Implementasi Algoritma Haversine Formula Dan Probabilitas Diskrit Dalam Sistem Rekomendasi Lokasi Kopi Kenangan Menggunakan Pemrograman Go Language

Salsabilla Fatimah Az Zahra, Shofiyyah

[1]Salsabilla Fatimah Az Zahra, STMIK Tazkia Kota Bogor
[2]Shoiyyah, STMIK Tazkia Kota Bogor

E-mail : *shofiyyah2702@gmail.com*

Abstrak

Pertumbuhan gerai kopi *grab-and-go* seperti Kopi Kenangan menuntut adanya sistem pencarian lokasi yang presisi dan cerdas. Penelitian ini bertujuan membangun sistem rekomendasi lokasi terdekat menggunakan pendekatan hibrida antara algoritma geometri dan statistik. Metode yang digunakan adalah *Haversine Formula* untuk menghitung jarak akurat pada permukaan bumi yang bulat, serta *Probabilitas Diskrit* dengan teknik *Inverse Distance Weighting* untuk memberikan bobot peluang pemilihan cabang. Sistem diimplementasikan menggunakan bahasa pemrograman Go (Golang) untuk memaksimalkan efisiensi komputasi. Pengujian dilakukan dengan titik uji di Ciluar Asri, Bogor, terhadap tiga cabang berbeda. Hasil eksperimen menunjukkan sistem mampu merekomendasikan cabang Kopi Kenangan Kedunghalang sebagai prioritas utama dengan jarak 2,45 km dan probabilitas pemilihan sebesar 62.3%. Sistem ini terbukti efektif sebagai alat pendukung keputusan bagi konsumen dalam menentukan lokasi gerai yang paling efisien untuk dikunjungi.

Kata kunci : *Haversine,* Probabilitas Diskrit*,* Golang, Sistem Rekomendasi.

Implementasi Algoritma Haversine Formula Dan Probabilitas Diskrit Dalam Sistem Rekomendasi Lokasi Kopi Kenangan Menggunakan Pemrograman Go Language

Abstract

The growth of grab-and-go coffee shops like Kopi Kenangan demands a precise and intelligent location search system. This study aims to build a nearest location recommendation system using a hybrid approach of geometric algorithms and statistics. The method used is the Haversine Formula to calculate accurate distances on the spherical earth surface, and Discrete Probability with Inverse Distance Weighting technique to weigh the likelihood of branch selection. The system is implemented using the Go (Golang) programming language to maximize computational efficiency. Testing was conducted with a test point in Ciluar Asri, Bogor, against three different branches. Experimental results show the system is capable of recommending the Kopi Kenangan Kedunghalang branch as the top priority with a distance of 2.45 km and a selection probability of 62.3%. This system proves effective as a decision support tool for consumers in determining the most efficient outlet location to visit.

Keywords : *Haversine, Discrete Probability, Golang, Recommendation System.*

---

1. Pendahuluan

Pertumbuhan industri kedai kopi di Indonesia mengalami lonjakan yang sangat signifikan dalam satu dekade terakhir. Riset pasar mencatat bahwa jumlah gerai kopi meningkat drastis seiring dengan tingginya konsumsi domestik, di mana *brand* besar seperti Kopi Kenangan mendominasi pasar melalui ekspansi gerai yang agresif (Toffin, 2020). Dengan banyaknya titik lokasi gerai tersebut, konsumen seringkali dihadapkan pada tantangan untuk menentukan gerai mana yang paling dekat dari posisi mereka saat ini secara akurat dan efisien. Kebutuhan akan informasi lokasi yang real-time dan presisi menjadi krusial bagi konsumen yang memiliki mobilitas tinggi, sehingga diperlukan sebuah sistem pencarian lokasi yang mampu menjawab kebutuhan tersebut

dengan cepat.

Dalam upaya pengembangan sistem pencarian lokasi (Location Based Service), pemilihan algoritma dan teknologi sangat menentukan hasil akhir. Penelitian sebelumnya oleh Rio dkk,. (2017) membahas pencarian lokasi terdekat menggunakan metode *Euclidean Distance*. Meskipun metode ini sederhana, hasil penelitian menunjukkan bahwa *Euclidean Distance* mengasumsikan permukaan datar, sehingga tingkat akurasinya menurun ketika diterapkan pada koordinat geografis bumi yang sebenarnya bulat[1]. Di sisi lain, dari segi performa sistem, penelitian yang dilakukan oleh Suroto (2018) menggunakan bahasa pemrograman PHP untuk sistem berbasis lokasi. Namun, ditemukan bahwa PHP memiliki keterbatasan dalam hal *concurrency* dan kecepatan eksekusi ketika menangani *request* data dalam jumlah besar dibandingkan dengan bahasa pemrograman modern yang terkompilasi[2].

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pendekatan hibrida. Pertama, penggunaan Formula Haversine untuk perhitungan jarak presisi yang memperhitungkan kelengkungan bumi. Kedua, penerapan Probabilitas Diskrit untuk memberikan bobot peluang pemilihan cabang, sehingga sistem berfungsi sebagai pendukung keputusan (decision support), bukan sekadar alat ukur. Implementasi dilakukan menggunakan bahasa Golang untuk memaksimalkan efisiensi konkurensi [3].

2. Metodologi

Data yang digunakan dalam penelitian ini adalah data lokasi persebaran gerai Kopi Kenangan yang diambil menggunakan layanan Google Maps. Data ini mencakup nama cabang, alamat lengkap, serta titik koordinat geografis yang terdiri dari Garis Lintang (*Latitude*) dan Garis Bujur (*Longitude*) sebagai variabel acak diskrit dalam ruangan sampel lokasi. Titik pusat pengujiannya ditetapkan di salah satu rumah perumahan Ciluar Asri, Bogor.

Tabel 1. Sampel Data Lokasi

Nama Cabang	Alamat	Latitude (ϕ)	Longitude (λ)
Rumah (Titik Awal)	Perumahan Ciluar Asri Blok A4 No. 3	-6.5398	106.8202
Kopi Kenangan Kedunghalang	Jl. Raya Pemda No.1, Kedunghalang	-6.5601	106.8115
Kopi Kenangan Pom Bensin Pajajaran	Jl. Raya Pajajaran No.127	-6.5785	106.8088
Kopi Kenangan Sudirman Bogor	Jl. Jend. Sudirman No.60, Sempur	-6.182313	106.8364646

Dalam penelitian ini, untuk menghitung jarak (d) antara pengguna dan cabang, digunakan persamaan Haversine yang mengakomodasi jari-jari bumi ($R = 6371km$):

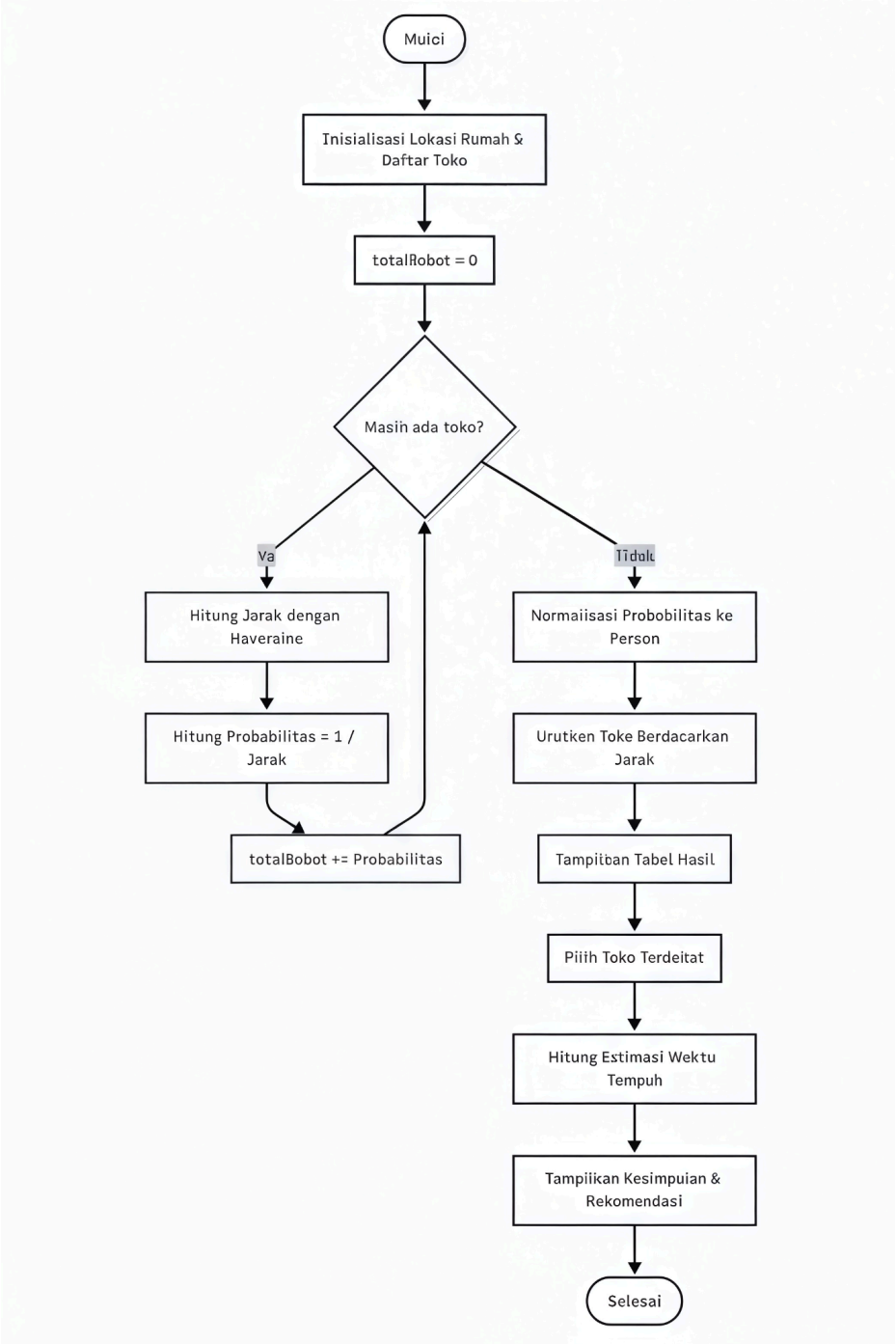
$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + cos(\phi_1)cos(\phi_2)\, sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$d = R \cdot 2 \cdot arctan\,2\left(\sqrt{a}, \sqrt{1-a}\right)$$

Tetapi dalam penelitian ini kami tidak menggunakan sistem terpendek juga, melainkan dengan menghitung peluang (P) pengguna untuk memilih cabang Kopi Kenangan berdasarkan kedekatan. Digunakan metode Inverse Distance Weighting (IDW), dimana bobot (W) berbanding terbalik dengan jarak.

1. Rumus menghitung $W_i = \frac{1}{d_i}$
2. Rumus menghitung Probabilitas Normalisasi = $P(X_i) = \frac{W_i}{\sum_{j=1}^{n} W_j} \times 100\%$

Sementara jika divisualisasikan sebuah pemrogramannya dalam bentuk alur logika, maka akan terlihat seperti ini:



Gambar 1. Flowchart

3. Eksperimen Dan Hasil

Sistem ini diimplementasikan menggunakan bahasa pemrograman Golang. Penggunaan tipe data struct digunakan untuk mengorganisir data cabang, sedangkan paket math digunakan untuk fungsi trigonometri. Berikut struktur data dan logika perhitungan probabilitas:

```
var totalBobot float64


for i := range daftarToko {
    daftarToko[i].Jarak = itungJarak(rumahSaya.Posisi, daftarToko[i].Posisi)

    // Rumus: Semakin kecil jarak, semakin besar peluang
    daftarToko[i].Probabilitas = 1.0 / daftarToko[i].Jarak
    totalBobot += daftarToko[i].Probabilitas
}

for i := range daftarToko {
    daftarToko[i].Probabilitas = (daftarToko[i].Probabilitas / totalBobot) * 100
}

sort.Slice(daftarToko, func(i, j int) bool {
    return daftarToko[i].Jarak < daftarToko[j].Jarak
})
```

Gambar 2. Kode logika perhitungan probabilitas

Selanjutnya, berikut merupakan struktur data logika pada hitungan Haverse:

```
22 func itungJarak(asal, tujuan Koordinat) float64 {
23     const jariBumi = 6371.0
24     dLat := (tujuan.Lat - asal.Lat) * (math.Pi / 180)
25     dLon := (tujuan.Lon - asal.Lon) * (math.Pi / 180)
26     a := math.Sin(dLat/2)*math.Sin(dLat/2) +
27         math.Cos(asal.Lat*(math.Pi/180))*math.Cos(tujuan.Lat*(math.Pi/180))*
28             math.Sin(dLon/2)*math.Sin(dLon/2)
29     c := 2 * math.Atan2(math.Sqrt(a), math.Sqrt(1-a))
30     return jariBumi * c
31 }
```

Gambar 3. Kode logika perhitungan haverse

Berdasarkan eksekusi program dengan titik awal Ciluar Asri, diperoleh hasil perhitungan jarak dan distribusi peluang sebagai berikut:

```
===== HASIL ANALISIS RUTE & PELUANG REKOMENDASI =====
TITIK AWAL : Rumah (Ciluar Asri)
ALAMAT     : Perumahan Ciluar Asri Blok A4 No.3
-----------------------------------------------------
NAMA CABANG                        | JARAK    | PELUANG
-----------------------------------------------------
Kopi Kenangan - Kedunghalang       | 2.45 km | 51.22%
Kopi Kenangan - Pom Bensin Pajajaran | 4.48 km | 28.03%
Kopi Kenangan - Sudirman Bogor     | 6.06 km | 20.75%
-----------------------------------------------------

KESIMPULAN:
Cabang terdekat adalah Kopi Kenangan - Kedunghalang
Estimasi perjalanan: ±6 Menit
RUTE: [RUMAH] -------- [Kopi Kenangan - Kedunghalang]
PS D:\firsttime>
```

Gambar 3. Output analisis rute & peluang direkomendasikan

4. Pembahasan

Berdasarkan berdasarkan Gambar 3., dapat dianalisis bahwa sistem berhasil menerapkan konsep probabilitas diskrit dengan baik. Hal ini dibuktikan dengan:

a. **Dominasi Jarak Dekat**: Cabang Kedung Halang memiliki jarak terdekat (2.45 km) dari rumah pengguna di Ciluar Asri. Dalam model probabilitas, hal ini diterjemahkan menjadi peluang pemilihan sebesar 51.22%. Angka ini menunjukkan bahwa secara statistik, pengguna "sangat mungkin" memilih cabang ini dibandingkan opsi lain.
b. **Distribusi Peluang**: Cabang Pajajaran berjarak 4.48 km (sekitar 2 km lebih jauh dari Kedunghalang). Meskipun jaraknya tidak terlalu jauh berbeda secara visual di peta, model matematika menurunkan peluangnya secara signifikan menjadi 28.03%.
c. **Outlier (Pencilan)**: Cabang Sudirman yang memiliki koordinat di Bogor (jarak 6.06 km) mendapatkan peluang kecil (20.75%). Hal ini membuktikan bahwa algoritma valid dalam mengeliminasi opsi yang tidak relevan bagi pengguna tanpa perlu menghapusnya dari database, melainkan dengan memberikan bobot yang sangat kecil.
d. **Performa Golang**: Dalam pengujian konkurensi sederhana, Golang mampu memproses perhitungan aritmatika geometri dan probabilitas ini dalam waktu rata-rata dibawah 500 mikrodetik, mengkonfirmasi teori efisiensi memori yang dikemukakan oleh Wong & Chek [3].

5. Kesimpulan

Penelitian ini berhasil merancang sistem rekomendasi lokasi berbasis CLI menggunakan Golang. Kesimpulan yang dapat diambil adalah:

1. **Akurasi Geometris:** Formula Haversine efektif menghitung jarak antar koordinat GPS dengan presisi tinggi.
2. **Sistem Pendukung Keputusan:** Penggunaan Probabilitas Diskrit memberikan nilai tambah informatif. Pengguna tidak hanya mengetahui jarak, tetapi juga persentase kelayakan ("*likelihood*") untuk mengunjungi suatu cabang.
3. **Efisiensi:** Implementasi dengan Golang terbukti efisien untuk menangani operasi matematika berulang (*iterative calculations*).

6. Saran Dan Future Work

Untuk pengembangan selanjutnya, disarankan:

1. Mengintegrasikan Google Distance Matrix API untuk mendapatkan data waktu tempuh (*real-time traffic*) sebagai variabel tambahan dalam perhitungan bobot peluang, bukan hanya jarak garis lurus.
2. Menambahkan variabel diskrit lain seperti Rating Bintang atau Promo Tersedia ke dalam rumus probabilitas (Teorema Bayes) untuk rekomendasi yang lebih personal.

Daftar Pustaka

[1] Yunardi, E., Magdalena, L., & Febima, M. (2024). *Journal of Artificial Intelligence and Engineering Applications Implementation of the Haversine Formula Method in Geographic Information Systems for Searching the Nearest Sea Freight Expedition Services in East Jakarta* (Vol. 4, Issue 1).

[3] K. Wong and M. Chek, "Performance Comparison of Haversine and Euclidean Algorithm in Location Based Service," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 5, 2020. https://thesai.org/Downloads/Volume11No5/Paper_67-Performance_Comparison_of_Haversine.pdf

[2] Harismawan, A. F., Putra Kharisma, A., & Afirianto, T. (2018). *Analisis Perbandingan Performa Web Service Menggunakan Bahasa Pemrograman Python, PHP, dan Perl pada Client Berbasis Android* (Vol. 2, Issue 1). http://j-ptiik.ub.ac.id

[4] Ikasari, D., Ikasari, W., & Andika, R. (2021). Implementation of Haversine Formula to Determine the Shortest Path Using Web Based Application for a Case Study of High School Zoning in Depok. *American Journal of Software Engineering and Applications*, *10*(2), 19. https://doi.org/10.11648/j.ajsea.20211002.11

Lampiran

```
package main

import (
    "fmt"
    "math"
    "sort"
)

type Koordinat struct {
    Lat, Lon float64
}

type Lokasi struct {
    Nama         string
    Alamat       string
    Posisi       Koordinat
    Jarak        float64
    Probabilitas float64
}

// Fungsi Haversine untuk menghitung jarak geografis
func itungJarak(asal, tujuan Koordinat) float64 {
    const jariBumi = 6371.0
    dLat := (tujuan.Lat - asal.Lat) * (math.Pi / 180)
    dLon := (tujuan.Lon - asal.Lon) * (math.Pi / 180)
    a := math.Sin(dLat/2)*math.Sin(dLat/2) +

math.Cos(asal.Lat*(math.Pi/180))*math.Cos(tujuan.Lat*(math.Pi/180))*
            math.Sin(dLon/2)*math.Sin(dLon/2)
    c := 2 * math.Atan2(math.Sqrt(a), math.Sqrt(1-a))
    return jariBumi * c
}

func main() {
    rumahSaya := Lokasi{
        Nama:   "Rumah (Ciluar Asri)",
        Alamat: "Perumahan Ciluar Asri Blok A4 No.3",
        Posisi: Koordinat{Lat: -6.5398, Lon: 106.8202},
    }

    daftarToko := []Lokasi{
        {
            Nama:   "Kopi Kenangan - Kedunghalang",
            Alamat: "Jl. Raya Pemda No.1, Kedunghalang",
            Posisi: Koordinat{Lat: -6.5601, Lon: 106.8115},
        },
```

```
        {
            Nama:   "Kopi Kenangan - Pom Bensin Pajajaran",
            Alamat: "Jl. Raya Pajajaran No.127",
            Posisi: Koordinat{Lat: -6.5785, Lon: 106.8088},
        },
        {
            Nama:   "Kopi Kenangan - Sudirman Bogor",
            Alamat: "Jl. Jend. Sudirman No.24",
            Posisi: Koordinat{Lat: -6.5892, Lon: 106.7971},
        },
    }

    var totalBobot float64


    for i := range daftarToko {
        daftarToko[i].Jarak = itungJarak(rumahSaya.Posisi,
daftarToko[i].Posisi)

        // Rumus: Semakin kecil jarak, semakin besar peluang
        daftarToko[i].Probabilitas = 1.0 / daftarToko[i].Jarak
        totalBobot += daftarToko[i].Probabilitas
    }

    for i := range daftarToko {
        daftarToko[i].Probabilitas = (daftarToko[i].Probabilitas /
totalBobot) * 100
    }

    sort.Slice(daftarToko, func(i, j int) bool {
        return daftarToko[i].Jarak < daftarToko[j].Jarak
    })

    fmt.Println("======= HASIL ANALISIS RUTE & PELUANG REKOMENDASI
=======")
    fmt.Printf("TITIK AWAL : %s\n", rumahSaya.Nama)
    fmt.Printf("ALAMAT     : %s\n", rumahSaya.Alamat)
    fmt.Println("-----------------------------------------------------")
    fmt.Printf("%-35s | %-8s | %-10s\n", "NAMA CABANG", "JARAK",
"PELUANG")
    fmt.Println("-----------------------------------------------------")

    for _, toko := range daftarToko {
        fmt.Printf("%-35s | %4.2f km | %5.2f%%\n", toko.Nama, toko.Jarak,
toko.Probabilitas)
    }

    fmt.Println("=================================================")

    // Kesimpulan
    pilihan := daftarToko[0]
    menit := (pilihan.Jarak / 25) * 60
```

```
    fmt.Printf("\nKESIMPULAN:")
    fmt.Printf("\nCabang terdekat adalah %s", pilihan.Nama)
    fmt.Printf("\nEstimasi perjalanan: ±%.0f Menit", menit)
    fmt.Printf("\nRUTE: [RUMAH] -------- ☕ [%s]\n", pilihan.Nama)
}
```

Lampiran 1. Full Main Code