Laporan Analisis Klasifikasi Decision Tree dengan Gentic Algorithm

Diberikan Data Training untuk membuat aturan yang benar untuk menguji inputan data baru berupa Data Uji dengan learning untuk menemukan aturan yang diharapkan bisa berlaku umum untuk data data yang belum pernah diketahui.

A.Genetic Algrotihm dengan Decision Tree Learning

Genetic Algorithm (GA) adalah bagian dari evolutionary Algorithm yang merupakan salah satu algoritma yang tepat digunakan dalam menyelesakan masalah optimasi mencari kemungkinan dari calon solusi untuk mendapatkan solusi yang optimal dalam penyelesain masalah.Dalam kasus ini mencari aturan yang paling optimal untuk pengambilan keputusan apakah boleh terbang/tidak.

Decision Tree Learning adalah salah satu metode belajar yang sangat popular dan banyak digunakan secara praktis. Metode ini merupakan metode yang berusaha menemukan fungsi fungsi pendekatan yang bernilai diskrit dan tahan terhadap kesalahan serta mampu mempelajar ekspresi ekspresi disjunctive.

B. Analisis dan penjelasan strategi penyelesain masalah

```
import pandas as pd
from pandas import DataFrame
import tkinter as tk
from tkinter import filedialog
```

```
15  ProbMutationRate  = 0.1
16  ProbCrossOver     = 0.8
17  UkPop             = 10
18  JumGen            = 15
19  Generasi          = 1000
```

```
51 def GeneratePopulation(x):
52     Kromosom = []
53     Populasi = []
54     for i in range(x):
55         for j in range(JumGen):
56             Kromosom.append(random.randint(0, 1))
57         Populasi.append(Kromosom)
58     return Populasi
```

import pandas berguna untuk mengolah data seperti mengimport file csv,DataFrame digunakan untuk mengexport file csv,library tkinter untuk tampilan gui saat melakukan export file csv untuk hasil dari data uji Setelah itu lakukan inisiasi variabel,pada baris ke 15 ProbMutation rate digunakan untuk probability dari mutasi,ProbCrossOver digunakan untuk probability Cross Over,JumGennya sebanyak 15 dan Generasi 1000 karena generasi yang lama akan digantikan dengan generasi yang lebih baik jadi saya memperbanyak jumlah generasi.Selanjutnya membuat fungsi generate Populasi yang diguanakn untuk membangun kromosom yang digunakan untuk mengcompare aturan keputusan sebayak jumlah dari generasi yaitu 15 karena disini terdapat 5 atribut yang mewakili 5 aturan yang menghasilkan 15 bit.

```
41  def CheckAturan(kolom, aturan, data):
42      i = kolom.index(data)
43      Checkaturan = aturan[i] == 1
44      return Checkaturan
```

Pada Baris ke 41 sampai 44 saya membuat fungsi checkAturan yang dimana parameternya berisi kolom,aturan dan juga data jadi disini akan mengecek aturan dari indeks yang ada pada data apabail sama dengan 1 maka akan mengembalikan nilaik checaturan tersebut yang akan digunakan pada fungsi selanjutnya yaitu decision tree

```
60 def DecisionTree(Kromosom, data):
61     aturan = Split(Kromosom, JumGen)
62
63     # del rules[-1:]
64     for i,j in enumerate(aturan):
65         suhu = j[0:3]
66         waktu = j[3:7]
67         kondisi = j[7:11]
68         kelembapan = j[11:14]
69         terbang = j[14]
70         if CheckAturan(Suhu, suhu, data[0]) and CheckAturan(Waktu, waktu, data[1
71             if terbang == 1:
72                 return 'Ya'
73             else:
74                 return 'Tidak'
75     return 'Tidak'
```

Membuat fungsi DecisionTree untuk mengcompare antara data training dengan kromosoom yang ada pada populasi yang sudah kita generate tadi sebanyak 5 atribut dan akan mengembalikan nilai Ya apabila 5 atribut tersebut dipenuhi dan akan mengembalikan nilai Tidak apabila tidak terpenuhi.

```python
def Fitnees(populations):
    fitnees = []
    for i in range(UkPop):
        total = 0
        Kromosom = populations[i]
        for j in data:
            DecisionTree1 = DecisionTree(Kromosom, j)
            if DecisionTree1 == j[4]:
                total += 1
        fitnees.append({'i': i,'fitnees': total / len(data)})
    return fitnees
```

Fungsi Fitnees disini untuk mencari individu terbaik dengan cara membandingkan jumlah yang benar pada fungsi DecisionTree lalu dihitung total benarnya sebanyak apa yang akan dibandingan dengan banyaknya data latih untuk melihat persentasi individu terbaik untuk mencari parent.

```python
def Parent(populations, fitnees):
    index = sorted(fitnees, key=lambda x
    parent1 = populations[index[0]['i']]
    parent2 = populations[index[1]['i']]
    return parent1, parent2
```

Fungsi parent didapat dari fintees dan akan dilakukan pencarian parent berdasarkan individu yang terbaik.

```python
def Crossover(parrent, ProbCrossOver):
    p1, p2 = parrent
    result = []
    if random.uniform(0, 1) > ProbCrossOver:
        return [p1[:], p2[:]]
    if len(p2) < len(p1):
        p1, p2 = p2, p1
    r1 =random.randint(0, len(p1)//2)
    r2 =random.randint(len(p1)//2, len(p1))
    i = [r1,r2]
    lenghtp1 = i[1] - i[0]
    p1_gap = lenghtp1 % JumGen
    p2_1 = [i[0], i[0] + lenghtp1]
    if p2_1 not in result:
        result.append(p2_1)
    p2_2 = [i[0], i[0] + p1_gap]
```

Selanjutnya membuat fungsi Crossover dengan membuat kondisi dimana jika hasil dari pembangkitan bilangan acak secara random lebih besar dari probabilty cross over diantara 80%.CrossOver merupakan salah satu komponen yang penting dalam GA. Dengan skema tertentu,dua individu dipilih sebagai orangtua selanjutnya ditentukan titik pindah silang secara acak.Jika diasumsikan L adalah panjang kromosom,maka titik pindah silang berada antara 1 sampai L-1

Fungsi Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat cross over.mutasi diterapkan dengan probabilitas yang sangat kecil.jika mutasi sering dilakukan,maka akan menghasilkan individu yang lemah karena gen pada individu yang unggul akan dirusak

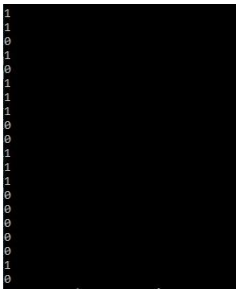
```python
def SurvivorSelection(populations, fitnees, child):
    index = sorted(fitnees, key=lambda x: x['fitnees'
    index1 = index[len(index)-1]['i']
    index2 = index[len(index)-2]['i']
    populations[index1] = child[0]
    populations[index2] = child[1]
    return populations
```

Fungsi Survivor Selection digunakan untuk pergantian populasi yang akan menghasilkan individu baru crossover dan mutasi dengan mempertahankan individu terbaik.Proses pergantiannya dilakukan dengan mengganti individu yang memiliki nilai fitnees kecil dan membandingkan anak dengan kedua orang tua.

3. **Parameter GA yang optimum**

```
ProbMutationRate  = 0.1
ProbCrossOver     = 0.8
UkPop             = 10
JumGen            = 15
Generasi          = 1000
```

Parameter GA yang optimum menurut saya ketika ProbabilityMutation nya kecil sekali seperti 0. 1 karena dapat merusak bit dan memperbesar probability cross over 0.8 dan membuat generasi yang besar karena akan dilakukan pergantian populasi yang banyak untuk menghasilkan kromosom yang terbaik



Dan output dari program adalah tebakan dari data test yaitu 1 artinya Ya dan 0 artinya Tidak berikut adalah hasil outputnya :{1,1,0,1,0,0,1,1,0,1,0,0,1,0,0,0,1,0,1,0},

```
Kromosom Tebaik
[0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
```

Dan gambar diatas adalah kromosom terbaik,Berapa Panjang Kromosom terbaik? tidak bisa diketahui karena berubah rubah.