PRAKTIKUM KECERDASAN BUATAN

"Disusun untuk memenuhi tugas ke-6 Mata Kuliah Praktikum Kecerdasan Buatan"

Dosen Pengampu : Fitri Nuraeni, S.Kom., M.Kom.

Disusun Oleh :

Khaila Thsabita S	2206026
Fajar Puniman	2206018
Doni Ramdan	2206024
Naufal Sirojudin	2206004

INSTITUT TEKNOLOGI GARUT
TEKNIK INFORMATIKA
2024

A. Certainty Factor (CF)

Code :

```
# Fakta Fakta
gejala = {
    "batuk" : True,
    "demam" : True,
    "pilek" : True,
    "sakit_tenggorokan" : True,
    "bersin" : True
}
# Aturan / CF Awal
penyakit = {
    "batuk" : (["batuk", "sakit_tenggorokan"], [1.0, 0.6]),
    "flu_ringan" : (["batuk", "demam", "pilek"], [0.4, 0.6, 0.8]),
    "flu_berat" : (["batuk", "demam", "pilek", "sakit_tenggorokan",
"bersin"], [0.4, 0.6, 0.6, 0.4, 0.8])
}

# Mengitung Certainty Factor
def hitung_nilai_cf(gejala_cf):
  # Mencari CF Penyakit
  cf_penyakit = 0.0
  nm_penyakit = ""

  for P, (GejalaP, CFP) in penyakit.items():
    # Inisialisasi nilai CF kombinasi
    cf_kombinasi = 0.0

    # Hitung CF gejala
    for G, cf_user in gejala_cf.items():
      # Hitung CF Gejala
      if G in GejalaP:
        for gp in GejalaP:
          i = 0;
          if gp == G:
            cf_gejala = CFP[i] * cf_user
          i = i+1
      else:
        cf_gejala = 0.0
```

```
        # Hitung CF Kombinasi
      cf_kombinasi = cf_kombinasi + (1 - cf_kombinasi) * cf_gejala

      # Mencari Nilai CF Penyakit
    if(cf_kombinasi > cf_penyakit):
      cf_penyakit = max(cf_penyakit, cf_kombinasi)
      nm_penyakit = P

  return nm_penyakit, cf_penyakit


# Contoh penggunaan
input_user = {"batuk" : 0.2,
              "demam" : 0.6,
              "pilek" : 0.8,
              "sakit_tenggorokan" : 0.2,
              "bersin" : 0.6
              }
cf = hitung_nilai_cf(input_user)
print("Hasil Diagnosa : ", cf)
```

Output :

```
Hasil Diagnosa :  ('flu_berat', 0.6675611648)
```

B. Memodifikasi Code Program Certainty Factor

Code :

```
# Aturan / CF Awal
penyakit = {
    "kulit_normal" : (["G001", "G002","G003", "G004", "G005", "G006",
"G011"], [0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8]),
    "Kulit_berminyak" : (["G007", "G008", "G009", "G016"], [0.8, 0.8,
0.8, 0.8]),
    "kulit_kering" : (["G001", "G014", "G015", "G016", "G017"], [0.6,
0.6, 0.8, 0.6, 0.6]),
    "kulit_kombinasi" : (["G007", "G014", "G015", "G016", "G017"],
[0.6, 0.4, 0.6, 0.4]),
    "kulit_sensitif" : (["G012", "G018", "G019", "G020"], [0.8, 0.8,
0.8, 0.8]),
}
# Mengitung Certainty Factor
def hitung_nilai_cf(gejala_cf):
```

```
  # Mencari CF Penyakit
  cf_penyakit = 0.0
  nm_penyakit = ""

  for P, (GejalaP, CFP) in penyakit.items():
    # Inisialisasi nilai CF kombinasi
    cf_kombinasi = 0.0

    # Hitung CF gejala
    for G, cf_user in gejala_cf.items():
      # Hitung CF Gejala
      if G in GejalaP:
        for gp in GejalaP:
          i = 0;
          if gp == G:
            cf_gejala = CFP[i] * cf_user
          i = i+1
      else:
        cf_gejala = 0.0

        # Hitung CF Kombinasi
      cf_kombinasi = cf_kombinasi + (1 - cf_kombinasi) * cf_gejala

      # Mencari Nilai CF Penyakit
    if(cf_kombinasi > cf_penyakit):
      cf_penyakit = max(cf_penyakit, cf_kombinasi)
      nm_penyakit = P

  return nm_penyakit, cf_penyakit
# Contoh penggunaan
input_user = {"G005" : 0.6,
              "G018" : 0.8,
              "G019" : 0.8,
              "G020" : 0.8,

              }
cf = hitung_nilai_cf(input_user)
print("Hasil Diagnosa : ", cf)
```

Output :

```
Hasil Diagnosa :  ('kulit_sensitif', 0.9533440000000001)
```

C. Teorema Bayes

Code :

```
penyakit = {
    "batuk": (["batuk","sakit_tenggorokan"], [1.0,0.6]),
    "flu_ringan":(["batuk","demam","pilek",],[0.4,0.6,0.8]),
    "flu_berat":
(["batuk","demam","pilek","sakit_tenggorokan","bersin"],[0.4,0.6,0.6,0
.4,0.8])
}
# menghitung probabilitas dengan metode inferensi teorema bayes
def hitung_nilai_probabilitas(gejala_user):
  #Dictionary untuk menyimpan pasterior probabilitas dari setiap
penyakit
  posterior_prob=0.0
  penyakit_prob="";

  #Iterasi melalui setiap penyakit
  for Penyakit, (gejala_penyakit, prob_penyakit) in penyakit.items():
    n_nilai_semesta = 0.0
    prob_gejala = []

  #Iterasi melalui setiap gejala
  for gejala in gejala_user:
    #jika gejala ada dalam penyakit
    if gejala in gejala_penyakit:
      prob_gejala.append(prob_penyakit[gejala_penyakit.index(gejala)])
      n_nilai_semesta += prob_penyakit[gejala_penyakit.index(gejala)]

  nilai_semesta_p_gejala = []
  for i in range(0, len(prob_gejala)):
    p_H = prob_gejala[i]/ n_nilai_semesta
    nilai_semesta_p_gejala.append(p_H)

  p_gejala =  []
  n_p_gejala = 0.0
  for i in range (0, len(nilai_semesta_p_gejala)):
    p_gejala.append(nilai_semesta_p_gejala[i]* prob_gejala[i])
    n_p_gejala =+ (nilai_semesta_p_gejala[i]* prob_gejala[i])

  n_p_gejala_penyakit =0
  for i in range(0, len (p_gejala)):
```

```
    p_gejala_penyakit = p_gejala[i]*prob_gejala[i]/n_p_gejala
    n_p_gejala_penyakit =+ p_gejala_penyakit

  print(Penyakit, n_p_gejala_penyakit)

  if(n_p_gejala_penyakit > posterior_prob):
    posterior_prob = max(posterior_prob, n_p_gejala_penyakit)
    penyakit_prob = Penyakit

  return penyakit_prob, posterior_prob
#Contoh penggunaan
input_user = ["demam", "pilek","bersin"]

probabilitas = hitung_nilai_probabilitas(input_user)
print("Hasil Diagnosa : ", probabilitas)
```

Output :

```
batuk 0.6
flu_ringan 0.4000000000000001
flu_berat 0.4000000000000001
Hasil Diagnosa :  ('batuk', 0.6)
```

D. Memodifikasi Code Program Teorema Bayes

Code :

```
penyakit = {
    "kulit_normal":
(["G001","G002","G003","G004","G005","G006","G0011"],
[0.8,0.8,0.8,0.8,0.8,0.8,0.8,0.8]),

"kulit_berminyak":(["G007","G008","G009","G016"],[0.8,0.8,0.8,0.8]),
    "kulit_kering":
(["G001","G005","G010","G011","G012"],[0.6,0.6,0.8,0.6,0.6]),
    "kulit_kombinasi":
(["G007","G014","G015","G016","G017"],[0.6,0.4,0.6,0.4,0.6]),
    "kulit_sensitif" :
(["G012","G018","G019","G020"],[0.8,0.8,0.8,0.8])
}
# menghitung probabilitas dengan metode inferensi teorema bayes
def hitung_nilai_probabilitas(gejala_user):
  #Dictionary untuk menyimpan pasterior probabilitas dari setiap
penyakit
  posterior_prob=0.0
```

```
  penyakit_prob="";

  #Iterasi melalui setiap penyakit
  for Penyakit, (gejala_penyakit, prob_penyakit) in penyakit.items():
    n_nilai_semesta = 0.0
    prob_gejala = []

  #Iterasi melalui setiap gejala
  for gejala in gejala_user:
    #jika gejala ada dalam penyakit
    if gejala in gejala_penyakit:
      prob_gejala.append(prob_penyakit[gejala_penyakit.index(gejala)])
      n_nilai_semesta += prob_penyakit[gejala_penyakit.index(gejala)]

  nilai_semesta_p_gejala = []
  for i in range(0, len(prob_gejala)):
    p_H = prob_gejala[i]/ n_nilai_semesta
    nilai_semesta_p_gejala.append(p_H)

  p_gejala =  []
  n_p_gejala = 0.0
  for i in range (0, len(nilai_semesta_p_gejala)):
    p_gejala.append(nilai_semesta_p_gejala[i]* prob_gejala[i])
    n_p_gejala =+ (nilai_semesta_p_gejala[i]* prob_gejala[i])

  n_p_gejala_penyakit =0
  for i in range(0, len (p_gejala)):
    p_gejala_penyakit = p_gejala[i]*prob_gejala[i]/n_p_gejala
    n_p_gejala_penyakit =+ p_gejala_penyakit

  print(Penyakit, n_p_gejala_penyakit)

  if(n_p_gejala_penyakit > posterior_prob):
    posterior_prob = max(posterior_prob, n_p_gejala_penyakit)
    penyakit_prob = Penyakit

  return penyakit_prob, posterior_prob
#Contoh penggunaan
input_user = ["G005", "G018","G019","G020"]

probabilitas = hitung_nilai_probabilitas(input_user)
print("Hasil Diagnosa : ", probabilitas)
```

Output :

```
kulit_sensitif 0.8
Hasil Diagnosa :  ('kulit_sensitif', 0.8)
```

```
kulit_normal 0.8000000000000002
kulit_berminyak 0
kulit_kering 0.6
kulit_kombinasi 0
kulit_sensitif 0.8
Hasil Diagnosa :  ('kulit_normal', 0.8000000000000002)
```

Penjelasan :

1. Definisi Data:

Kamus penyakit berisi informasi tentang berbagai jenis penyakit, termasuk:

- Nama penyakit (misalnya: "kulit_normal").
- Daftar gejala yang terkait dengan penyakit tersebut (misalnya: ["G001", "G002", ...]).
- Probabilitas awal (prior) untuk setiap gejala pada setiap penyakit (misalnya: [0.8, 0.8, ...]).

2. Fungsi hitung_nilai_probabilitas:

- Fungsi ini menerima daftar gejala yang dialami pengguna (gejala_user) sebagai input.
- Inisialisasi variabel posterior_prob untuk menyimpan probabilitas posterior tertinggi dan penyakit_prob untuk menyimpan nama penyakit dengan probabilitas posterior tertinggi.
- Melakukan iterasi melalui setiap penyakit dalam kamus penyakit:
  - Menghitung nilai n_nilai_semesta, yaitu total probabilitas semua gejala pada penyakit tersebut.
  - Inisialisasi daftar prob_gejala untuk menyimpan probabilitas setiap gejala yang dialami pengguna.
  - Melakukan iterasi melalui setiap gejala yang dialami pengguna:
    - Jika gejala tersebut terdapat dalam daftar gejala penyakit, hitung probabilitas kemunculan gejala tersebut dan tambahkan ke daftar prob_gejala.
    - Hitung nilai n_nilai_semesta dengan menambahkan probabilitas gejala yang telah dihitung.

- Menghitung nilai nilai_semesta_p_gejala, yaitu probabilitas setiap gejala *diberikan* penyakit tersebut.
- Menghitung nilai p_gejala, yaitu probabilitas gabungan dari semua gejala *diberikan* penyakit tersebut.
- Menghitung nilai n_p_gejala, yaitu normalisasi probabilitas p_gejala.
- Menghitung nilai n_p_gejala_penyakit, yaitu probabilitas posterior untuk penyakit tersebut *diberikan* gejala yang dialami pengguna.
- Memperbarui nilai posterior_prob dan penyakit_prob jika nilai n_p_gejala_penyakit lebih besar dari nilai sebelumnya.

Mengembalikan nama penyakit dengan probabilitas posterior tertinggi (penyakit_prob) dan nilai probabilitas posteriornya (posterior_prob).

3. Contoh Penggunaan:

- Kode mendefinisikan daftar gejala pengguna (input_user).
- Memanggil fungsi hitung_nilai_probabilitas dengan daftar gejala pengguna sebagai input.
- Hasil diagnosis beserta probabilitas posteriornya ditampilkan.

Penjelasan Algoritma :

Langkah-langkah utama

1. **Inisialisasi variabel:**
    - posterior_prob untuk menyimpan probabilitas posterior tertinggi.
    - penyakit_prob untuk menyimpan nama penyakit dengan probabilitas posterior tertinggi.
2. **Iterasi melalui setiap penyakit:**
    - Hitung probabilitas n_nilai_semesta untuk semua gejala pada penyakit tersebut.
    - Inisialisasi daftar prob_gejala untuk menyimpan probabilitas setiap gejala yang dialami pengguna.
    - Iterasi melalui setiap gejala yang dialami pengguna:
        - Jika gejala terdapat dalam daftar gejala penyakit, hitung probabilitasnya dan tambahkan ke prob_gejala.
        - Hitung n_nilai_semesta dengan menambahkan probabilitas gejala yang telah dihitung.
    - Hitung nilai_semesta_p_gejala, yaitu probabilitas setiap gejala *diberikan* penyakit tersebut.

- Hitung p_gejala, yaitu probabilitas gabungan dari semua gejala *diberikan* penyakit tersebut.
- Hitung n_p_gejala, yaitu normalisasi probabilitas p_gejala.
- Hitung n_p_gejala_penyakit, yaitu probabilitas posterior untuk penyakit tersebut *diberikan* gejala yang dialami pengguna.
- Perbarui posterior_prob dan penyakit_prob jika n_p_gejala_penyakit lebih besar dari nilai sebelumnya.

3. **Kembalikan hasil:**
    - Nama penyakit dengan probabilitas posterior tertinggi (penyakit_prob).
    - Nilai probabilitas posterior tertinggi (posterior_prob).