1. **Implementasi GA untuk membangun model regresi linear**

Data : Data terdiri dari 1 variabel independen X (nilainya terdiri dari 1 s/d 300) dan 1 variabel dependen Y (nilai random yang terdistribusi normal).

Kromosom : Kromosom terdiri dari dua genotip (merepresentasikan β dan θ pada simple regresi linear) yang mana masing-masing genotip memiliki fenotip bernilai bilangan real. Pada program ini Saya menetapkan bilangan real tersebut dalam range -50.0 s/d 50.0. Salah satu contoh dari kromosomnya adalah [-22.01963543, -9.99296509].

Fungsi fitness : Pada simple regresi linear, kita ingin meminimalkan nilai error, sehingga fungsi fitnessnya adalah nilai error yang dihasilkan dari y aktual dan y prediksi. Pada program ini Saya menggunakan formula MSE sebagai fungsi fitness. Selain itu juga dihitung nilai R squarednya untuk menilai seberapa baik model yang dibangun.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_{predict_i})^2$$

$$Y_{predict} = \beta + \theta X$$

```
def get_fitness_chromosome(individual):
    b = individual[0]
    theta = individual[1:]

    Y_predict = b + (X*theta)
    Y_res = Y-Y_predict
    error = np.sum((Y_res)**2)/len(Y)
    r_squared = r2_score(Y,Y_predict)*100
    return error, r_squared
```

Selection : Proses selection (Parent Selection) pada program ini dilakukan dengan random sample (parent 1 dan parent 2) sebanyak jumlah populasi dikurang 1 (karena ada proses elitisme) dari 20% kromosom dengan nilai fitness terendah.

Crossover : Pada tahap crossover, nilai probabilitas crossover yang digunakan adalah 0.7. Nilai ini menentukan terjadi atau tidak proses crossover. Jika nilai random yang dibangkitkan < 0.7 maka terjadi proses crossover. Jenis crossover yang digunakan pada program ini adalah one point crosssover.

```
def crossover(parent1, parent2, pc):
    offspring1 = []
    offspring2 = []
    point = random.randint(0,len(parent1)-1)
    if random.random()<pc:
        for idx in range(0, point):
            offspring1.append(parent1[idx])
            offspring2.append(parent2[idx])
        for idx in range(point,len(parent1)):
            offspring1.append(parent2[idx])
            offspring2.append(parent1[idx])
    else:
        offspring1 = parent1
        offspring2 = parent2
    return offspring1, offspring2
```

Mutasi : Pada tahap mutasi, nilai probabilitas mutasi yang digunakan adalah 0.25. Nilai ini menunjukkan terjadi atau tidak proses mutasi. Jika nilai random yang dibangkitkan < 0.25 maka terjadi proses mutasi. Jenis mutasi yang digunakan pada program ini adalah one point mutation. Nilai fenotip dari gen yang dilakukan mutasi akan diganti dengan suatu bilangan real lainnya dengan range yang sama yaitu -50.0 s/d 50.0.

```python
def mutasi(offspring, pm):
    offspring_m = []
    point = random.randint(0,len(offspring)-1)
    if random.random()<pm:
        for idx in range(0, point):
            offspring_m.append(offspring[idx])
        for idx in range(point, len(offspring)):
            off1 = random.uniform(-50.0,50.0)
            offspring_m.append(off1)
        return offspring_m
    else:
        return offspring
```
✓ 0.0s

Survivor Selection : Pada program ini dilakukan prinsip elitisme. Prinsip elitisme merupakan prinsip dimana algoritma selalu menyimpan kromosom yang memiliki nilai fitness terbaik (MSE terkecil). Selain itu pada tahap survivor selection hanya 20% (dengan nilai fitness terkecil) dari jumlah populasi yang akan dipertimbangkan untuk generasi berikutnya.

Perbandingan model simple regresi linear dengan GA dan LSM :

Pada program ini dilakukan eksperimen dengan parameter-parameter sebagai berikut,
Individual_size = 2 (jumlah gen dalam 1 kromosom),
Population_size = 100,
Generation = 500,
Pc (Probabilitas Crossover) = 0.7,
Pm (Probabilitas Mutasi) = 0.25,
Selection_size = 20% dari Population_size,
Berdasarkan tabel, nilai error dan R2 yang dihasilkan dengan GA tidak jauh berbeda dengan menggunakan LSM.

Metode	Persamaan	Error	R2
Simple Regresi Linear dengan GA	$y = (-5.684) + 1.0201X$	376.0525	95.28
Simple Regresi Linear dengan LSM	$y = (-0.9644) + 1.0068X$	367.3572	95.39

Referensi :

- Kodingan GA di LMS/ https://medium.com/the-andela-way/on-genetic-algorithms-and-their-application-in-solving-regression-problems-4e37ac1115d5
- https://github.com/ghinakh/genetic-algorithm/blob/main/%5BTUPRO1%5D%20genetic-algorithm.ipynb

2. Ide dari implementasi GA untuk mengoptimasi investasi saham

Case : Budi akan menyisihkan 1 juta dari gajinya setiap bulan selama 1 tahun, karena banyak teman Budi yang melakukan investasi saham maka Budi juga tertarik untuk melakukan investasi saham. Permasalahannya Budi belum tau akan melakukan investasi saham kemana saja oleh karena itu Budi mencari informasi terlebih dahulu. Hal yang Budi pertimbangkan adalah tingkat keuntungan. Budi ingin proporsi saham yang dapat menghasilkan tingkat keuntungan yang optimal. Terdapat 4 saham yang dipertimbangkan Budi. Sehingga Budi mengumpulkan data historis dalam 1 tahun kebelakang yang diperlukan untuk menghitung proporsi saham (Closing Price, IHSG, dan Risk Free). Dari data historis tersebut dapat dihitung

nilai Beta Portofolio, Alpha Portofolio, dan Expected Return Market dari masing-masing saham.

Implementasi dengan GA

Kromosom : Terdiri dari 4 genotip yang mana nilai fenotipnya adalah bilangan real antara 0 s/d 1. Fenotip ini menunjukkan proporsi pada masing-masing saham. Sehingga ketika fenotip dari 4 gen dijumlahkan maka hasilnya adalah 1.
Fungsi Fitness : Budi ingin memaksimalkan keuntungan (optimal). Oleh karena itu, fungsi fitnessnya adalah expexted return portofolio. Permasalahan portofolio optimal bergantung pada model asset pricing yang digunakan. Salah satunya adalah Single Index Model (SIM). Persamaan dari fungsi fitnessnya adalah Expected return portofolio = Rf + sum(proporsi * beta * (Rm - Rf)). Dimana Rf adalah suku bunga risk free rate dan Rm adalah expected return dari indeks saham yang dipilih sebagai acuan. Misalkan kromosomnya adalah [0.1, 0.5, 0.4, 0]. Untuk menghitung fitnessnya maka dibutuhkan nilai Rf, Rm, dan beta. Misalkan Rf nya 3%, Rm nya 8%, dan beta untuk masing-masing saham 0.8, 1.2, 1.0, dan 1.5. Dengan menggunakan rumus maka expected return portofolionya adalah 3% + (0.1 * 0.8 * (8%-3%)) + (0.5 * 1.2 * (8%-3%)) + (0.4 * 1.0 * (8%-3%)) + (0 * 1.5 * (8%-3%)) = 0.085872.
Crossover, Mutasi, dan Seleksi : Crossover, mutasi, dan seleksi dilakukan seperti proses GA pada umumnya.

Hasil akhir dari program diharapkan dapat menghasilkan kromosom yang mengoptimalkan portofolio saham. Misalnya hasil akhirnya adalah 61.093361%, 36.465086%, 1.219779%, 1.221775% dengan fitness 5.03182%. Maka 1 juta per bulan alokasinya akan menjadi seperti berikut:
Saham A : 61.093361% * 1jt = 610933
Saham B : 36.465086% * 1jt = 364650
Saham C : 1.219779% * 1jt = 12197
Saham D : 1.221775% * 1jt = 12217
Expected Return dalam 1 bulan : 5.03182% * 1jt = 50318
Expected Return dalam 1 tahun (asumsi Budi selalu invest per bulan selama 1 tahun) : 5.03182% * 12jt = 603818

Referensi : Tampilan Penentuan Portofolio Saham Optimal Menggunakan Algoritma Genetika (ub.ac.id)