In [40]:
```
import numpy as np
import pandas as pd
import pickle

dataset1 = pd.read_csv('data_train.csv')
dataset2 = pd.read_csv('data_validation.csv')
```

In [41]:
```
class NaiveBayes:

    def datasplitting(self, data1, data2):
        self.x_train = data1.iloc[:, :-1].values
        self.x_test = data2.iloc[:, :-1].values
        self.y_train = data1.iloc[:, -1].values
        self.y_test = data2.iloc[:, -1].values
        return self.x_train, self.x_test, self.y_train, self.y_test

    def save_model(self, filename):
        with open(filename, 'wb') as file:
            pickle.dump(self, file)

    @classmethod
    def load_model(cls, filename):
        with open(filename, 'rb') as file:
            return pickle.load(file)

    def fit(self, X, y):
        # Kelas untuk kolom target
        self.classes = np.unique(y)

        # Menghitung rata-rata, variansi, dan prior untuk setiap kelas kolom target
        self.mean = []
        self.var = []
        self.priors = []

        for idx, classes in enumerate(self.classes):
            x_class = X[y == classes] # Ambil data dengan kelas tertentu
            self.mean.append(x_class.mean(axis=0))
            self.var.append(x_class.var(axis=0))
            self.priors.append(len(x_class) / len(X))

    def predict(self, x_test):
        predictionY = []

        # Menghitung posterior untuk setiap kelas dan mencari kelas dengan posterior tertinggi
        for test_row in x_test:
            posteriorList = []
            for index, classes in enumerate(self.classes):
                # Hitung probabilitas dari setiap kelas
                priors = np.log(self.priors[index])
                # Hitung probabilitas dari setiap atribut dengan distribusi Gaussian
                likelihood = np.sum(np.log(self.GaussianFormula(index, test_row)))
                posterior = priors + likelihood
                posteriorList.append(posterior)
            predictionY.append(self.classes[np.argmax(posteriorList)]) # Cari posterior tertinggi

        return np.array(predictionY) # Mengembalikan hasil prediksi

    def GaussianFormula(self, index, x):
        rata2 = self.mean[index]
        variansi = self.var[index]
        return (1 / (np.sqrt(2 * np.pi * variansi))) * np.exp(-((x - rata2) ** 2) / (2 * variansi))
```

In [42]:
```
nb = NaiveBayes()
x_train, x_test, y_train, y_test = nb.datasplitting(dataset1, dataset2)
nb.fit(x_train, y_train)
predictions = nb.predict(x_test)

print("Kolom target data validasi:")
print(y_test)
print()
print("Hasil prediksi:")
print(predictions)
print()

print("Jumlah kolom target data validasi yang sama dengan hasil prediksi:", np.sum(y_test == predictions))
print("Jumlah baris total data validasi:", len(y_test))
print()
accuracy = np.sum(y_test == predictions) / len(y_test) # Perhitungan akurasi diukur dari kolom target validasi yang sama dengan hasil prediksi dibagi jumlah baris totaldata validasi

# Print Confusion matrix
confusion_matrix = pd.crosstab(y_test, predictions, rownames=['Validation'], colnames=['Predicted'])
print(confusion_matrix)
print()

print("Hasil akurasi dengan Naive-Bayes sebesar", accuracy)
print()

print("Simpan dan load model Naive-Bayes...")
nb.save_model('naive_bayes_model.txt')
loaded_nb = NaiveBayes.load_model('naive_bayes_model.txt')
print()

predictions = loaded_nb.predict(x_test)
accuracy = np.sum(y_test == predictions) / len(y_test)

print("Hasil akurasi dengan Naive-Bayes setelah load model sebesar", accuracy)
```

```
Kolom target data validasi:
[1 2 3 0 3 1 3 0 3 2 3 2 3 0 3 0 2 1 1 2 3 2 0 1 2 0 3 1 0 3 1 3 3 0 2 3 1
 3 2 1 1 2 0 2 3 1 1 2 2 3 2 2 3 0 1 3 1 3 3 2 2 3 3 1 3 2 3 2 3 3 2 3 1 0
 1 2 0 3 1 0 3 3 1 2 3 2 3 3 0 2 1 1 1 2 2 1 3 2 0 3 3 3 1 2 3 1 3 3 3 3 3
 3 2 2 3 1 0 2 1 3 1 2 2 3 3 0 2 2 1 3 3 1 0 0 3 1 0 2 3 0 1 3 3 1 2 3 1 2
 1 1 3 0 0 2 1 1 2 0 1 3 3 3 0 2 3 0 0 2 1 2 2 1 0 1 0 1 3 1 2 0 3 1 1 2 0
 2 0 3 2 0 3 2 0 0 2 0 1 3 3 1 1 2 2 3 2 3 3 3 0 2 1 2 3 1 1 2 3 0 0 2 2 1
 2 3 1 2 0 3 0 2 2 2 2 2 3 0 2 3 3 3 0 0 1 0 3 3 1 1 0 2 2 1 3 3 0 2 2 0 0
 1 0 2 2 3 3 0 2 3 3 0 0 2 1 2 1 0 3 2 2 2 0 1 0 2 1 1 0 1 3 1 3 0 1 0 1 3
 2 3 0 3 1 0 0 1 2 2 0 3 1 1 0 2 3 1 3 2 3 1 2 3 0 0 3 1 2 1 0 0 1 0 3 2 3
 2 2 3 3 0 1 0 1 1 0 2 2 2 2 2 1 1 3 0 2 2 2 2 1 3 3 0 3 0 0 3 2 0 3 2 1 2
 0 0 2 2 2 3 0 0 2 2 0 2 1 1 3 2 3 3 1 1 0 3 1 0 3 1 0 2 0 3 1 0 1 0 1 2 1
 1 1 2 3 1 2 2 3 1 1 2 2 2 0 1 3 0 0 2 0 0 0 3 0 1 1 0 0 2 1 0 3 2 1 0 1 1
 2 1 1 1 2 0 1 2 0 1 0 3 0 0 1 2 2 1 3 1 1 3 3 0 2 2 0 1 2 1 0 1 2 0 2 0 2
 3 0 0 3 0 1 3 3 3 2 2 3 0 1 2 1 2 1 0 3 1 3 3 2 1 3 0 3 3 1 3 0 3 0 2 2 0
 2 3 1 3 0 2 3 3 0 2 1 2 3 2 3 1 2 0 3 3 3 1 1 3 1 0 0 0 3 0 2 2 1 3 2 1 2
 3 0 3 0 0 3 3 2 0 1 2 0 2 0 1 0 2 0 1 0 0 1 3 0 1 1 1 0 0 2 0 1 3 0 2 1 1
 0 2 3 1 3 0 2 3]

Hasil prediksi:
[2 2 3 0 3 1 3 0 3 1 3 2 3 0 3 0 2 1 0 2 3 1 0 1 1 1 2 2 0 2 2 3 3 0 2 3 2
 3 1 1 0 3 0 2 2 1 1 2 1 3 1 2 3 0 1 3 2 3 3 2 2 3 3 1 3 2 2 2 2 3 2 3 1 0
 1 2 0 3 1 0 3 3 0 2 3 1 3 3 0 2 1 1 1 2 1 1 3 2 1 3 3 3 1 2 3 2 3 2 3 3 3
 3 2 2 3 2 0 3 1 3 1 2 2 3 3 1 2 2 1 3 3 1 0 0 3 0 0 1 3 0 1 3 3 1 2 3 1 2
 1 2 3 1 0 2 1 0 3 0 0 3 3 3 0 2 3 0 1 1 0 2 3 1 0 1 1 1 3 0 2 0 3 1 1 2 0
 2 0 3 2 0 3 2 0 1 2 0 1 3 3 1 2 2 2 3 2 3 3 3 0 2 0 2 3 1 2 3 2 0 0 2 2 2
 3 3 0 2 0 3 0 2 2 2 2 1 3 0 1 3 3 3 0 0 1 1 3 3 2 2 0 2 1 2 2 3 0 2 2 1 1
 2 0 3 2 3 3 0 1 3 3 0 0 2 1 1 1 0 3 2 2 2 0 1 0 1 1 1 0 0 3 0 3 0 1 0 1 3
 2 2 0 2 1 0 0 1 2 2 0 3 1 1 0 2 3 1 3 1 3 1 2 2 0 0 2 1 2 1 0 1 1 0 2 2 3
 2 2 3 3 0 1 0 1 2 0 2 2 2 2 3 1 2 3 0 2 1 3 3 2 3 2 0 3 0 0 3 2 0 3 2 1 2
 1 0 1 2 1 3 0 0 2 1 0 2 1 0 3 2 3 3 1 0 0 3 1 0 3 0 0 2 0 3 1 0 1 1 1 2 0
 1 1 3 3 1 2 2 3 1 1 2 2 2 0 1 3 0 0 1 0 0 0 3 1 1 2 0 0 2 1 0 3 2 1 0 2 0
 2 1 2 1 2 0 1 2 0 2 0 3 0 0 1 2 1 1 3 2 1 2 3 0 2 2 0 2 2 2 0 1 2 0 2 0 1
 3 0 0 3 0 2 2 3 3 1 3 3 0 1 2 1 3 1 0 3 1 3 3 1 2 3 0 3 3 1 2 0 3 0 2 2 0
 2 3 0 3 0 2 3 3 0 2 1 2 3 3 3 1 2 0 3 2 3 2 2 3 1 1 0 0 3 0 2 1 1 3 2 1 1
 3 0 3 0 0 3 3 3 1 2 2 0 2 0 1 0 2 0 2 0 0 2 3 0 1 2 1 0 1 2 0 1 3 0 2 1 1
 0 1 3 1 3 0 1 3]

Jumlah kolom target data validasi yang sama dengan hasil prediksi: 469
Jumlah baris total data validasi: 600

Predicted     0   1    2    3
Validation
0           125  17    0    0
1            18  93   33    0
2             0  30  110   15
3             0   0   18  141

Hasil akurasi dengan Naive-Bayes sebesar 0.7816666666666666
```

Simpan dan load model Naive-Bayes...

Hasil akurasi dengan Naive-Bayes setelah load model sebesar 0.7816666666666666