**1. Deskripsi Masalah LAPORAN ANALISIS : GENETIC ALGORITHM FOR LEARNING**

*Decision tree* adalah model prediksi menggunakan struktur pohon atau struktur berhirarki. Konsep dari pohon keputusan adalah mengubah data menjadi *decision tree* dan aturan-aturan keputusan. Dalam laporan ini, akan dibangung sebuah sistem klasifikasi *decision tree* yang dibangung dengan menggunakan *genetic algorithm* untuk mengklasifikasikan beberapa data uji yang ada.

**2. Objective yang dibangun**

Membuat sebuah sistem klasifikasi decision tree dengan mengimplementasi-kan Genetic Algorithm sebagai alat untuk menguji beberapa data uji (Data Test) baru.

**3. Analisis Algoritma**

- Pertama perlu inisiasi jumlah populasi, rule awal, probabilitas mutasi, serta banyak loop data.

```
##################################
Jumlah_Populasi = 10
Start_Rule_Count = 4
Mutasi_Prob = 0.02
loop = 6000
##################################
```

- Membuat fungsi yang bertugas untuk menjawab pertanyaan daripada data uji yang nantinya akan dibaca.

```
def jawab(individu, question):
    answer = -1;
    for rule in range(0, int(len(individu) / 15)):
        isValid = True;

        for j in range(0, (len(question) - 1)):
            if (question[j] == 1 and individu[(rule * 15) + j] != 1):
                isValid = False;
                break;

        if (isValid):
            answer = rule
            break;

    if (answer == -1):
        return int(individu[len(individu) - 1] == 0);
    else:
        return individu[answer * 15 + 14];
```

- Membuat fungsi untuk men-*generate* Populasi. Populasi merupakan kumpulan dari individu yang akan degenerate secara random antara bilangan -∞ $s/d$ ∞ (real). Menghitung nilai Fitness dengan menggunakan fungsi jawab yang telah dibuat sebelumnya.

```
# Fungsi Menentukan Populasi
def Populasi(kesehatan, individu):
    return (kesehatan[individu] / sum(kesehatan))


# Function untuk mencari Fitness
def Fitness(individu, dataTrain):
    benar = 0;
    for i in range(0, len(dataTrain)):
        ans = jawab(individu, dataTrain[i]);
        if (ans == dataTrain[i][14]):
            benar += 1;

    return benar / len(dataTrain);
```

- Membuat fungsi untuk menghitung nilai dari crossover dan mutasi dari semua generasi yang ada. Persilangan 1 sebagai proses inserting dan persilangan untuk penentuan crossover.

```
def Crossover1(parent, anak, titik1, titik2):
    for i in range(titik2[0], titik2[1] + 1):
        anak.pop(titik2[0]);

    for i in range(titik1[0], titik1[1] + 1):
        anak.insert(titik2[0] + (i - titik1[0]), parent[0][i]);

    return anak;


def Crossover2(parent, anak, titik2):
    for i in range(titik2[0], titik2[1] + 1):
        anak[i] = parent[1][i];
    return anak;
```

```
def Mutation(individu):
    for ind in range(0, len(individu)):
        if (random() <= Mutasi_Prob):
            individu[ind] = int(individu[ind] == 0);

    return individu
```

- Membuat fungsi untuk load data set dari file csv kedalam program, data set akan diubah menjadi biner dengan function yang ada lalu akan di load dan digunakan di dalam program.

```
# Fungsi untuk load data train
def load(fileLok):
    arrTrain = loadText(fileLok);
    Train = [ubahKeBiner(data) for data in arrTrain];
    return Train;


# Fungsi untuk load data
def loadText(fileLok):
    f = open(fileLok, "r");
    dataString = [];
    kalimat = f.readline();
    while (kalimat != ""):
        kalimat = kalimat.replace("\n", "")
        dataString.append(kalimat);
        kalimat = f.readline();
    f.close();
    return dataString;
```

- Masuk ke main program, pertama-tama data csv akan di load terlebih dahulu kedalam program untuk diproses.
- Dilakukan perulangan yang berfungsi untuk melakukan penyeleksian terhadap semua data dengan fitness yang berbeda. Nantinya akan dicari bestfit dari semua data set yang ada.
- Terdapat banyak proses yang dilakukan didalam perulangan seperti inisiasi orang tua, menentukan 2 titik random, melakukan proses crossover, proses mutasi, serta pemilihan survivor daripada data yang memiliki bestfit terbaik.
- Dalam menentukan orang tua, agar mempermudah dalam melakukan seleksi maka Parent ke 0 diset menjadi yang paling pendek.
- Dilakukan juga proses crossover dan mutasi dengan memanggil fungsi yang telah dibuat sebelumnya.
- Akan dibuat juga perulangan sebagai proses seleksi dari data uji yang ada dan dianggap terbaik untuk setiap perulangan, lalu akan kembali di compare dengan nilai lainnya di perulangan yang berbeda sehingga didapatkan nilai fitness yang terbaik.

**4. Kesimpulan**

Semakin besar jumlah loopnya, maka akurasinya akan semakin baik.