LAPORAN PRAKTIKUM
KECERDASAN BUATAN
"UTS"

NAMA : **AFRIDHO IKHSAN**
NPM : **2210631170002**
KELAS : **4A**

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2023

DAFTAR ISI

SOAL

Soal Tipe 1 (Naive Bayes) :
Sebuah perusahaan e-commerce ingin memprediksi apakah sebuah produk akan memiliki tingkat penjualan yang tinggi berdasarkan beberapa fitur terkait. Terdapat 5 fitur: kategori produk, harga produk, rating produk, jumlah ulasan, dan promosi yang dilakukan. Diberikan data latih dalam bentuk file CSV (sales.csv). Poin-poin ketentuan: 1. Hitung probabilitas setiap fitur dalam setiap kategori tingkat penjualan (Tinggi atau Rendah). 2. Hitung probabilitas tingkat penjualan (Tinggi atau Rendah). 3. Tentukan apakah sebuah produk dengan kategori elektronik, harga 1 juta, rating 4.5, jumlah ulasan 100, dan tidak sedang dalam promosi akan memiliki tingkat penjualan tinggi atau rendah.
Berikut adalah dataset dalam bentuk tabel Excel untuk soal tersebut : silahkan download

Soal Tipe 2 (Fuzzy Logic) :
Sebuah perusahaan manufaktur ingin meningkatkan efisiensi proses produksi mereka dengan mengoptimalkan penggunaan energi pada mesin-mesin pabrik. Mereka memutuskan untuk menggunakan sistem kontrol fuzzy untuk mengatur kecepatan mesin berdasarkan beban kerja dan suhu lingkungan. Berikut adalah langkah-langkah yang harus diperhatikan:

1. Tentukan variabel input dan output menggunakan Antecedent dan Consequent.
2. Buat fungsi keanggotaan untuk variabel beban kerja, suhu lingkungan, dan kecepatan mesin
menggunakan trimf.
3. Buat aturan fuzzy menggunakan ctrl.Rule.
4. Buat sistem kontrol fuzzy menggunakan ctrl.ControlSystem.
5. Tampilkan grafik fungsi keanggotaan untuk variabel beban kerja, suhu lingkungan, dan kecepatan mesin.

CATATAN : Langkah-langkah ini akan membantu perusahaan dalam mengatur kecepatan mesin secara cerdas berdasarkan kondisi kerja dan lingkungan, sehingga dapat mengurangi konsumsi energi dan meningkatkan efisiensi produksi mereka.

TANTANGAN :
1. Variabel input: Beban Kerja (dalam persentase), Suhu Lingkungan (dalam derajat Celsius),
variabel output: Kecepatan Mesin (dalam RPM).
2. Gunakan nilai interval dari 0 hingga 100 untuk beban kerja, 20 hingga 40 derajat Celsius
untuk suhu lingkungan, dan 0 hingga 3000 untuk kecepatan mesin.
3. Buat fungsi keanggotaan untuk beban kerja: ringan, sedang, berat.
4. Buat fungsi keanggotaan untuk suhu lingkungan: rendah, sedang, tinggi.
5. Buat fungsi keanggotaan untuk kecepatan mesin: lambat, sedang, cepat.
6. Tentukan aturan fuzzy berdasarkan analisis ahli teknik mesin.

7. Tampilkan grafik fungsi keanggotaan untuk beban kerja, suhu lingkungan, dan kecepatan mesin menggunakan matplotlib.

Berikut dibawah ini adalah kode Python lengkap untuk membuat dataset dan menyimpannya ke dalam file CSV:

```
import pandas as pd
import numpy as np

# Generate random data
np.random.seed(42)
n_samples = 1000

# Variabel input
beban_kerja = np.random.randint(0, 101, n_samples) # Rentang nilai 0-100
suhu_lingkungan = np.random.uniform(20, 40, n_samples) # Rentang nilai 20-40 derajat Celsius

# Variabel output
kecepatan_mesin = np.random.randint(0, 3001, n_samples) # Rentang nilai 0-3000 RPM

# Create DataFrame
df = pd.DataFrame({'Beban Kerja (%)': beban_kerja,
'Suhu Lingkungan (C)': suhu_lingkungan,
'Kecepatan Mesin (RPM)': kecepatan_mesin})

# Save to CSV file
df.to_csv('production_data.csv', index=False)
print("Dataset berhasil dibuat dan disimpan sebagai 'production_data.csv'")
```

JAWABAN

1. Untuk melakukan ketiga hal tersebut, kita bisa menggunakan konsep naive bayes, berikut hal-hal yang perlu dilakukan:
   1.1. Melakukan impor library eksternal yang dibutuhkan dalam perhitungan

```
from csv import reader
from random import seed
from random import randrange
from math import sqrt
from math import exp
from math import pi
```

   - from random import randrange: Mengimpor fungsi randrange dari modul random. Fungsi ini digunakan untuk memilih angka acak dari rentang tertentu.
   - from math import sqrt, exp, pi: Mengimpor fungsi sqrt, exp, dan konstanta pi dari modul math. Fungsi dan konstanta ini akan digunakan dalam perhitungan statistik dan probabilitas dalam algoritma Naive Bayes.

   1.2. Mendefinisikan fungsi-fungsi yang nantinya akan digunakan untuk perhitungan
   - Pembuatan fungsi untuk pemrosesan data (str_column_to_int)

```
def str_column_to_int (dataset, column):
    class_values = [row[column] for row in dataset]
    unique = set(class_values)
    lookup = dict()
    for i, value in enumerate (unique):
        lookup[value] = i
    for row in dataset:
        row[column] = lookup[row[column]]
    return lookup
```

   fungsi str_column_to_int digunakan untuk mengonversi kolom kelas (label) dari string ke integer.

   - Pembuatan fungsi untuk pembagian Validasi Silang

```
def cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = int(len(dataset) / n_folds)
    for _ in range(n_folds):
      fold = list()
      while len(fold) < fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
      dataset_split.append(fold)
    return dataset_split
```

Fungsi cross_validation_split membagi dataset menjadi beberapa bagian untuk keperluan validasi silang (cross-validation).

- Pembuatan fungsi untuk perhitungan akurasi metriks

```
def accuracy_metric(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    return correct / float(len(actual)) * 100.0
```

Fungsi accuracy_metric menghitung akurasi prediksi dengan membandingkan nilai aktual dengan nilai prediksi.

- Pembuatan fungsi untuk mengevaluasi Algoritma

```
def evaluate_algorithm(dataset, algorithm, n_folds, args):
    folds = cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        accuracy = accuracy_metric(actual, predicted)
        scores.append(accuracy)
    return scores
```

Fungsi evaluate_algorithm mengevaluasi algoritma menggunakan cross_validation_split.

- Pembuatan fungsi untuk pemisahan data berdasarkan class

```
def separate_by_class(dataset):
    separated = dict()
    for i in range(len(dataset)):
        vector = dataset[i]
        class_value = vector[-1]
        if (class_value not in separated):
            separated[class_value] = list()
        separated[class_value].append(vector)
    return separated
```

Fungsi separate_by_class memisahkan dataset menjadi subset berdasarkan nilai kelas (label).

- Pembuatan fungsi-fungsi yang diperlukan untuk meringkas kumpulan data

```
def mean(numbers):
    return sum(numbers)/float(len(numbers))
```

```
def stdev(numbers):
    avg = mean (numbers)
    variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
    return sqrt(variance)
```

```
def summarize_dataset(dataset):
    summaries = [(mean(column), stdev(column), len(column)) for column in zip(*dataset)]
    del(summaries[-1])
    return summaries
```

Fungsi yang diperlukan diantaranya adalah fungsi untuk perhitungan mean, standar deviasi, dan pada akhirnya digunakan pada fungsi summarize_dataset untuk meringkas data-data numerik pada setiap kolom menjadi data mengenai rata-rata, deviasi standar, dan jumlah datanya saja.

- Pembuatan fungsi untuk peringkasan data berdasarkan class

```
def summarize_by_class(dataset):
    separated = separate_by_class(dataset)
    summaries= dict()
    for class_value, rows in separated.items():
        summaries[class_value] = summarize_dataset(rows)
    return summaries
```

Fungsi summarize_by_class menghitung statistik (rata-rata, deviasi standar, jumlah data) untuk setiap kolom berdasarkan kelasnya.

- Pembuatan fungsi untuk perhitungan probabilitas

```
def calculate_probability(x, mean, stdev):
    if stdev == 0:
        if x == mean:
            return 1.0
        else:
            return 1e-300
    else:
        exponent = exp(-((x - mean) ** 2 / (2 * stdev ** 2)))
        return (1 / (sqrt(2 * pi) * stdev)) * exponent
```

Fungsi calculate_probability menghitung distribusi probabilitas Gaussian untuk suatu nilai x.

- Pembuatan fungsi untuk perhitungan probabilitas kelas

```python
def calculate_class_probabilities(summaries, row):
    total_rows = sum([summaries [label][0][2] for label in summaries])
    probabilities = dict()
    for class_value, class_summaries in summaries.items():
        probabilities[class_value] = summaries[class_value][0][2]/float(total_rows)
        for i in range(len(class_summaries)):
            mean, stdev, _ = class_summaries[i]
            probabilities[class_value] *= calculate_probability(row[i], mean, stdev)
    return probabilities
```

Fungsi calculate_class_probabilities menghitung probabilitas prediksi untuk setiap kelas berdasarkan distribusi Gaussian.

- Pembuatan fungsi untuk melakukan prediksi

```python
def predict(summaries, row):
    probabilities = calculate_class_probabilities (summaries, row)
    best_label, best_prob = None, -1
    for class_value, probability in probabilities.items():
        if best_label is None or probability > best_prob:
            best_prob = probability
            best_label = class_value
    return best_label
```

Fungsi predict bertujuan untuk melakukan prediksi kelas untuk sebuah baris data tertentu menggunakan model Naive Bayes yang telah dilatih.

1.3. Mendefinisikan dataset

Dataset yang diberikan memiliki struktur berikut:

Kategori Produk	Harga Produk	Rating Produk	Jumlah Ulasan	Promosi	Tingkat Penjualan
Elektronik	Rp.10,000,00	4.5	100	Tidak	Tinggi
Fashion	Rp.500,000	4.0	50	Ya	Rendah
Elektronik	Rp.1,500,000	4.8	200	Tidak	Tinggi
Olahraga	Rp.800,000	4.2	80	Tidak	Rendah
Elektronik	Rp.1,200,000	4.7	150	Ya	Tinggi
Fashion	Rp.700,000	4.3	90	Tidak	Rendah
Makanan	Rp.200,000	3.5	30	Tidak	Rendah
Elektronik	Rp.900,000	4.6	120	Tidak	Tinggi
Olahraga	Rp.600,000	4.0	70	Ya	Rendah
Fashion	Rp.850,000	4.5	110	Tidak	Tinggi
Elektronik	Rp.1,300,000	4.2	180	Ya	Tinggi
Makanan	Rp.300,000	3.8	40	Tidak	Rendah
Olahraga	Rp.700,000	4.6	130	Tidak	Tinggi
Fashion	Rp.550,000	4.1	60	Ya	Rendah
Elektronik	Rp.1,100,000	4.9	220	Tidak	Tinggi
Makanan	Rp.250,000	3.7	35	Ya	Rendah
Fashion	Rp.800,000	4.4	95	Tidak	Tinggi
Elektronik	Rp.950,000	4.3	170	Tidak	Tinggi
Olahraga	Rp.650,000	4.5	100	Ya	Tinggi
Fashion	Rp.900,000	4.2	75	Ya	Rendah

Namun, karena dataset masih disajikan dengan terdapat nilai yang masih dalam berbentuk string, untuk memudahkan dalam perhitungan probabilitas, nilai yang berbentuk string seperti 'Kategori Produk' & 'Sedang Promo' perlu diubah ke dalam angka, selain itu juga perlu dilakukan perubahan format penyajian data, penyajian data yang saya pilih disini adalah dengan menggunakan Array of Arrays. Seperti berikut:

```
dataset = [
    [1, 0, 0, 0, 10000000, 4.05, 100, 0, 'Tinggi'],
    [0, 1, 0, 0, 500000, 4.0, 50, 1, 'Rendah'],
    [1, 0, 0, 0, 1500000, 4.08, 200, 0, 'Tinggi'],
    [0, 0, 1, 0, 800000, 4.02, 80, 0, 'Rendah'],
    [1, 0, 0, 0, 1200000, 4.07, 150, 1, 'Tinggi'],
    [0, 1, 0, 0, 700000, 4.03, 90, 0, 'Rendah'],
    [0, 0, 0, 1, 200000, 3.05, 30, 0, 'Rendah'],
    [1, 0, 0, 0, 900000, 4.06, 120, 0, 'Tinggi'],
    [0, 0, 1, 0, 600000, 4.0, 70, 1, 'Rendah'],
    [0, 1, 0, 0, 850000, 4.05, 110, 0, 'Tinggi'],
    [1, 0, 0, 0, 1300000, 4.02, 180, 1, 'Tinggi'],
    [0, 0, 0, 1, 300000, 3.08, 40, 0, 'Rendah'],
    [0, 0, 1, 0, 700000, 4.06, 130, 0, 'Tinggi'],
    [0, 1, 0, 0, 550000, 4.01, 60, 1, 'Rendah'],
    [1, 0, 0, 0, 1100000, 4.09, 220, 0, 'Tinggi'],
    [0, 0, 0, 1, 250000, 3.07, 35, 1, 'Rendah'],
    [0, 1, 0, 0, 800000, 4.04, 95, 0, 'Tinggi'],
    [1, 0, 0, 0, 950000, 4.03, 170, 0, 'Tinggi'],
    [0, 0, 1, 0, 650000, 4.05, 100, 1, 'Tinggi'],
    [0, 1, 0, 0, 900000, 4.02, 75, 1, 'Rendah']
]
```

Format urutan elemen dari setiap data produk di dalam dataset tersebut yaitu: [<elektronik>, <fashion>, <olahraga>, <makanan>, <harga>, <rating>, <jumlah ulasan>, <sedang promo>, <tingkat penjualan>]

Indeks kategori produk seperti 'Elektronik' (Indeks 0), 'Fashion' (Indeks 1), 'Olahraga' (Indeks 2), 'Makanan' (Indeks 3) memiliki arti bahwa suatu produk termasuk bagian dari ketegori tersebut jika bernilai 1. Misal:

[1, 0, 0, 0, 10000000, 4.05, 100, 0, 'Tinggi']

Pada contoh array di atas, diantara indeks 0 sampai indeks 3, yang bernilai 1 adalah indeks ke-1, maka produk yang direpresentasikan oleh array tersebut termasuk ke dalam kategori produk 'Elektronik'. Sisa nya yang bernilai 0 berarti produk tersebut bukan merupakan bagian dari ketegori-kategori tersebut.

1.4. Melakukan pemisahan data berdasarkan kelas
Dari dataset yang telah didefinisikan sebelumnya, kita bisa membagi data-data produk berdasarkan nilai dari kolom/indeks terakhir (Tingkat Penjualan: Tinggi/Rendah). Untuk melakukannya, seperti yang terlihat pada gambar di bawah, kita dapat menggunakan fungsi yang sudah didefinisikan sebelumnya, yaitu separate_by_class().

```
separated = separate_by_class(dataset)
for label in separated:
  print(label)
  for row in separated [label]:
    print(row)
```

Dan tampilkan data nya menggunakan perulangan akan menghasilkan output seperti berikut:

```
Tinggi
[1, 0, 0, 0, 10000000, 4.05, 100, 0, 'Tinggi']
[1, 0, 0, 0, 1500000, 4.08, 200, 0, 'Tinggi']
[1, 0, 0, 0, 1200000, 4.07, 150, 1, 'Tinggi']
[1, 0, 0, 0, 900000, 4.06, 120, 0, 'Tinggi']
[0, 1, 0, 0, 850000, 4.05, 110, 0, 'Tinggi']
[1, 0, 0, 0, 1300000, 4.02, 180, 1, 'Tinggi']
[0, 0, 1, 0, 700000, 4.06, 130, 0, 'Tinggi']
[1, 0, 0, 0, 1100000, 4.09, 220, 0, 'Tinggi']
[0, 1, 0, 0, 800000, 4.04, 95, 0, 'Tinggi']
[1, 0, 0, 0, 950000, 4.03, 170, 0, 'Tinggi']
[0, 0, 1, 0, 650000, 4.05, 100, 1, 'Tinggi']
Rendah
[0, 1, 0, 0, 500000, 4.0, 50, 1, 'Rendah']
[0, 0, 1, 0, 800000, 4.02, 80, 0, 'Rendah']
[0, 1, 0, 0, 700000, 4.03, 90, 0, 'Rendah']
[0, 0, 0, 1, 200000, 3.05, 30, 0, 'Rendah']
[0, 0, 1, 0, 600000, 4.0, 70, 1, 'Rendah']
[0, 0, 0, 1, 300000, 3.08, 40, 0, 'Rendah']
[0, 1, 0, 0, 550000, 4.01, 60, 1, 'Rendah']
[0, 0, 0, 1, 250000, 3.07, 35, 1, 'Rendah']
[0, 1, 0, 0, 900000, 4.02, 75, 1, 'Rendah']
```

1.5. Mengkonversi kolom kelas menjadi angka (integer)
Nanti, kita akan meringkas dataset, baik itu menjadi mean & standar deviasi ataupun berdasarkan kelas. Namun, karena kita sudah mendefinisikan fungsi pembantu seperti summarize_dataset() dan summarize_by_class(), yang dimana elemen array dari dataset yang kita inputkan ke fungsi-fungsi tersebut harus bernilai angka semua (integer), oleh karena itu, kita perlu mengubah sisa data yang masih dalam bentuk string, yaitu kolom/indeks terakhir (Tingkat Penjualan).

Untuk melakukannya, kita bisa menggunakan fungsi str_column_to_int() yang sudah saya definisikan sebelumnya.

```
# Mengkonversi kolom kelas menjadi integer
str_column_to_int(dataset, len(dataset[0])-1)
```

Dari penggunaan fungsi tersebut, menghasilkan hasil seperti berikut:

```
{0: 'Rendah', 1: 'Tinggi'}
```

Tingkat penjualan 'Rendah' direpresentasikan dengan nilai 0, sedangkan tingkat penjualan 'Tinggi' direpresentasikan dengan nilai 1.

1.6. Meringkas dataset berdasarkan kelas
Kita perlu meringkas dataset berdasarkan kelas (Setiap data pada suatu fitur diringkas menjadi mean & standar deviasi, danf fitur tersebut dibagi lagi berdasarkan kelas) untuk nanti digunakan dalam perhitungan probabilitas suatu produk (produk sample) akan termasuk ke golongan Tingkat Penjualan Tinggi/Rendah berdasarkan data mengenai setiap fiturnya. Untuk melakukannya, kita bisa menggunakan fungsi summarize_by_class().

```
model = summarize_by_class(dataset)

for label in model:
    print(label)
    for row in model [label]:
      print(row)
```

Berikut hasil dari peringkasan dataset berdasarkan kelas:

```
1
(0.6363636363636364, 0.504524979109513, 11)
(0.18181818181818182, 0.4045199174779452, 11)
(0.18181818181818182, 0.40451991747794525, 11)
(0.0, 0.0, 11)
(1813636.3636363635, 2727553.0159000494, 11)
(4.054545454545454, 0.02067057636527653, 11)
(143.1818181818182, 43.719144963775726, 11)
(0.2727272727272727, 0.46709936649691375, 11)
0
(0.0, 0.0, 9)
(0.4444444444444444, 0.5270462766947299, 9)
(0.2222222222222222, 0.44095855184409843, 9)
(0.3333333333333333, 0.5, 9)
(533333.3333333334, 246221.4450449026, 9)
(3.697777777777778, 0.47349175752534955, 9)
(58.888888888888886, 21.327473153449546, 9)
(0.5555555555555556, 0.5270462766947299, 9)
```

1.7. Mendefinisikan produk yang ingin dihitung probabilitasnya
Di soal, kita diminta untuk mencari probabilitas suatu produk akan termasuk ke golongan Tingkat Penjualan Tinggi/Rendah. Dan produk tersebut adalah produk dengan kategori elektronik, harga 1 juta, rating 4.5, jumlah ulasan 100, dan tidak sedang dalam promosi, yang jika direpresentasikan melalui format penyajian data pada dataset sebelumnya, adalah sebagai berikut:

```
sample = [1, 0, 0, 0, 1000000, 4.50, 100, 0]
```

1.8. Menghitung probabilitas produk sample termasuk ke dalam golongan tingkat penjualan tinggi/rendah berdasarkan data mengenai setiap fiturnya. (Ketentuan 1)

Berikut kode untuk menghitung probabilitas produk sample akan termasuk ke dalam golongan tingkat penjualan tinggi/rendah berdasarkan

data mengenai setiap fiturnya. Dengan melakukan looping di setiap data fitur pada produk sample

```
print('\nMenghitung probabilitas produk termasuk ke golongan Tingkat Penjualan Tinggi/Rendah berdasarkan data mengenai setiap fiturnya')
# 1. Menghitung probabilitas setiap fitur dalam setiap kategori (tingkat penjualan tinggi/rendah)
for index, fitur in enumerate(sample):
    if (index == 0):
      print("\nProbabilitas produk merupakan Elektronik:")
    elif(index == 1):
      print("\nProbabilitas produk merupakan Fashion:")
    elif(index == 2):
      print("\nProbabilitas produk merupakan Olahraga:")
    elif(index == 3):
      print("\nProbabilitas produk merupakan Makanan:")
    elif(index == 4):
      print("\nProbabilitas fitur Harga Produk:")
    elif(index == 5):
      print("\nProbabilitas fitur Rating Produk:")
    elif(index == 6):
      print("\nProbabilitas fitur Jumlah Ulasan:")
    elif(index == 7):
      print("\nProbabilitas fitur Sedang Promo:")

    for label in model:
      print('\n' + str(label))
      for row in model [label]:
        mean, stdev, col = row

        print(calculate_probability(fitur, mean, stdev))
```

Output:
Menghitung probabilitas produk termasuk ke golongan Tingkat Penjualan Tinggi/Rendah berdasarkan data mengenai setiap fiturnya

Probabilitas produk merupakan Elektronik:

1
0.6098508468118082
0.1275382735516702
0.12753827355167027
1e-300
1.1725429702349774e-07
0.0
4.6085640194678455e-05
0.2541459736568529

0
1e-300
0.4342967746448798
0.19096030608845055
0.3280201493519872
1.551536545329503e-07
7.522556602758265e-08
0.00047007782719667764
0.530451278411322

Probabilitas produk merupakan Fashion:

1
0.3569158815594213
0.8914603393062909
0.8914603393062908

1.0
1.1725426843887874e-07
0.0
4.2770652658331946e-05
0.7202335022004414

0
1.0
0.530451278411322
0.7968265396375414
0.6388960110447045
1.5515228961461312e-07
4.80666932068047e-14
0.00041344804454897363
0.4342967746448798

Probabilitas produk merupakan Olahraga:

1
0.3569158815594213
0.8914603393062909
0.8914603393062908
1.0
1.1725426843887874e-07
0.0
4.2770652658331946e-05
0.7202335022004414

0
1.0
0.530451278411322
0.7968265396375414
0.6388960110447045
1.5515228961461312e-07
4.80666932068047e-14
0.00041344804454897363
0.4342967746448798

Probabilitas produk merupakan Makanan:

1
0.3569158815594213
0.8914603393062909
0.8914603393062908
1.0
1.1725426843887874e-07
0.0
4.2770652658331946e-05

0.7202335022004414

0
1.0
0.530451278411322
0.7968265396375414
0.6388960110447045
1.5515228961461312e-07
4.80666932068047e-14
0.00041344804454897363
0.4342967746448798

Probabilitas fitur Harga Produk:

1
0.0
0.0
0.0
1e-300
1.3989884509741612e-07
0.0
0.0
0.0

0
1e-300
0.0
0.0
0.0
2.6887197230105743e-07
0.0
0.0
0.0

Probabilitas fitur Rating Produk:

1
1.4572707599022034e-13
1.77665608678111e-25
1.7766560867811348e-25
1e-300
1.1725439706959499e-07
2.7554637925582663e-100
5.9599882865038736e-05
1.4008817544272402e-18

0
1e-300

1.0508286494691626e-13
3.3154629229321664e-21
6.6415668898544405e-16
1.5515843182153646e-07
0.20057022948362446
0.0007240552751941271
5.204788373304043e-13

Probabilitas fitur Jumlah Ulasan:

1
0.0
0.0
0.0
1e-300
1.1725712685725392e-07
0.0
0.005602682693501026
0.0

0
1e-300
0.0
0.0
0.0
1.5528882822364356e-07
0.0
0.002918223535420438
0.0

Probabilitas fitur Sedang Promo:

1
0.3569158815594213
0.8914603393062909
0.8914603393062908
1.0
1.1725426843887874e-07
0.0
4.2770652658331946e-05
0.7202335022004414

0
1.0
0.530451278411322
0.7968265396375414
0.6388960110447045
1.5515228961461312e-07

4.80666932068047e-14
0.00041344804454897363
0.4342967746448798

1.9. Menghitung probabilitas tingkat penjualan (Ketentuan 2)
Pada ketentuan 2, diminta untuk menghitung probabilitas tingkat penjualan (Tinggi atau Rendah) dari produk sample.

```
print('\n')
# 2. Menghitung probabilitas produk sample akan masuk ke kategori tingkat penjualan tinggi/rendah
probabilitas = calculate_class_probabilities(model, sample)
print(probabilitas)
```

Output:
{1: 4.146374809985276e-110, 0: 8.305564312723e-312}

Berdasarkan hasil yang telah ditampilkan di atas, kita bisa tahu, kedua kelas (Tingkat penjualan: Tinggi/Rendah) sama-sama memiliki probabilitas yang kecil Itu terjadi karena banyak data fitur yang memiliki standar deviasi 0, sehingga di fungsi calculate_probability(), sudah diantisipasi jika standar deviasi bernilai 0, maka akan melakukan return 1^-300 (Hampir mendekati 0) untuk menghindari terjadi nya error dalam perhitungan, karena jika standar deviasi bernilai 0, akan terjadi pembagian bilangan dengan 0.

Meskipun begitu, jika dibandingkan property 1 (Tingkat penjualan: Tinggi) memiliki probabilitas yang lebih besar daripada property 0 (Tingkat penjualan: Rendah). Karena jangkauan minus (-) nya lebih jauh property 0 dibandingkan property 1.

1.10. Menentukan apakah produk dengan kategori elektronik, harga 1 juta, rating 4.5, jumlah ulasan 100, dan tidak sedang dalam promosi akan memiliki tingkat penjualan tinggi atau rendah. (Ketentuan 3)

Untuk melakukannya, kita bisa cukup memanfaatkan fungsi predict() yang sudah didefinisikan sebelumnya.

```
label = predict(model, sample)
print('Data=%s, Predicted: %s' % (row, label))
```

Yang hasilnya sebagai berikut:
Data=(0.5555555555555556, 0.5270462766947299, 9), Predicted: 1

Dari output di atas, kita bisa tahu bahwa produk dengan kategori elektronik, harga 1 juta, rating 4.5, jumlah ulasan 100, dan tidak sedang dalam promosi akan diprediksi termasuk ke golongan **Tingkat Penjualan 'Tinggi'**. Itu artinya, hasilnya selaras dengan perbandingan nilai probabilitas antara kelas tingkat penjualan rendah dengan tingkat penjualan tinggi yang dilakukan sebelumnya, yang dimana nilai

probabilitas tingkat penjualan Tinggi lebih besar daripada tingkat penjualan Rendah.

2. Kita dapat menggunakan pustaka Scikit-fuzzy dan mengontrolnya dengan menggunakan kontrol dari Scikit-fuzzy untuk membuat variabel penampung input dan output. Berikut adalah langkah-langkah yang perlu dilakukan:

2.1. Install *library scikit-fuzzy*

```
!pip install scikit-fuzzy > /dev/null 2>&1
```

2.2. Melakukan import library eksternal yang diperlukan

```
import pandas as pd
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

2.3. Membuat variabel input dan output menggunakan Antecedent dan Consequent

- Input

```
beban_kerja = ctrl.Antecedent(np.arange(0, 101, 1), 'Beban Kerja (%)')
suhu_lingkungan = ctrl.Antecedent(np.arange(20, 41, 1), 'Suhu Lingkungan (C)')
```

- Output

```
kecepatan_mesin = ctrl.Consequent(np.arange(0, 3001, 1), 'Kecepatan Mesin (RPM)')
```

2.4. Membuat fungsi keanggotaan untuk variabel kelembaban dan waktu_siram menggunakan trimf

- Variabel beban_kerja

```
beban_kerja['ringan'] = fuzz.trimf(beban_kerja.universe, [0, 0, 50])
beban_kerja['sedang'] = fuzz.trimf(beban_kerja.universe, [0, 50, 100])
beban_kerja['berat'] = fuzz.trimf(beban_kerja.universe, [50, 100, 100])
```

- Variabel suhu_lingkungan

```
suhu_lingkungan['rendah'] = fuzz.trimf(suhu_lingkungan.universe, [20, 20, 30])
suhu_lingkungan['sedang'] = fuzz.trimf(suhu_lingkungan.universe, [20, 30, 40])
suhu_lingkungan['tinggi'] = fuzz.trimf(suhu_lingkungan.universe, [30, 40, 40])
```

- Variabel kecepatan_mesin

```
kecepatan_mesin['lambat'] = fuzz.trimf(kecepatan_mesin.universe, [0, 0, 1500])
kecepatan_mesin['sedang'] = fuzz.trimf(kecepatan_mesin.universe, [0, 1500, 3000])
kecepatan_mesin['cepat'] = fuzz.trimf(kecepatan_mesin.universe, [1500, 3000, 3000])
```

2.5. Membuat aturan fuzzy dengan ctrl.Rule

```
rule1 = ctrl.Rule(beban_kerja['ringan'] & suhu_lingkungan['rendah'], kecepatan_mesin['cepat'])
rule2 = ctrl.Rule(beban_kerja['sedang'] & suhu_lingkungan['sedang'], kecepatan_mesin['sedang'])
rule3 = ctrl.Rule(beban_kerja['berat'] & suhu_lingkungan['tinggi'], kecepatan_mesin['lambat'])
```

2.6. Membuat sistem kontrol fuzzy dengan ctrl.ControlSystem

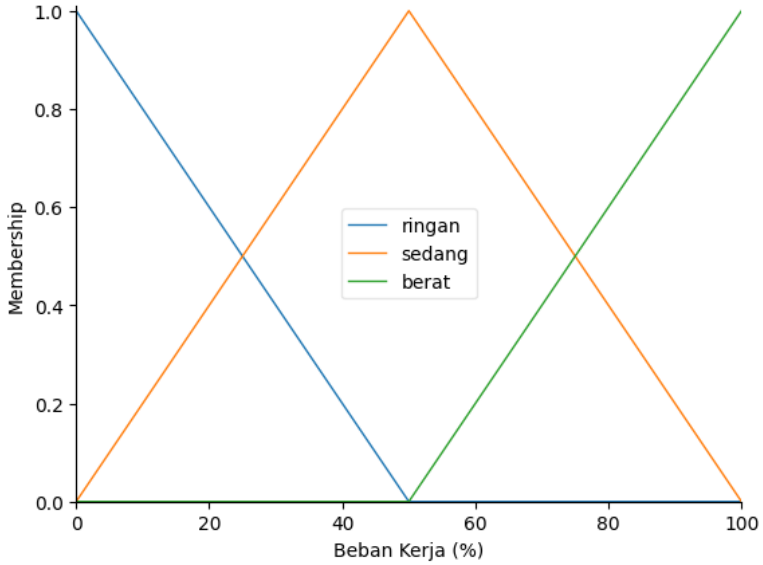
```
sistem_kontrol = ctrl.ControlSystem([rule1, rule2, rule3])
mesin_ctrl = ctrl.ControlSystemSimulation(sistem_kontrol)
```

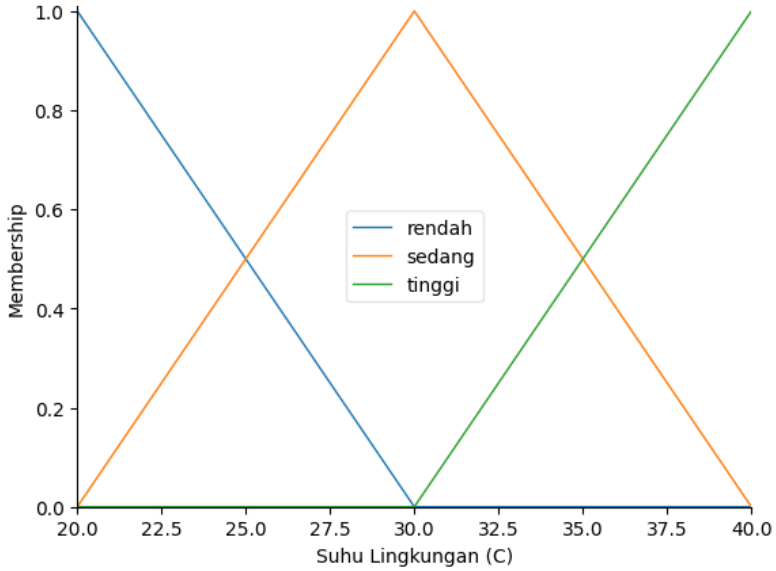
2.7. Menampilkan grafik fungsi keanggotaan

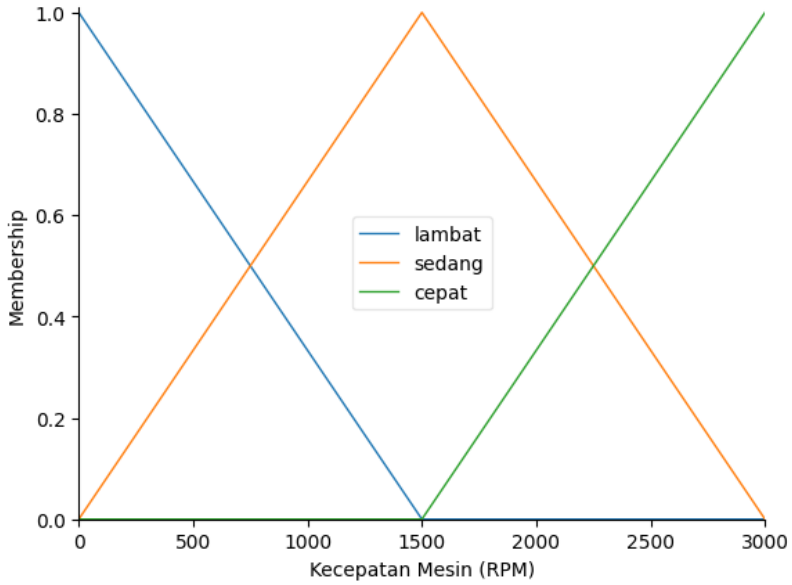
```
beban_kerja.view()
suhu_lingkungan.view()
kecepatan_mesin.view()

plt.show()
```

Output:






1.0
0.8
0.6
0.4
0.2
0.0
Membership
lambat
sedang
cepat
0
500
1000
1500
2000
2500
3000
Kecepatan Mesin (RPM)