Latihan Pertemuan 4

Nama : Rangga Pebrianto

NIM : G6601231006

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import matplotlib.pyplot as plt
import pandas as pd

# Mount Google Drive jika belum dilakukan
from google.colab import drive
drive.mount('/content/drive')

# Load data dari file CSV di Google Drive
data = pd.read_csv('/content/drive/My Drive/Tugas LKP 4/Iris.csv')  # Gantilah path sesuai dengan lokasi file CSV Anda

# Menampilkan 10 baris pertama dari data
print(data.head(10))

# Menampilkan deskripsi statistik data
data.describe()
```

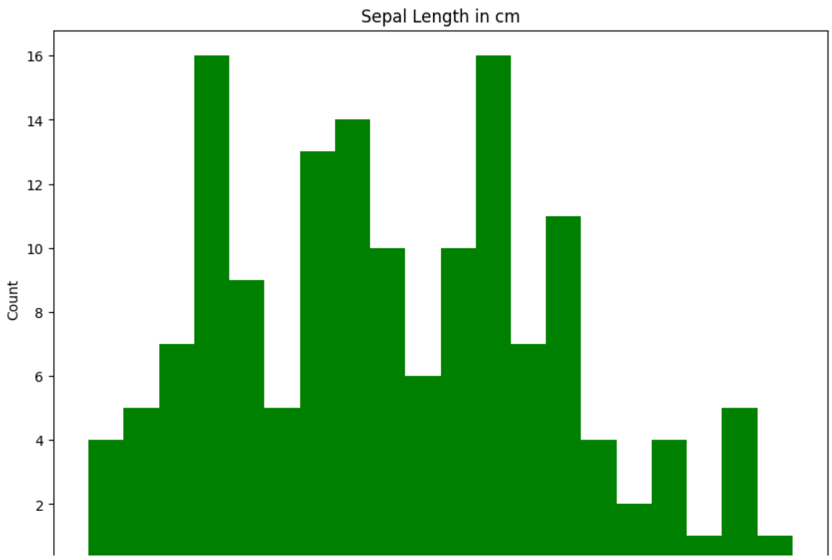
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call driv
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
5           5.4          3.9           1.7          0.4  Setosa
6           4.6          3.4           1.4          0.3  Setosa
7           5.0          3.4           1.5          0.2  Setosa
8           4.4          2.9           1.4          0.2  Setosa
9           4.9          3.1           1.5          0.1  Setosa
```

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
plt.figure(figsize = (10, 7))
x = data["sepal.length"]

plt.hist(x, bins = 20, color = "green")
plt.title("Sepal Length in cm")
plt.xlabel("Sepal_Length_cm")
plt.ylabel("Count")

plt.show()
```

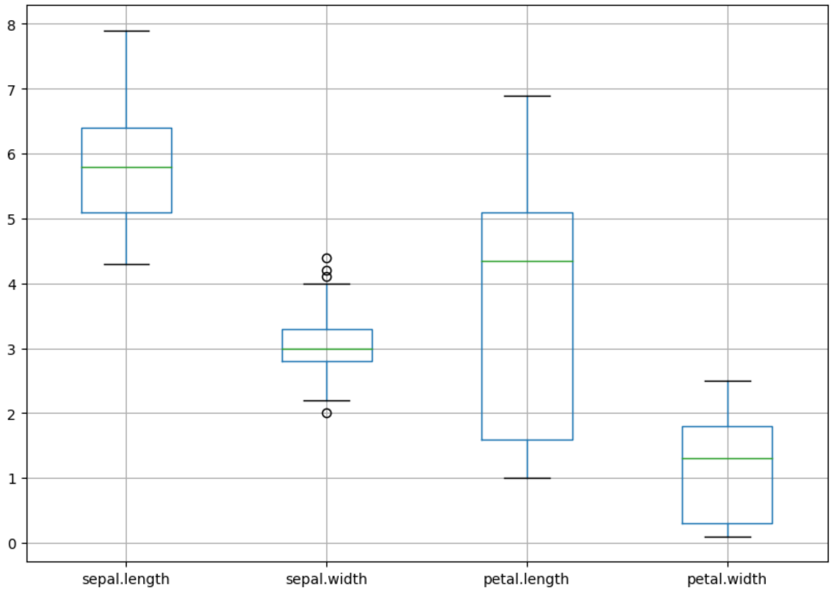


```
# show the box plot
new_data = data[["sepal.length", "sepal.width", "petal.length", "petal.width"]]
print(new_data.head())

plt.figure(figsize = (10, 7))
new_data.boxplot()

plt.show()
```

```
   sepal.length  sepal.width  petal.length  petal.width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
```



Klasifikasi Naïve Bayes dengan Python

```
import pandas as pd
import numpy as np

iris = pd.read_csv("/content/drive/My Drive/Tugas LKP 4/Iris.csv")
iris.head()

#  variabel bebas
x = iris.drop(["variety"], axis = 1)
```

```
x.head()

#variabel tidak bebas
y = iris["variety"]
y.head()
```

```
0    Setosa
1    Setosa
2    Setosa
3    Setosa
4    Setosa
Name: variety, dtype: object
```

```
# classification
# please install scikit library
# pip install -U scikit-learn

# separate the dataset
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 5)
#import from library
from sklearn.naive_bayes import GaussianNB
# Call Gaussian Naive Bayes
iris_model = GaussianNB()
```

```
# Insert the training dataset to  Naive Bayes function
NB_train = iris_model.fit(x_train, y_train)
```

```
# Next step: Prediction the x_test to the model built and save to the  y_pred variable
# show the result of prediction
y_pred = NB_train.predict(x_test)
np.array(y_pred)

# show the y_test based on separation dataset
np.array(y_test)
```

```
array(['Versicolor', 'Virginica', 'Virginica', 'Setosa', 'Virginica',
       'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Versicolor',
       'Versicolor', 'Virginica', 'Virginica', 'Virginica', 'Setosa',
       'Setosa', 'Virginica', 'Virginica', 'Setosa', 'Setosa',
       'Versicolor', 'Virginica', 'Setosa', 'Versicolor', 'Versicolor',
       'Virginica', 'Versicolor', 'Versicolor', 'Versicolor', 'Virginica'],
      dtype=object)
```

```
# this value will show all probability for each predicted class
NB_train.predict_proba(x_test)
```

```
array([[5.30085641e-063, 9.99981862e-001, 1.81383617e-005],
       [2.33789935e-148, 6.30886847e-001, 3.69113153e-001],
       [1.96263359e-196, 4.91363272e-007, 9.99999509e-001],
       [1.00000000e+000, 1.14656751e-018, 4.49822440e-027],
       [5.25023564e-268, 1.22602746e-012, 1.00000000e+000],
       [3.69701688e-035, 9.99999868e-001, 1.32443809e-007],
       [1.00000000e+000, 2.51013111e-017, 5.52818684e-026],
       [2.28418536e-131, 1.07634152e-001, 8.92365848e-001],
       [1.00000000e+000, 5.44261227e-016, 1.20097067e-024],
       [8.96209989e-099, 9.91260655e-001, 8.73934453e-003],
       [3.07281738e-099, 9.89683769e-001, 1.03162310e-002],
       [1.46506375e-128, 7.68601918e-001, 2.31398082e-001],
       [1.77488452e-219, 1.47076918e-007, 9.99999853e-001],
       [2.41148452e-222, 1.76713485e-006, 9.99998233e-001],
       [1.00000000e+000, 3.93692194e-014, 2.48181020e-022],
       [1.00000000e+000, 5.59124686e-011, 3.78482609e-019],
       [2.39814803e-138, 1.07467031e-001, 8.92532969e-001],
       [6.77425372e-218, 1.10894386e-006, 9.99998891e-001],
       [1.00000000e+000, 1.60329612e-015, 1.97866676e-023],
       [1.00000000e+000, 2.21768247e-018, 5.23113060e-027],
       [7.81027411e-072, 9.99854360e-001, 1.45639852e-004],
       [1.84145581e-198, 7.60711220e-006, 9.99992393e-001],
       [1.00000000e+000, 4.78405317e-018, 5.44647058e-026],
       [1.00076675e-119, 9.46810579e-001, 5.31894215e-002],
       [1.74217049e-073, 9.99826922e-001, 1.73077694e-004],
       [2.93596473e-203, 1.39231762e-007, 9.99999861e-001],
       [9.79471405e-084, 9.99040188e-001, 9.59812045e-004],
       [1.19325218e-077, 9.99775997e-001, 2.24002949e-004],
       [5.43425264e-104, 9.89719162e-001, 1.02808383e-002],
       [4.62201633e-207, 9.85874738e-007, 9.99999014e-001]])
```

```
# show the confusion matrix based on the prediction result
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

```
array([[ 8,  0,  0],
       [ 0, 10,  1],
       [ 0,  2,  9]])
```

```
#evaluate performance from the confusion matrix
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00         8
  Versicolor       0.83      0.91      0.87        11
   Virginica       0.90      0.82      0.86        11

    accuracy                           0.90        30
   macro avg       0.91      0.91      0.91        30
weighted avg       0.90      0.90      0.90        30
```