LAPORAN PRAKTIKUM
KECERDASAN BUATAN
"STUDI KASUS"

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

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2023

DAFTAR ISI

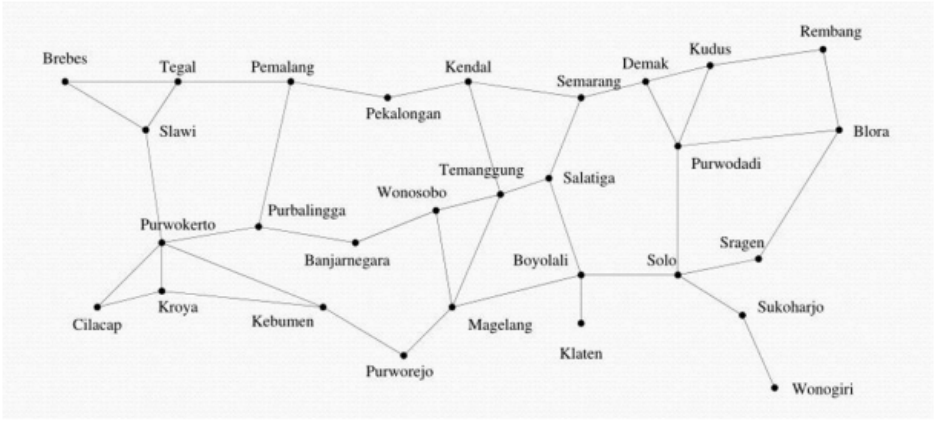
SOAL

Latihan: Uji Coba Menggunakan Berbagai Jenis Fungsi Keanggotaan (Membership Function)

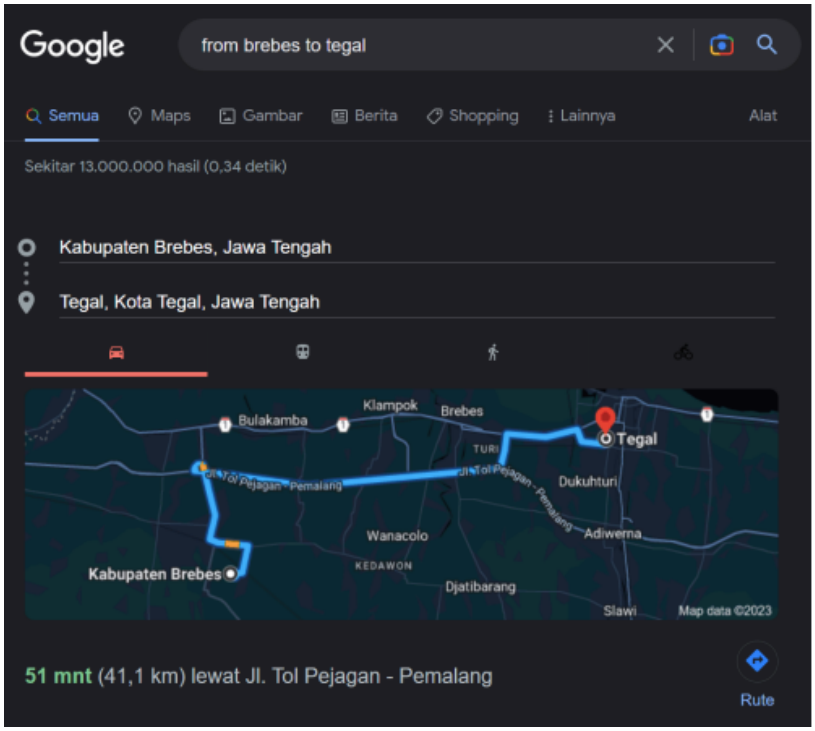
1. **Kasus 1 – Metode Pencarian**

   Andri adalah seorang karyawan yang bekerja di salah satu perusahaan di Kawasan Industri Wijayakusuma, kota Semarang. Pada saat libur hari raya idul fitri nanti, Andri berencana untuk mengunjungi keluarganya di Cilacap. Namun, ia tidak tahu rute perjalanan yang harus ia lalui untuk sampai tujuan. Andri hanya memiliki peta jawa tengah yang berbentuk graph berisi nama nama kota di jawa tengah. Namun, dia belum memiliki jarak masing masing kota tersebut. Bantulah andri untuk menentukan rute perjalanan terbaiknya!

PETA JAWA TENGAH

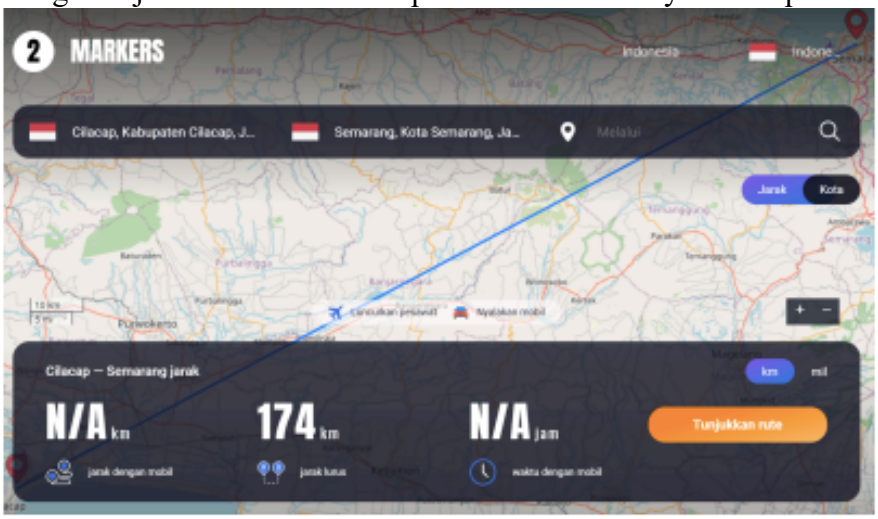


1. Carilah jarak masing masing kota menggunakan bantuan google! Kamu dapat mencarinya di google untuk jarak masing masing kota dengan
   menggunakan keyword : “From city1 to city2”. Example :



Didapatkan jarak antara kota 1 dan kota 2 adalah 41,1 km.

2. Carilah jarak garis lurus ke lokasi tujuan!
   Kamu dapat menggunakan website https://id.2markers.com/ masukan cilacap sebagai kota asal dan nama kota lainnya sebagai tujuan untuk mengukur jarak lurus dari cilacap ke kota kota lainnya. Example :



3. Gunakan Teknik Best First Search untuk menentukan rute perjalanan untuk Andri!

2. Kasus 1 – Ketidakpastian

Seorang penjual baju online ingin memprediksi apakah seorang pembeli akan membeli produknya atau tidak. Penjual tersebut memiliki dataset pembeli yang telah membeli dan tidak membeli produk. Dataset tersebut

terdiri dari variabel independen berupa usia dan penghasilan, serta variabel dependen berupa pembelian (1=ya, 0=tidak). Gunakan metode Naive Bayes untuk memprediksi apakah seseorang akan membeli produk dari toko online tersebut berdasarkan usia dan penghasilan.

3. **Kasus 1 – Fuzzy Logic**
   Seorang petani ingin mengetahui kapan waktu yang tepat untuk melakukan penyiraman tanaman jagung di lahan pertaniannya. Ia telah memasang sebuah sensor yang mengukur kelembaban tanah pada lahan tersebut, tetapi ia masih kesulitan menentukan waktu yang tepat untuk melakukan penyiraman.

   Dalam kasus ini, Fuzzy Logic dapat digunakan untuk membantu petani menentukan waktu yang tepat untuk melakukan penyiraman. Fuzzy Logic digunakan untuk mengukur kelembaban tanah pada lahan pertanian, dan menghasilkan keluaran berupa "kering", "lembab", atau "basah", yang kemudian digunakan untuk menentukan waktu penyiraman yang tepat.

   Untuk menjawab permasalahan di atas gunakan library Scikit-fuzzy lalu menggunakan control dari scikit-fuzzy untuk membuat nilai variabel penampung input, dengan range 0 sampai dengan 100 dengan increment sebesar 1. Artinya, himpunan dari variabel "Kelembaban" terdiri dari 101 elemen yang dimulai dari 0 dan berakhir pada 100. seperti berikut ini

   ctrl.Antecedent(np.arange(0, 101, 1), 'kelembapan')

   buat juga variabel untuk menampung nilai outputnya dengan range 0 sampai dengan 30 dengan increment sebesar 1. Artinya, himpunan dari variabel "waktu_siram" terdiri dari 31 elemen yang dimulai dari 0 dan berakhir pada 30, seperti berikut :

   ctrl.Consequent(np.arange(0, 31, 1), 'waktu_siram')

   Note :
   Nilai **kelembaban** dapat di set menjadi 3 kondisi, berdasarkan rentang nilai kelembaban
   0-100 yaitu: kering (0-30), lembab (31-60) dan basah (61-101).
   Nilai **waktu_siram** dapat di set menjadi 3 kondisi, berdasarkan rentang nilai waktu_siram
   0-30 yaitu: cepat (0-10), sedang (11-20) dan lambat (21-31)

   Gunakan urutan langkah-langkah berikut untuk menjawab studi kasus ini :
   1. Membuat variabel input dan output menggunakan Antecedent dan Consequent
   2. Membuat fungsi keanggotaan untuk variabel kelembaban dan waktu_siram

menggunakan trimf.
3. Membuat aturan fuzzy menggunakan ctrl.Rule
4. Membuat sistem kontrol fuzzy menggunakan ctrl.ControlSystem
5. Menampilkan grafik fungsi keanggotaan menggunakan view()

JAWABAN

1. Berikut langkah yang perlu dilakukan untuk melakukan pencarian rute menggunakan teknik Best First Search:
   1.1. Inisialisasi variable graph untuk menampung informasi mengenai jarak antar kota.

   Data mengenai jarak antar kota yaitu:
   - Cilacap = Kroya(29,1 km), Purwokerto (49,6 km)
   - Kroya = Cilacap(29,1 km), Purwokerto (32, 2 km), Kebumen (53, 6 km)
   - Kebumen = Kroya(53,6 km), Purworejo (42 km), Purwokerto (71,2 km)
   - Purworejo = Kebumen(42 km), Magelang (43 km)
   - Magelang = Purworejo(43 km), Wonosobo (60,4 km), Temanggung (25,3 km), Boyolali (64,3 km)
   - Boyolali = Magelang(64,3 km), Klaten (34,9 km), Salatiga (24,4 km), Solo (36,6 km)
   - Klaten = Boyolali(34,9 km
   - Solo = Boyolali(36,6 km), Purwodadi (67,4 km), Sragen (34,8 km), Sukoharjo (14,2 km)
   - Sukoharjo = Solo(14,2 km), Wonogiri (40,7 km)
   - Wonogiri = Sukoharjo(40,7 km)
   - Sragen = Solo(34,8 km), Blora (83,7 km
   - Blora = Purwodadi(65,3 km), Rembang(36,1 km), Sragen (83,7 km)
   - Rembang = Blora(36,1 km), Kudus(60,9 km)
   - Kudus = Rembang(60,9 km), Purwodadi (47,9 km), Demak (42,5 km)
   - Purwodadi = Demak(39 km), Kudus (47,9 km), Blora (65,3 km), Solo (67,4 km)
   - Demak = Semarang(31,4 km), Purwodadi (39 km), Kudus (42,5 km)
   - Semarang = Demak(31,4 km), Salatiga(51,5 km), Kendal (31,7 km)
   - Salatiga = Semarang(51,5 km), Boyolali(24,4 km), Temanggung(55,6 km)
   - Temanggung = Kendal(71,2 km), Salatiga(55,6 km), Magelang(25,3 km), Wonosobo (40 km)
   - Wonosobo = Temanggung(40 km), Magelang(60,4 km), Banjarnegara(42,5 km)
   - Banjarnegara = Wonosobo(42,5 km), Purbalingga(33,4 km)
   - Purbalingga= Banjarnegara(33,4 km), Purwokerto(16,8 km),
   - Purwokerto = Purbalingga(16,8 km), Kebumen(71,2 km), Kroya(32,2 km), Cilacap (49,6 km), Slawi (89 km)
   - Slawi = Purwokerto(89 km), Brebes(25,3 km), Tegal(12 km)
   - Brebes = Slawi(25,3 km), Tegal(40,9 km)
   - Tegal = Brebes(40,9 km), Slawi(12 km)

- Pemalang = Tegal(32,4 km), Purbalingga(68,8 km), Pekalongan(47,5 km)
- Pekalongan = Pemalang(47,5 km), Kendal(74,7 km)
- Kendal = Pekalongan(74,7 km), Temanggung(71,2 km), Semarang (31,7 km)

```
graph = {
    'Brebes' : [('Tegal', 40.9), ('Slawi', 25.3)],
    'Tegal' : [('Brebes', 40.9), ('Slawi', 1.3), ('Pemalang', 32.4)],
    'Pemalang' : [('Tegal', 32.7), ('Purbalingga', 68.8), ('Pekalongan', 47.2)],
    'Kendal' : [('Pekalongan', 76.7), ('Temanggung', 71.2), ('Semarang', 31.7)],
    'Semarang' : [('Kendal', 31.7), ('Salatiga', 51.5), ('Demak', 31.4)],
    'Demak' : [('Semarang', 31.4), ('Purwodadi', 39), ('Kudus', 62.1)],
    'Kudus' : [('Demak', 62.1), ('Purwodadi', 43.5), ('Rembang', 60.9)],
    'Rembang' : [('Kudus', 60.9), ('Blora', 36.1)],
    'Slawi' : [('Brebes', 25.3), ('Tegal', 1.3), ('Purwokerto', 89)],
    'Pekalongan' : [('Pemalang', 47.2), ('Kendal', 76.7)],
    'Purwokerto' : [('Slawi', 89), ('Purbalingga', 20.2), ('Kebumen', 71.2), ('Kroya', 31.1), ('Cilacap', 49.6)],
    'Purbalingga' : [('Purwokerto', 20.2), ('Pemalang', 68.8), ('Banjarnegara', 33.4)],
    'Wonosobo' : [('Banjarnegara', 43.2), ('Magelang', 60.4), ('Temanggung', 40)],
    'Temanggung' : [('Wonosobo', 40), ('Magelang', 28.2), ('Salatiga', 55)],
    'Salatiga' : [('Temanggung', 55), ('Semarang', 51.5), ('Boyolali', 24.4)],
    'Purwodadi' : [('Demak', 39), ('Kudus', 43.5), ('Blora', 65.3), ('Solo', 67.4)],
    'Blora' : [('Rembang', 36.1), ('Purwodadi', 65.3), ('Sragen', 83.4)],
    'Banjarnegara' : [('Purbalingga', 33.4), ('Wonosobo', 43.2)],
    'Cilacap' : [('Purwokerto', 49.6), ('Kroya', 47.6)],
    'Kroya' : [('Cilacap', 47.6), ('Purwokerto', 31.1), ('Kebumen', 53.6)],
    'Kebumen' : [('Kroya', 53.6), ('Purwokerto', 71.2), ('Purworejo', 42)],
    'Magelang' : [('Purworejo', 43.9), ('Wonosobo', 60.4), ('Temanggung', 28.2), ('Boyolali', 62.4)],
    'Boyolali' : [('Magelang', 62.4), ('Salatiga', 24.4), ('Solo', 36.6), ('Klaten', 34.9)],
    'Solo' : [('Boyolali', 36.6), ('Purwodadi', 67.4), ('Sragen', 34.8), ('Sukoharjo', 12.3)],
    'Sragen' : [('Solo', 34.8), ('Blora', 83.4)],
    'Purworejo' : [('Kebumen', 42), ('Magelang', 43.9)],
    'Klaten' : [('Boyolali', 34.9)],
    'Sukoharjo' : [('Solo', 12.3), ('Wonogiri', 21.8)],
    'Wonogiri' : [('Sukoharjo', 21.8)],
}
```

1.2. Inisialisasi variable H_table untuk menampung data mengenai jarak garis lurus dari setiap kota ke kota tujuan.

Berikut data mengenai jarak garis lurus dari setiap kota ke kota cilacap :

- Brebes: 94 KM
- Tegal: 92 KM
- Pemalang: 95 KM
- Kendal: 156 KM
- Semarang: 174 KM
- Demak: 199 KM
- Kudus: 224 KM
- Rembang: 278 KM
- Slawi: 80 KM
- Pekalongan: 115 KM
- Purwokerto: 39 KM
- Purbalingga: 51 KM
- Wonosobo: 105 KM
- Temanggung: 134 KM
- Salatiga: 169 KM
- Purwodadi: 219 KM
- Blora: 276 KM
- Banjarnegara: 65 KM

- Cilacap: 0 KM
- Kroya: 23 KM
- Kebumen: 71 KM
- Magelang: 134 KM
- Boyolali: 174 KM
- Solo: 200 KM
- Sragen: 223 KM
- Purworejo: 109 KM
- Klaten: 181 KM
- Sukoharjo: 197 KM
- Wonogiri: 211 KM

```
H_table = {
   'Brebes': 94,
'Tegal': 92,
'Pemalang': 95,
'Kendal': 156,
'Semarang': 174,
'Demak': 199,
'Kudus': 224,
'Rembang': 278,
'Slawi': 80,
'Pekalongan': 115,
'Purwokerto': 39,
'Purbalingga': 51,
'Wonosobo': 105,
'Temanggung': 134,
'Salatiga': 169,
'Purwodadi': 219,
'Blora': 276,
'Banjarnegara': 65,
'Cilacap': 0,
'Kroya': 23,
'Kebumen': 71,
'Magelang': 134,
'Boyolali': 174,
'Solo': 200,
'Sragen': 223,
'Purworejo': 109,
'Klaten': 181,
'Sukoharjo': 197,
'Wonogiri': 211,
}
```

1.3. Pendefinisian fungsi yang dibutuhkan dalam pencarian rute

```
def path_h_cost(path):
  g_cost = 0
  for (node, cost) in path:
    g_cost += cost
    last_node = path[-1][0]
    h_cost = H_table[last_node]
    f_cost = g_cost + h_cost
    return h_cost, last_node

def Greedy_best_search(graph, start, goal):
  visited =[]
  queue = [[(start,0)]]

  while queue:
    queue.sort(key=path_h_cost)
    path = queue.pop(0)
```

```
    node = path[-1][0]

    if node in visited:
      continue
    visited.append(node)
    if node == goal:
      return path
    else:
      adjacent_nodes = graph.get(node, [])
      for (node2, cost) in adjacent_nodes:
        new_path = path.copy()
        new_path.append((node2, cost))
        queue.append(new_path)
```

1.4. Memanggil fungsi Greedy_best_search dan menampilkan hasil nya.

```
solution = Greedy_best_search(graph, 'Semarang', 'Cilacap')
print ('Rute menggunakan teknik Greedy Best First Search adalah ', solution)
```

Output:
Rute menggunakan teknik Greedy Best First Search adalah [('Semarang', 0), ('Kendal', 31.7), ('Pekalongan', 76.7), ('Pemalang', 47.2), ('Purbalingga', 68.8), ('Purwokerto', 20.2), ('Cilacap', 49.6)]

2. Berikut langkah yang dilakukan untuk dapat memprediksi apakah seseorang akan membeli produk dari toko online berdasarkan usia dan penghasilan menggunakan algoritma Naive Bayes

2.1. Melakukan import data pembeli

pembeli.csv

pembeli.csv ×

1 to 10 of 99 entries Filter

44	6089827	1
24	19920954	1
43	18916524	0
60	17686528	1
34	17421279	0
54	17775195	0
39	3007140	1
49	10884915	1
46	16906403	1
25	15208583	0
20	18375111	0

Show 10 per page

1 2 3 4 5 6 7 8 9 10

Ada tiga kolom dalam tabel tersebut, dengan kolom pertama menunjukkan usia pelanggan, kolom kedua menunjukkan pendapatan mereka, dan kolom ketiga memuat informasi apakah mereka telah membeli produk dari penjual baju online tersebut (1) atau tidak (0).

2.2. Pendefinisian fungsi yang dibutuhkan dalam algoritma Naive Bayes, yang diantaranya:

- **load_csv**: membaca file CSV dan mengembalikan dataset dalam bentuk daftar.

```
def load_csv(filename):
  dataset = list()
  with open(filename, 'r') as file:
    csv_reader = reader(file)
    for row in csv_reader:
      if not row:
        continue
      dataset.append(row)
  return dataset
```

- **str_column_to_float**: Untuk mengonversi kolom dataset dari string menjadi float.

```
def str_column_to_float(dataset, column):
  for row in dataset:
    row[column] = float(row[column].strip())
```

- **str_column_to_int**: mengonversi kolom dataset dari string menjadi integer dan membuat kamus lookup untuk nilai kelas.

```
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
```

- **cross_validation_split**: Untuk membagi dataset menjadi k bagian untuk melakukan cross-validation.

```
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
```

- **accuracy_metric**: Untuk menghitung akurasi prediksi dengan membandingkan nilai sebenarnya dengan nilai yang diprediksi.

```
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
```

- **evaluate_algorithm**: Untuk mengevaluasi algoritma dengan menggunakan validasi silang dan menghitung akurasi untuk setiap lipatan.

```
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
```

- **separate_by_class**: Untuk memisahkan dataset berdasarkan kelas.

```
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
```

- **mean**: Untuk menghitung rata-rata dari suatu daftar angka.

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

- **stdev**: Untuk menghitung standar deviasi dari sebuah daftar angka.

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

- **summarize_dataset**: Untuk menghitung rata-rata, standar deviasi, dan jumlah data untuk setiap kolom dalam dataset.

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

- **summarize_by_class**: Untuk memisahkan dataset berdasarkan kelas dan menghitung statistik ringkasan untuk setiap kelas.

```
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
```

- **calculate_probability**: Untuk menghitung probabilitas distribusi Gaussian pada suatu nilai.

```
def calculate_probability(x, mean, stdev):
    exponent = exp(-((x-mean) **2 / (2 * stdev**2 )))
    return (1/(sqrt(2*pi) * stdev)) * exponent
```

- **calculate_class_probabilities**: Untuk menghitung probabilitas memprediksi setiap kelas untuk sebuah baris.

```
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
```

- **predict**: Untuk memprediksi kelas untuk sebuah baris berdasarkan probabilitas terbesar.

```
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
```

- **naive_bayes**: Fungsi penerapan algoritma Naive Bayes untuk klasifikasi, menggunakan ringkasan statistik yang dihitung dari data pembeli.

```
def naive_bayes (train, test):
    summarize = summarize_by_class(train)
    predictions = list()
    for row in test:
        output = predict(summarize, row)
        predictions.append(output)
    return(predictions)
```

Sebelum mendefinisikan fungsi-fungsi di atas, perlu juga melakukan import library eksternal yang pastinya akan digunakan pada fungsi-fungsi tersebut.

```
# Naive Bayes On The Iris Dataset
from csv import reader
from random import seed
from random import randrange
from math import sqrt
from math import exp
from math import pi
```

2.3. Memanggil fungsi naivce_bayes dan menampilkan hasil prediksi
Setelah menyelesaikan semua fungsi yang diperlukan untuk mengimplementasikan algoritma Naïve-Bayes, langkah berikutnya adalah memasukkan file .csv ke dalam program untuk diolah dengan menggunakan perhitungan Naive-Bayes.

```
filename = 'pembeli.csv'
dataset = load_csv(filename)
for i in range(len(dataset[0])-1):
    str_column_to_float(dataset, i)
#convert class column to integers
str_column_to_int(dataset, len(dataset[0])-1)
```

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

```
model = summarize_by_class(dataset)
```

Lalu, kita akan mencoba mengevaluasi peluang bahwa salah satu individu dalam dataset akan melakukan pembelian produk dari toko online. Prediksi (1) akan menunjukkan bahwa individu tersebut akan membeli produk dari toko online, sementara prediksi (0) akan menunjukkan sebaliknya, bahwa individu tersebut tidak akan melakukan pembelian produk dari toko online.

```
model = summarize_by_class(dataset)
row = [39, 3007140]
label = predict(model, row)
print('Data=%s, Predicted: %s' % (row, label))
```

```
Data=[39, 3007140], Predicted: 1
```

Pada kode di atas, dapat terlihat, saya mencoba memprediksi pembeli jika usia nya 60 tahun dan pendapatan Rp.3.007.140,00 pada variable "row", apakah akan melakukan pembelian produk.

Dan hasilnya:
"Data=[39, 10884915], Predicted: 0."

Yang artinya pembeli dengan data tersebut, jika diprediksi menggunakan metode Naive Bayes tidak melakukan pembelian.

3. 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:

3.1. Install *library scikit-fuzzy*

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

3.2. Melakukan import library eksternal yang diperlukan

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

3.3. Membuat variabel input dan output menggunakan Antecedent dan Consequent

```
kelembaban = ctrl.Antecedent(np.arange(0, 101, 1), 'kelembapan')
waktu_siram = ctrl.Consequent(np.arange(0, 31, 1), 'waktu_siram')
```

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

- Variabel kelembaban

```
kelembaban['kering'] = fuzz.trimf(kelembaban.universe, [0, 0, 30])
kelembaban['lembab'] = fuzz.trimf(kelembaban.universe, [0, 30, 60])
kelembaban['basah'] = fuzz.trimf(kelembaban.universe, [38, 60, 100])
```

- Variabel waktu_siram

```
waktu_siram['cepat'] = fuzz.trimf(waktu_siram.universe, [0, 0, 10])
waktu_siram['sedang'] = fuzz.trimf(waktu_siram.universe, [0, 10, 20])
waktu_siram['lambat'] = fuzz.trimf(waktu_siram.universe, [10, 20, 30])
```

3.5. Membuat aturan fuzzy dengan ctrl.Rule

```
rule1 = ctrl.Rule(kelembaban['kering'], waktu_siram['cepat'])
rule2 = ctrl.Rule(kelembaban['lembab'], waktu_siram['sedang'])
rule3 = ctrl.Rule(kelembaban['basah'], waktu_siram['lambat'])
```

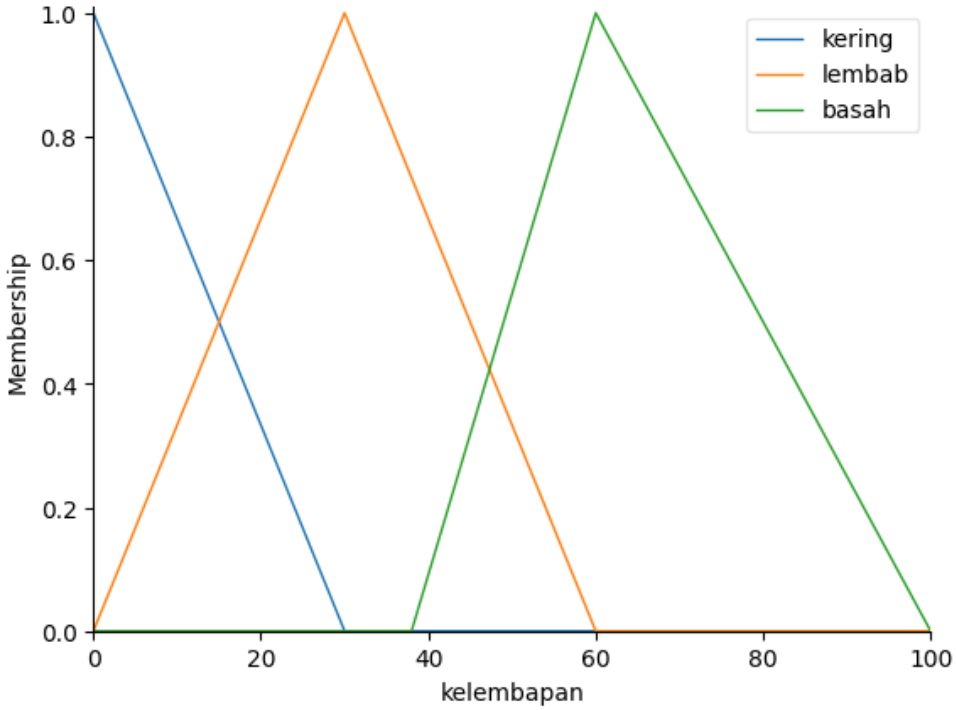
3.6. Membuat sistem kontrol fuzzy dengan ctrl.ControlSystem

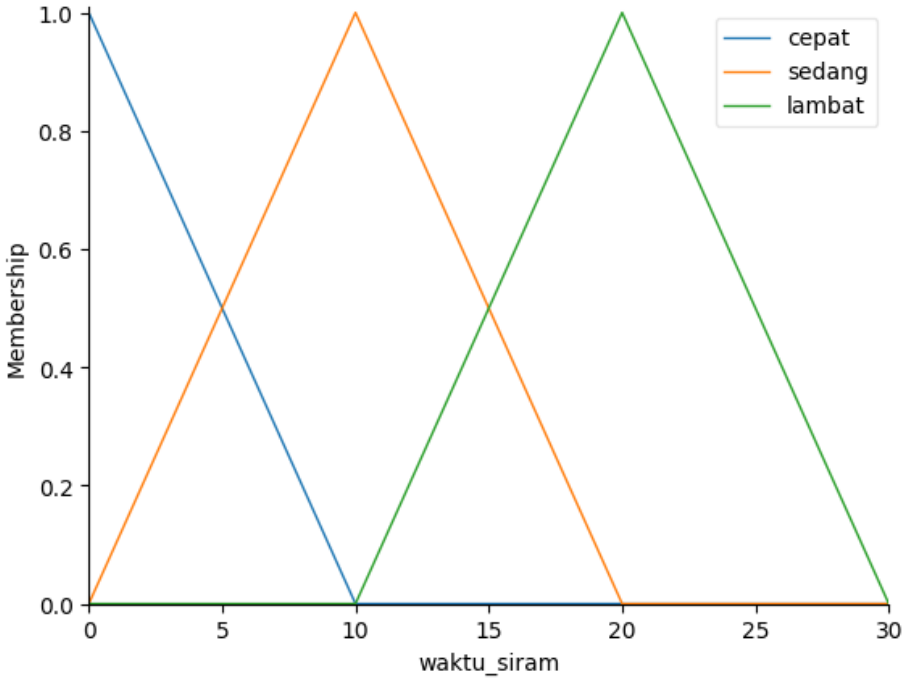
```
sistem_kontrol = ctrl.ControlSystem([rule1, rule2, rule3])
```

3.7. Menampilkan grafik fungsi keanggotaan

```
kelembaban.view()
waktu_siram.view()
```

Output:


kering
lembab
basah
Membership
kelembapan
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


cepat
sedang
lambat
Membership
waktu_siram
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30