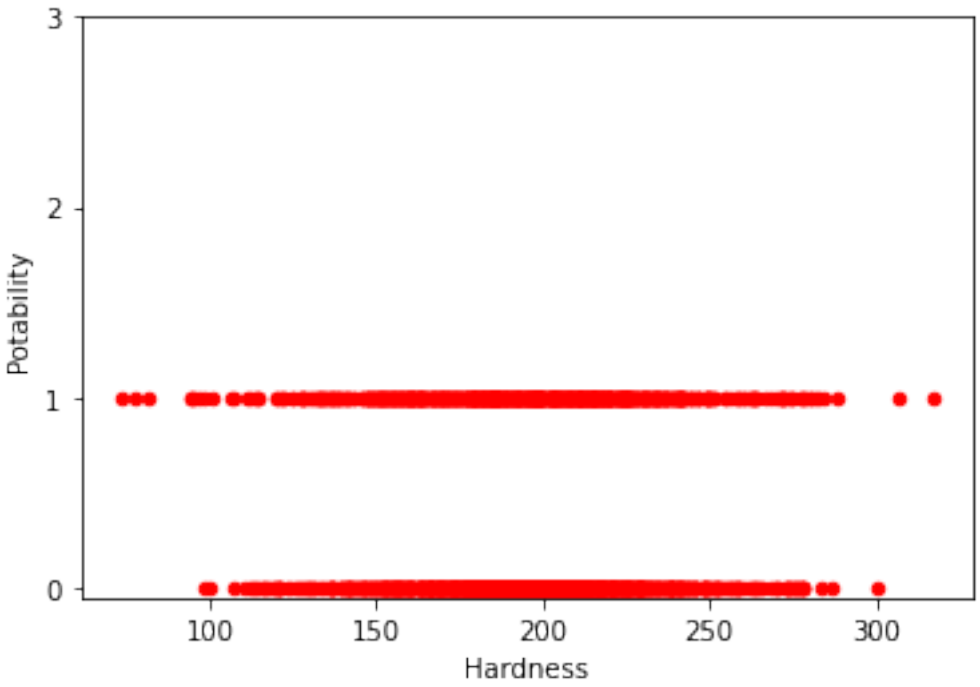


1.10 2. Tes Korelasi Hardness dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Hardness, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Hardness.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Hardness', y='Potability', yticks=[0.0, 1.0, 2.0, 3.
 ↪0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah -0.0014631528959479442
```

[ ]:
```
<AxesSubplot:xlabel='Hardness', ylabel='Potability'>
```

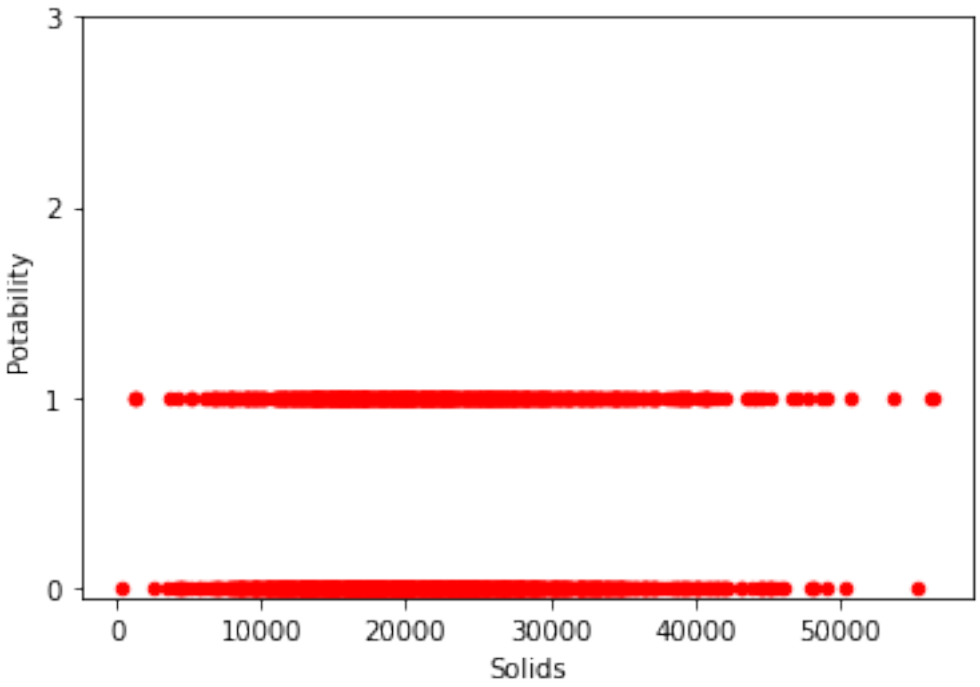


1.11 3. Tes Korelasi Solids dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Solids, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Solids.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Solids', y='Potability', yticks=[0.0, 1.0, 2.0, 3.
 ↪0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah 0.0389765781817347
```

[ ]:
```
<AxesSubplot:xlabel='Solids', ylabel='Potability'>
```

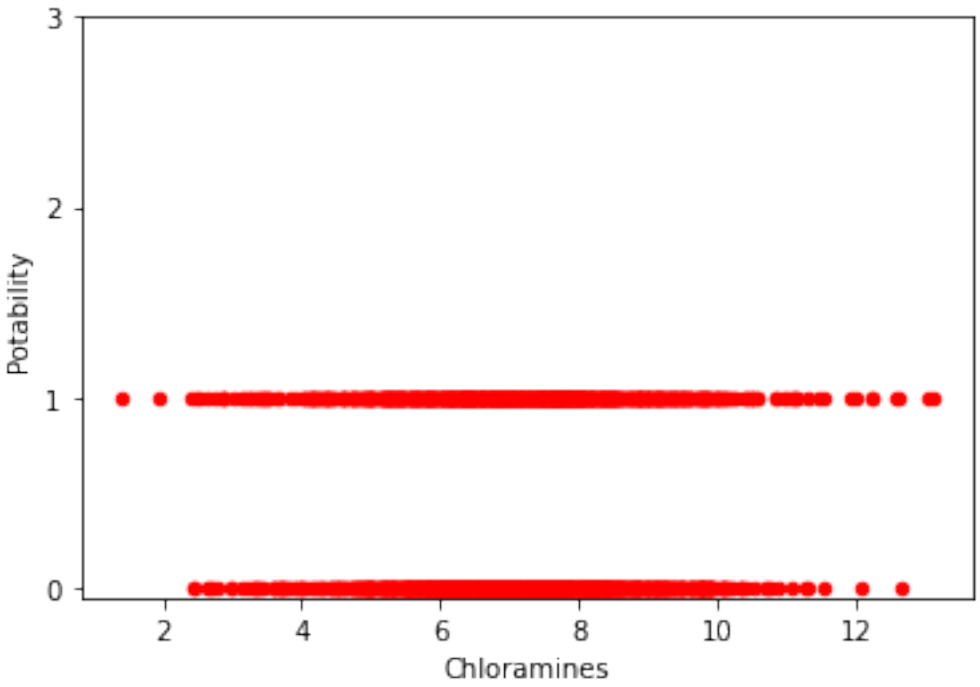


1.12 4. Tes Korelasi Chloramines dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Chloramines, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Chloramines.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Chloramines', y='Potability', yticks=[0.0, 1.0, 2.0,␣
 ↪3.0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah 0.020778921840524087
```

[ ]:
```
<AxesSubplot:xlabel='Chloramines', ylabel='Potability'>
```

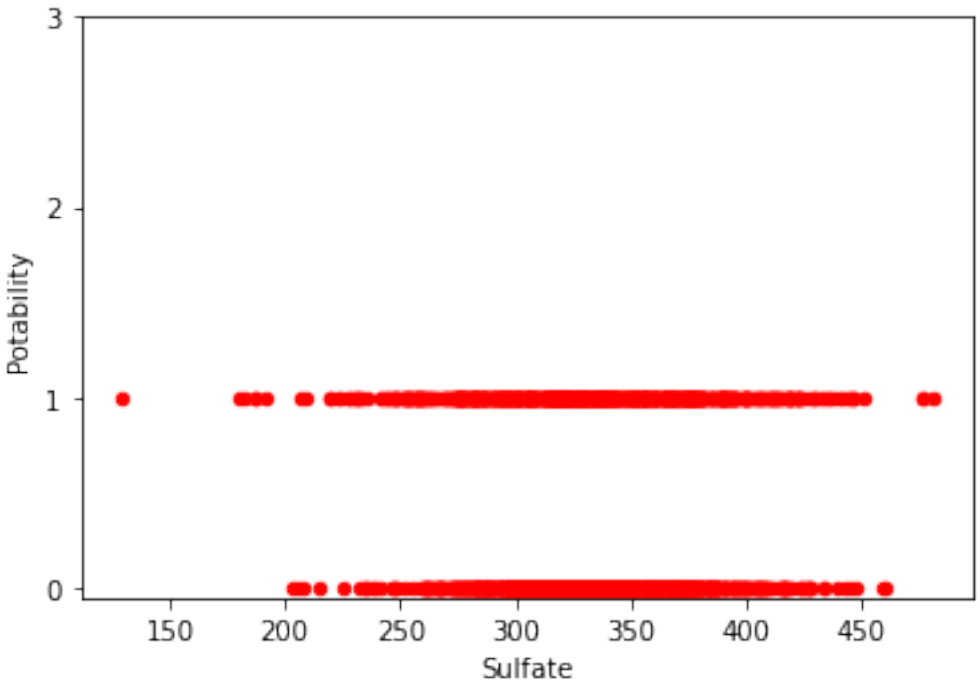


1.13 5. Tes Korelasi Sulfate dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Sulfate, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Sulfate.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Sulfate', y='Potability', yticks=[0.0, 1.0, 2.0, 3.
 ↪0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah -0.01570316441927379
```

[ ]:
```
<AxesSubplot:xlabel='Sulfate', ylabel='Potability'>
```

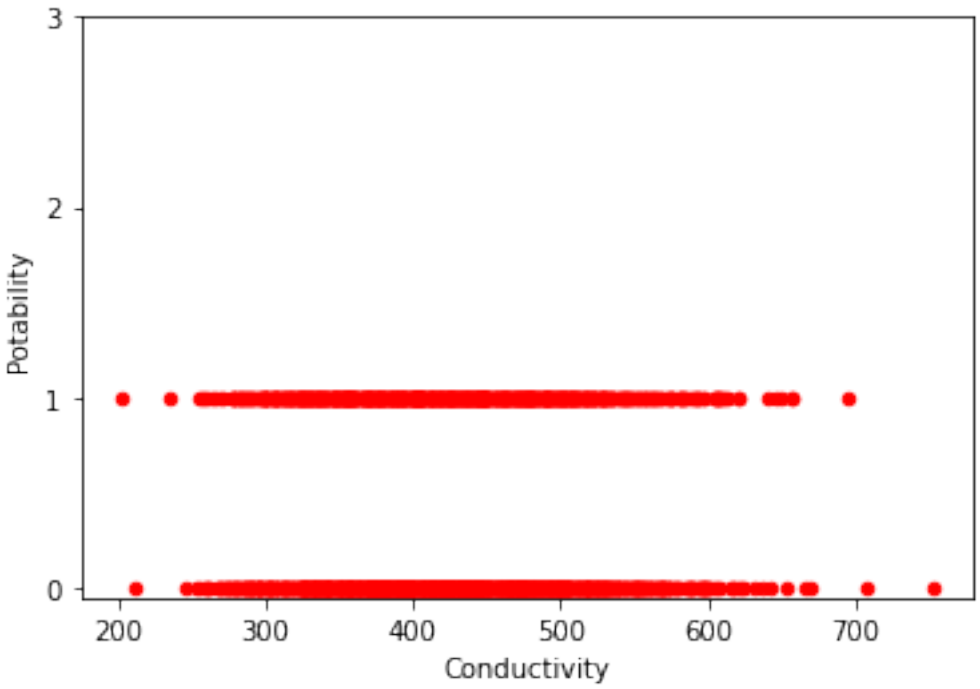


1.14 6. Tes Korelasi Conductivity dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Conductivity, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Conductivity.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Conductivity', y='Potability', yticks=[0.0, 1.0, 2.
 ↪0, 3.0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah -0.016257120111377105
```

[ ]:
```
<AxesSubplot:xlabel='Conductivity', ylabel='Potability'>
```

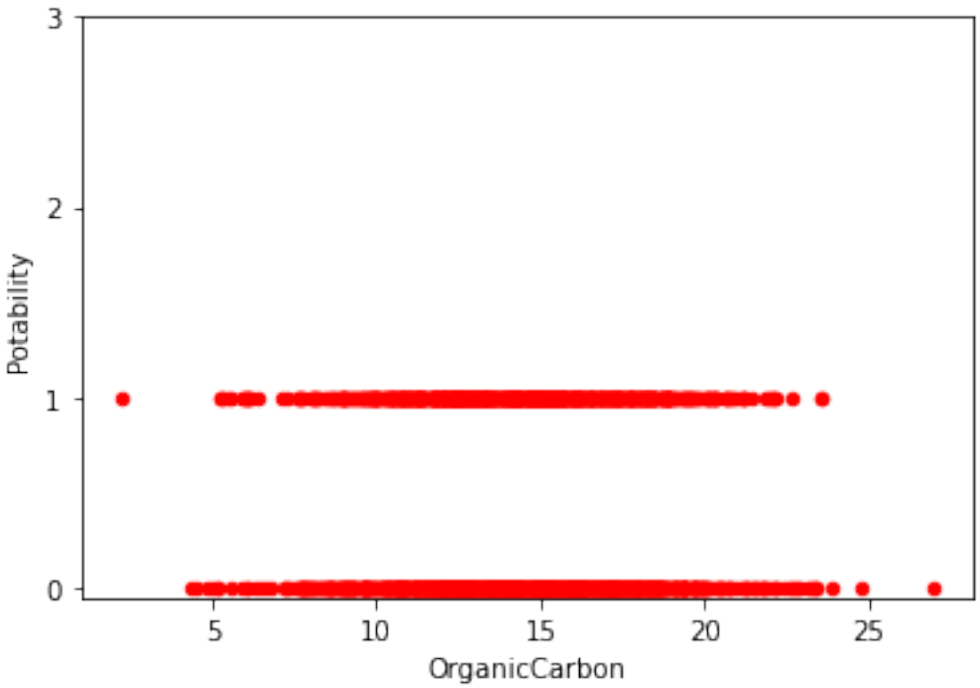


1.15 7. Tes Korelasi OrganicCarbon dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(OrganicCarbon, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(OrganicCarbon.
 ↪corr(Potability, method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='OrganicCarbon', y='Potability', yticks=[0.0, 1.0, 2.
 ↪0, 3.0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah -0.015488461910747282
```

[ ]:
```
<AxesSubplot:xlabel='OrganicCarbon', ylabel='Potability'>
```

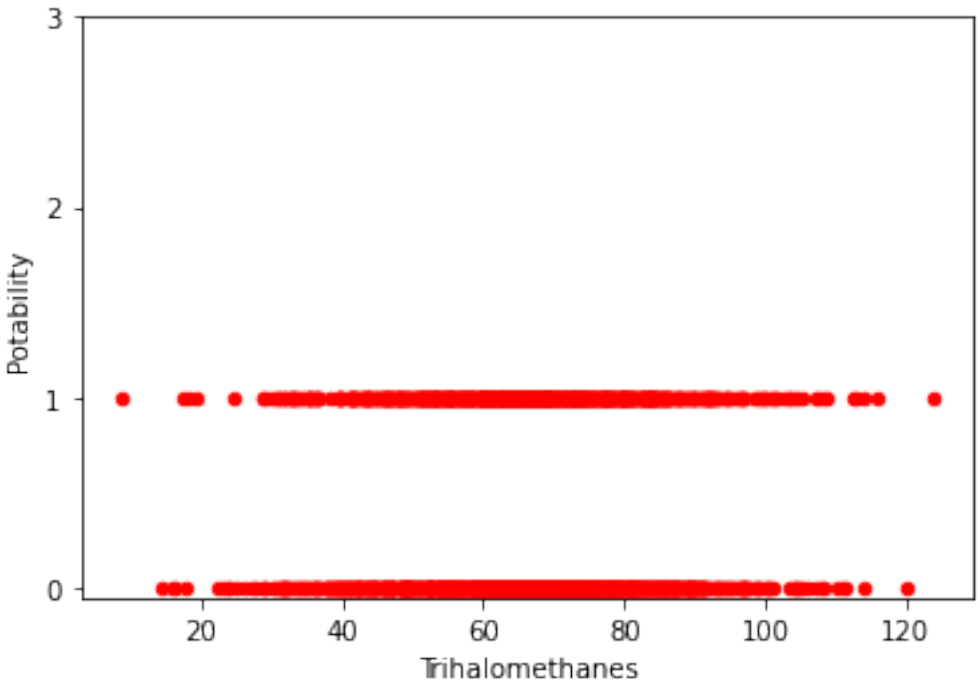


1.16 8. Tes Korelasi Trihalomethanes dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Trihalomethanes, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Trihalomethanes.
 ↪corr(Potability, method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Trihalomethanes', y='Potability', yticks=[0.0, 1.0,␣
 ↪2.0, 3.0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah 0.009236711064713004
```

[ ]:
```
<AxesSubplot:xlabel='Trihalomethanes', ylabel='Potability'>
```

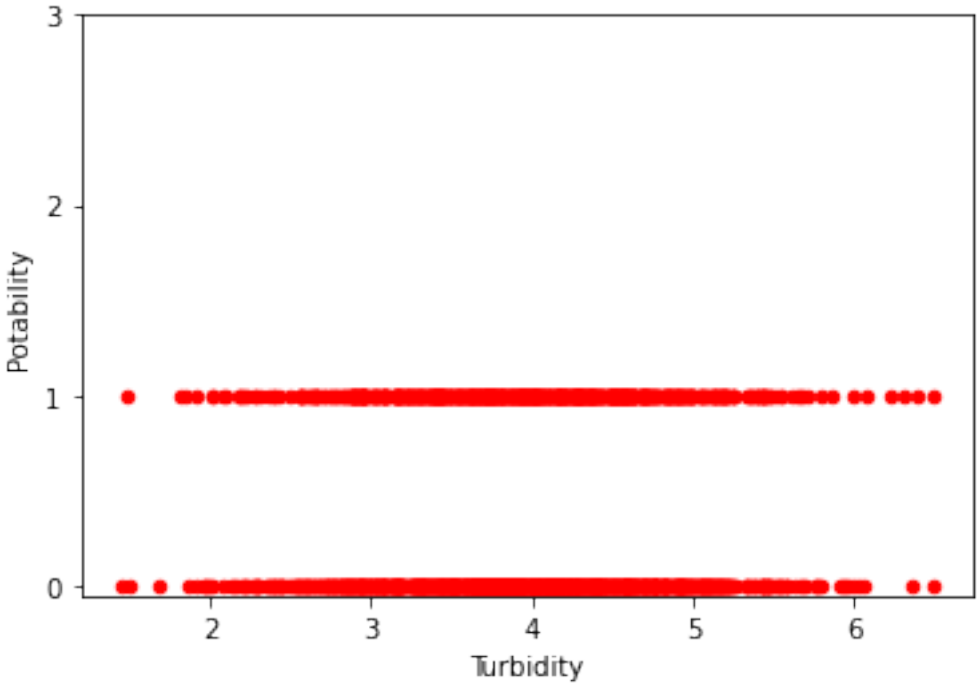


1.17 9. Tes Korelasi Turbidity dengan Potability

[ ]:
```
# Metode bagian 1
stat, p = st.pearsonr(Turbidity, Potability)
if p < alpha:
    print("H0 ditolak, sehingga ada korelasi antara kolom non-target dan kolom␣
 ↪target")
else:
    print("H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom␣
 ↪non-target dan kolom target")

# Metode bagian 2
print("Nilai koefisien korelasi adalah {}".format(Turbidity.corr(Potability,␣
 ↪method='pearson')))

# Metode bagian 3
df.plot(kind='scatter', x='Turbidity', y='Potability', yticks=[0.0, 1.0, 2.0, 3.
 ↪0], color='red')
```

```
H0 tidak bisa ditolak, sehingga tidak ada korelasi antara kolom non-target dan
kolom target
Nilai koefisien korelasi adalah 0.022331042640622675
```

[ ]:
```
<AxesSubplot:xlabel='Turbidity', ylabel='Potability'>
```


3
2
1
0
Potability
2
3
4
5
6
Turbidity