REKOMENDASI *LANDMARK* PADA OBJEK BERGERAK MENGGUNAKAN METODE ANALISIS *DATAFRAME*

Muhammad Haikal Ilyasa[*)], Aghus Sofwan dan Yosua Alvin Adi Soetrisno

Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: lakiah23mh@students.undip.ac.id*

Abstrak

Di era perkembangan industri 4.0, data menjadi elemen terpenting dalam penggerak industri. Setiap data dapat menjadi nilai jual tersendiri karena terdapat informasi penting didalamnya. Salah satu data yang banyak dicari dan mudah dianalisa adalah data geospatial atau data GPS(Global Positioning System). Perpaduan antara GPS dan smartphone banyak digunakan oleh masyarakat untuk melacak dan menampilkan data posisi transportasi. Dengan menggunakan analisis *dataframe* dan data geospasial seseorang dapat mengetahui informasi yang diinginkan misalnya adalah rekomendasi dari landmark yang sering dikunjungi ataupun bekas jejak posisi dari GPS smartphone pengguna. Dengan demikian pengguna hasil analisis data dapat mendapatkan rekomendasi tempat atau landmark yang sering dikunjungi oleh banyak orang.

Kata kunci: data, data geospasial, GPS (*Global Positioning System*), *dataframe*, rekomendasi, *landmark* .

Abstract

In the era of industrial development 4.0, data is the most important element in driving the industry. Each data can be a selling point in itself because there is important information in it. One of the most sought-after and easy to analyze data is geospatial data or GPS (Global Positioning System) data. The combination of GPS and smartphones is widely used by the public to track and display transportation position data. By using dataframe analysis and geospatial data, one can find out the desired information, for example, is a recommendation from a frequently visited landmark or a trace position from the user's smartphone GPS. Thus, users of data analysis results can get recommendations for places or landmarks that are frequently visited by many people.

Keywords: data, geospatial data, GPS (Global Positioning System), dataframe, recommendation, landmark.

1. Pendahuluan

Di era globalisasi digital saat ini, data memiliki peran yang sangat penting untuk menentukan arah perkembangan suatu industri. Terutama adalah data geospasial yang terdapat data posisi dari GPS(*Global Positioning System*). Dengan mengolah data tersebut, seseorang dapat membuat sebuah rekomendasi lokasi tertentu untuk memberikan kesan yang penting ketika berkunjung atau mencari tempat. Dengan menggunakan hasil analisis data tersebut seseorang dapat menemukan lokasi yang sering dikunjungi dengan mengaplikasikan *geofencing* pada luas bangunan[1]. Perusahaan maupun perorangan dapat menggunakan hasil analisis data ini untuk mengembangkan usahanya sehingga mereka dapat menentukan lokasi terbaik untuk mengembangkan usaha mereka. Meningkatnya persaingan bisnis kecil maupun besar mendorong adanya kebutuhan data guna menemukan Langkah yang terbaik untuk mengembangkan bisnis, hal inilah yang mendorong pentingnya analisis data dan data sains[2].

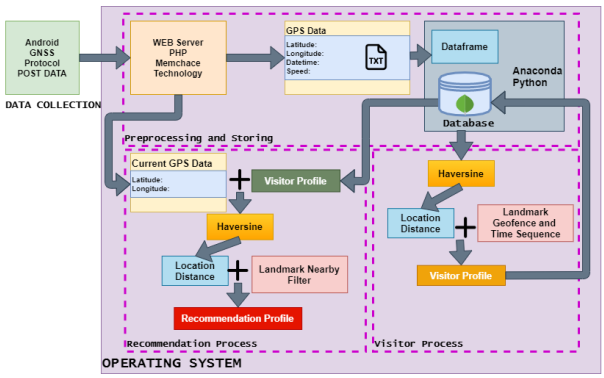
Analisa *dataframe* pada data geospasial akan menyediakan hasil analisa untuk menentukan lokasi *landmark* yang sering dikunjungi dan seseorang dapat menemukan tempat berkunjung baru sehingga perusahaan dan perorangan dapat menentukan yang membutuhkan informasi tersebut dapat menggunakannya dengan bijak[3]. Sistem ini memberi tahu penggunanya dimana lokasi *landmark*, frekuensi kunjungan, dan histori lintasan dengan menggunakan data geospatial posisi dan waktu geografis dari GPS *Smartphone*. Sistem ini terdiri dari kumpulan kode analisis dengan Bahasa pemrograman Python dan *library* data sains yang dimuat dalam bentuk file iPythonNotebook(ipynb)[4]. Keluaran dari sistem ini adalah file berupa rekomendasi lokasi dan frekuensi kunjungan *landmark*[5].

2. Metode

2.1. Deskripsi Sistem

Dari segi server, sistem rekomendasi *landmark* dibangun dengan menggunakan bahasa python. Server didesain untuk mengelola data mentah dari Android dan Webserver

kemudian dilakukan sejumlah proses mulai dari parsing data[6] dan penyimpanan hingga data diproses kedalam dua proses pengolahan *dataframe*[7] hingga pada akhirnya sistem dapat memberikan sebuah rekomendasi *landmark*. Ilustrasi desain sistem rekomendasi *landmark*[8] dapat dilihat pada gambar 1.



Gambar 1. Desain Server

Pada gambar 1 dapat dilihat bahwa server didesain server untuk melalui 4 tahap pengolahan, yaitu parsing data dan penyimpanan, proses pengujung, dan proses rekomendasi[9]. Anaconda Distribution berfungsi sebagai manajer paket untuk mempermudah bahasa python mengatur paket *library* yang digunakan serta memudahkan sambungan pada Database mongodb[10][15] yang akan digunakan sebagai media penyimpanan.

2.2. Analisis Kebutuhan

2.2.1. Kebutuhan Fungsional

Kebutuhan fungsional merupakan gambaran mengenai fungsi-fungsi yang dapat dilakukan oleh sistem ini. Kebutuhan fungsional sistem meliputi:

1) Pengguna (admin) dapat menemukan *landmark* dengan pengunjung terbanyak.
2) Pengguna (admin) dapat menerima lokasi *landmark* yang direkomendasikan dengan memasukan data lokasi.
3) Pengguna (admin) dapat menyimpan data pengunjung *landmark* pada database sehingga dapat digunakan oleh media aplikasi.
4) Sistem dapat memberikan petunjuk lokasi *landmark* yang direkomendasikan melalui *Dataframe*.

2.2.2. Kebutuhan Non Fungsional

Kebutuhan non-fungsional adalah kebutuhan sistem meliputi kinerja, kelengkapan operasi pada fungsi-fungsi yang ada, serta kesesuaian dengan lingkungan penggunanya. Kebutuhan non-fungsional ini melingkupi beberapa kebutuhan yang mendukung kebutuhan fungsional, rumusan kebutuhan non-fungsional meliputi:

1) Kebutuhan Operasional
   - Sistem dapat bekerja dengan baik pada sistem operasi Ubuntu 16.04 LTS dengan RAM minimal 4GB dan Windows10 dengan RAM minimal 8GB.
   - Sistem hanya dapat diakses oleh pemilik perusahaan aplikasi terkait.
   - Sistem ini dibangun dengan menggunakan bahasa pemrograman Python 3.
   - Sistem dilengkapi dengan *library* untuk meminimalkan dokumentasi dan mempermudah proses pengerjaan.
   - *Library* harus menyesuaikan versi Python dan kebutuhan *library* umum yang diinstal.
   - Sistem ini menggunakan *database* MongoDB sebagai pengelola data.

2) Performa Sistem
   Sistem yang dibangun merupakan sistem rekomendasi yang berjalan pada lingkungan server. Terdapat beberapa keterbatasan yang ditemui pada perangkat server. Oleh karena itu perlu diperhatikan guna menjadi acuan dalam pengembangan sistem, diantaranya:
   - Fisik server memiliki keterbatasan sumber daya yang dapat mempengaruhi kinerja sistem sehingga perlu diperhatikan sumber daya yang tersedia pada fisik server.
   - Kinerja sistem saat dipakai bersamaan dapat menimbulkan beban yang signifikan pada server yang dapat menyebabkan sistem terhenti secara tiba-tiba.

Dari keterbatasan pada perangkat server tersebut, maka diusulkan beberapa alternatif untuk menunjang performa sistem rekomendasi *landmark* dengan keterbatasan yang ada, diantaranya:

- Merancang server dengan pemanfaatan sumber daya seefisien mungkin namun tidak mengurangi fungsi dan performa aplikasi.
- Merancang perulangan dan dokumentasi kode perulangan yang mudah dipahami untuk memaksimalkan hasil serta mendapatkan hasil *output* program yang sesuai.

2.2.3. Kebutuhan Perangkat Keras

Dalam pembuatan sistem rekomendasi ini, dibutuhkan beberapa spesifikasi perangkat keras. Spesifikasi perangkat keras tersebut dapat dimasukkan ke dalam kebutuhan perangkat keras dalam analisis kebutuhan. Karena menggunakan Anaconda Distribution, Jupyter Notebook beserta *library* Data Sains[5][12], perangkat keras yang dibutuhkan dalam membuat aplikasi ini adalah sebuah komputer server dan VPS dengan spesifikasi yang ditunjukkan pada tabel 1.

Tabel 1. Implementasi kebutuhan perangkat keras

Spesifikasi	Keterangan
Processor	Intel(R) Core(TM) i7-4500U CPU
RAM	8192 MB
VGA	Nvidia GeForce 635M
Harddisk	500 GB
Seri Model	ASUS UX302LG

2.2.4. Kebutuhan Perangkat Lunak

Dalam pembuatan aplikasi ini, dibutuhkan beberapa spesifikasi perangkat lunak[9]. Spesifikasi perangkat lunak tersebut dapat dimasukkan ke dalam kebutuhan perangkat lunak dalam analisis kebutuhan. Perangkat lunak yang dibutuhkan baik untuk merancang sistem, membuat sistem maupun menjalankan sistem adalah seperti yang ditunjukkan pada tabel 2.

Tabel 2. Implementasi kebutuhan perangkat lunak

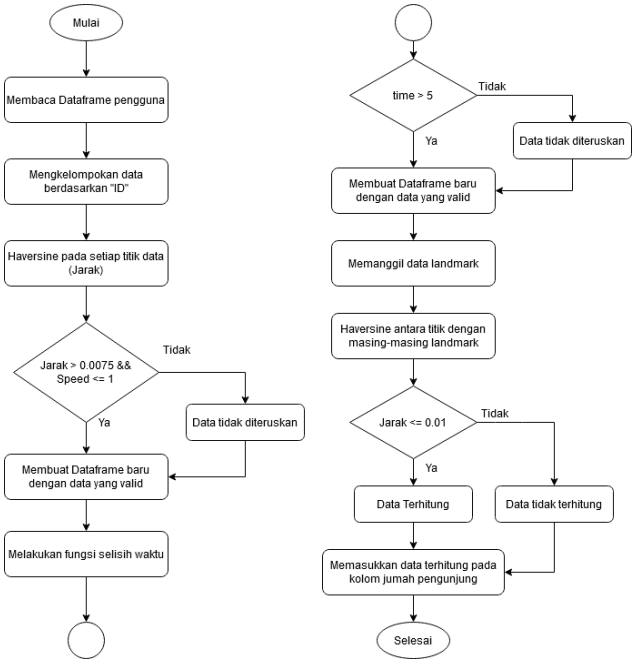
Spesifikasi	Keterangan
Sistem Operasi	Windows 10 Pro 64 bit / Linux Ubuntu 16.04
Bahasa Pemrograman	python3
Teks Editor	Notepad++, Sublime Text Editor
Package Manager	Anaconda Distribution
IDE	Jupyter Notebook
Library	numpy, pandas, matplotlib
MongoDB	MongoDB v4.1.1 , MongoDBCompass

2.3. Desain Sistem Rekomendasi *Landmark*

Desain sistem merupakan suatu proses perencanaan sebuah sistem yang berjalan dan merupakan tahap lanjutan dari analisa sistem, dimana pada perancangan sistem digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean kedalam suatu bahasa pemrograman[5][13]. Tujuan dari perancangan sistem secara global adalah membentuk kerangka sistem pengolahan data dan divisualkan melalui diagram alir[7]. Rancangan sistem yang baik akan diterapkan suatu kejadian untuk menentukan dan mengembangkan metode prosedur dan proses suatu data agar tujuan dari suatu organisasi dapat tercapai, sedangkan tujuan dari perancangan sistem secara umum adalah memberikan gambaran secara umum kepada pengguna mengenai sistem yang akan dirancang melalui diagram alirnya[8].

2.3.1. Profil Pengunjung *Landmark*

Bagian ini merupakan proses awal pada sistem rekomendasi *landmark.* Pada proses ini data pengguna akan melewati sejumlah bagian proses guna mencari data jumlah pengunjung *landmark*[5][13]. Pada bagian ini juga data mentah akan disaring dan dikelompokkan sebelum melewati proses perhitungan pengunjung *landmark.* Proses perhitungan pengunjung *landmark* ini melibatkan fungsi *haversine* dan fungsi selisih waktu untuk menentukan data yang valid pada proses penentuan *landmark*[6][14]. Diagram alir profil pengunjung *landmark* dapat dilihat pada gambar 2.



Gambar 2. Diagram Alir Analisis Jumlah Kunjungan *Landmark*

Dari gambar 2 dapat dilihat diagram alir proses mencari jumlah pengunjung *landmark.* Mula-mula sistem akan membaca *dataframe* oleh data pengguna lalu mengelompokkan data tersebut berdasarkan kolom ID pengguna. Kemudian data tersebut akan melakukan fungsi *haversine* pada setiap titik sebelum dan sesudahnya. Data yang valid adalah data yang memiliki batas jarak 0,0075km. Setelah *dataframe* mentah di*filter* berdasarkan kriteria yang memenuhi, selanjutnya *dataframe* akan melewati 2 proses yaitu mencari selisih waktu dan perhitungan *landmark.* Kriteria data yang bisa dianggap bahwa data telah singgah atau mengunjungi tempat tersebut apabila ia berdiam disana selama 5 detik. Setelah melalui proses pemisahan data berdasarkan selisih waktu, data akan melalui proses perhitungan jumlah pengunjung. Mula-mula sistem akan memanggil data *landmark.* Kemudian, data yang telah melalui proses selisih waktu akan dilakukan fungsi *haversine* pada setiap titik dengan data *landmark.* Pada proses ini sistem melakukan *geofencing* terhadap titik-titik data dan menghitung setiap titik apabila titik tersebut berjarak kurang dari 0.01 km dari landmark maka penghitung akan menghitung 1 (satu) pengunjung.

2.3.2. Proses Rekomendasi *Landmark*

Proses rekomendasi menunjukan langkah-langkah yang ditempuh sistem untuk menemukan rekomendasi dari titik input yang merupakan data geospasial[9][10]. Bagian ini adalah tahap akhir dari proses rekomendasi *landmark.*

Secara garis besar algoritmanya adalah menginput sebuah titik kemudian menghitung jarak posisi titik terhadap *landmark* dan mengurutkannya berdasarkan jarak terdekat dan jumlah pengunjungnya. Diagram alir proses rekomendasi *landmark* dapat dilihat pada gambar 3.

Gambar 3. Diagram Alir Proses Rekomendasi *landmark*

Dapat dilihat dari gambar 3 sistem menangkap sebuah input yang terdiri atas data GPS berupa *latitude* dan *longitude*. Kemudian sistema memanggil data *landmark* profil pengunjung dari proses sebelumnya. Sistem akan menggunakan fungsi Haversine dari titk input *landmark* untuk menghitung jarak keduanya dan mencatat jaraknya pada kolom baru. Sistem akan melakukan perulangan hingga semua titik *landmark* telah melalui fungsi *haversine* dan mendapatkan jarak dari posisi ke *landmark*. Langkah selanjutnya adalah mensortir data pada jarak terdekat pada *landmark* dan menunjukkan jumlah pengunjungnya. Kemudian sistem merilis sebuah *dataframe* yang berisi tabel rekomendasi dan selesailah seluruh bagian proses rekomendasi *landmark*.

3. Hasil dan Analisis

3.1. Proses Parsing Data

Pada implementasi parsing data ini menggambarkan bagaimana program parsing data berjalan dari awal mulai hingga akhir. Program ini dijalankan dengan tujuan untuk memperbaiki format data dan memisahkan data yang kosong (NaN) pada *dataframe* yang bisa menyebabkan terjadinya kegagalan dalam menjalankan program[11][12]. Program awalnya akan memanggil database data asli yang didapatkan dari GPS. Data asli pada gambar 4 dibawah meliputi data pengguna aplikasi yang menghidupkan fitur GPS pada *smartphone.* Data yang terkirim terdiri dari data geospasial (*longitude* & *latitude*), data waktu dan tanggal (*Datetime*), nomor id, nama pengguna, dan kecepatan (*Speed*).

```
sys:1: DtypeWarning: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
   Latitude  Longitude             Datetime   ID   Name     Speed
0  -7.05748  110.44104   07/21/2020 11:16:27   16   boim  1.126032
1  -7.05748  110.44104   07/21/2020 11:16:29   16   boim  1.549194
2  -7.05752  110.44102   07/21/2020 11:16:31   16   boim  0.403623
3  -7.05753  110.44103   07/21/2020 11:16:34   16   boim  0.559931
4  -7.05753  110.44104   07/21/2020 11:16:35   16   boim  0.559931
```

Gambar 4. *dataframe* data pengguna

Pada proses parsing data sistem akan mengkonversi kolom ID yang bertipe data campuran menjadi tipe data angka numerik. Sistem akan mengubah ID menjadi bilangan real positif. Kemudian program juga akan mengganti ID yang tidak terdefinisi angka menjadi NaN (*Not a Number*). Berikut hasil plot informasi *dataframe* pengguna sebelum dan sesudah dilakukan proses *parsing* data:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 540635 entries, 0 to 540634
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Latitude   540635 non-null  float64
 1   Longitude  540635 non-null  float64
 2   Datetime   540635 non-null  object
 3   ID         540635 non-null  object
 4   Name       540635 non-null  object
 5   Speed      540635 non-null  float64
dtypes: float64(3), object(3)
memory usage: 24.7+ MB
```

Gambar 5. data-u sebelum *parsing*

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 540633 entries, 0 to 540634
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Latitude   540633 non-null  float64
 1   Longitude  540633 non-null  float64
 2   Datetime   540633 non-null  object
 3   ID         540633 non-null  float64
 4   Name       540633 non-null  object
 5   Speed      540633 non-null  float64
dtypes: float64(4), object(2)
memory usage: 28.9+ MB
```

Gambar 6. data-u sesudah *parsing*

Dari gambar 5 dan 6 diatas adalah keadaan sebelum dan sesudah melalui proses parsing data. Dapat kita lihat sebelumnya pada gambar 5 dimana kolom ID yang awalnya bertipe data *object* berubah menjadi *float64* pada gambar 6. Kemudian total data mengalami perubahan dari data awal yaitu 540.635 data menjadi 540.633 data (2 data invalid). Praproses juga mempengaruhi ukuran akhir dari 24.7 MB menjadi 28.9 MB, hal ini dikarenakan perubahan

dari tipe data *string* menjadi *float64*. Dengan demikian seluruh kode telah berhasil dijalankan dan hasil dari parsing data telah dilalui dan dapat diteruskan ke testing proses selanjutnya yaitu proses pengunjung *landmark.*

3.2. Program Pengunjung *Landmark*

Untuk memvalidasi program sistem, pengujian validasi *White-Box* dilakukan dengan melakukan validasi pada desain algoritma sistem rekomendasi[13]. Sesuai dengan tujuan program, algoritma penghitung pengunjung dibuat untuk memberikan informasi berupa jumlah pengunjung pada *landmark* yang terdapat pada database. Sistem ini menggunakan banyak proses *filter* yaitu berupa implementasi fungsi *haversine* serta mempertimbangan selisih waktu untuk memfilter titik-titik user dan mengimplementasikan *geofence* untuk menghitung jumlah pengunjung.

Untuk mengevaluasi langkah-langkah algoritma terlebih dahulu menentukan target serta cara kerjanya. Demikian adalah poin-poin yang harus dievaluasi dalam Program sistem rekomendasi :

- Program berhasil menangkap titik-titik (data) *user* dari *database*.
- Program berhasil menangkap titik-titik (data) *landmark* dari *database*.
- Program berhasil mengelompokkan data user berdasarkan ID.
- Program berhasil menerapkan fungsi *haversine* pada titik-titk user untuk memfilter jarak pada data user ID sama yang berdekatanuntuk mengidentifikasi apabila sudah beda *landmark* (e > 0.0075 km).
- Program berhasil mem-*filter* data user berdasarkan durasi kunjungan (d > 5 menit).
- Program berhasil melakukan *geofence* dengan menggunakan fungsi haversine untuk menemukan jumlah pengujungnya.

Berikut adalah tabel hasil *geofencing dataframe landmark* pada data yang sudah melalui parsing data dan filterisasi. Data profil pengunjung *landmark* dapat dilihat pada Tabel 3 berikut ini.

Tabel 3. Profil Pengunjung *Landmark*

Landmark	*visit_count*
0.0	22
1.0	2
4.0	3
5.0	6
6.0	3
7.0	4
8.0	2
11.0	3
12.0	1
13.0	5

Dari sejumlah hasil daripada program dan juga melalui tabel 3 dapat ditarik kesimpulan melalui uji validasi algoritma rekomendasi *landmark*. Program terbukti dapat menangkap data user dan *landmark* dari *database*, kemudian mengelompokkan data berdasarkan ID. Program juga berhasil melakukan filterisasi pada *dataframe* untuk mendapatkan data yang valid lalu dapat melakukan proses *geofencing* yang dibuktikan pada tabel profil pengunjung *landmark* . Sehingga tabel 4 berikut ini.

Tabel 4. Validasi sistem pengunjung *landmark*

No.	Uji Validasi	Keterangan
1	Menangkap titik-titik (data) *user* dari *database*.	Berhasil
2	Menangkap titik-titik *landmark* dari database.	Berhasil
3	Mengelompokkan data user ber-dasarkan ID.	Berhasil
4	Menerapkan fungsi *haversine* pada titik-titk user untuk memf-ilter jarak pada data user ID (e > 0.0075 km).	Berhasil
5	Mem-*filter* data user berdasarka-n durasi kunjungan (d > 5 menit).	Berhasil
6	Melakukan *geofence* dengan menggunakan fungsi *haversine* untuk menemukan jumlah peng-ujungnya.	Berhasil

Dapat dilihat dari tabel diatas program dinyatakan berhasil sehingga dapat dikatakan program sudah sesuai dengan rancangan algoritma dan dapat segera diimplementasikan pada sistem penghitung pengunjung *landmark* serta data output dapat digunakan pada proses selanjutnya.

3.3. Program Rekomendasi *Landmark*

Pengujian validasi *White-Box* dilakukan dengan melakukan pengujian pada desain algoritma sistem rekomendasi[15]. Pengujian algoritma sistem bertujuan untuk mevalidasi sistem rekomendasi telah bekerja sesuai dengan tujuan. Sesuai dengan tujuan program, algoritma sistem rekomendasi dibuat untuk memberikan rekomendasi pada *user*. Dalam hal ini algoritma rekomendasi *landmark* mempertimbangkan jumlah pengunjungnya dan jaraknya[16]. Untuk mengevaluasi langkah-langkah algoritma terlebih dahulu menentukan target serta cara kerjanya. Demikian adalah poin-poin yang harus dievaluasi dalam program sistem rekomendasi :

- Program berhasil menangkap titik-titik *landmark* dari *database*.
- Program berhasil menangkap titik peminta rekomendasi.
- Program berhasil menerapkan fungsi *haversine* pada titik-titik *landmark* untuk menentukan variabel jarak terdekat pada *landmark* rekomendasi.
- Program berhasil mempertimbangkan variabel jumlah pengunjung terbanyak untuk memberikan rekomendasi landmark.
- Program berhasil mem-*filter* dan mengurutkan rekomendasi landmark berdasarkan jarak terdekat dan jumlah pengunjungnya.

Pada Tugas Akhir ini dilakukan pengujian dengan mencoba algoritma rekomendasi pada 5 jenis input berupa

titik lokasi yang dipilih untuk mevalidasi hasil. Berikut kelima titik uji tersebut adalah berikut ini.

- Lokasi 1 : Kos Griya Sipodang (-7.056782, 110.436396)
- Lokasi 2 : Kos Gang Arjuna 6 (-7.057549, 110.442805)
- Lokasi 3 : Kampus Departemen Teknik Elektro (-7.049541, 110.440040)
- Lokasi 4 : Kos Graha Oriza (-7.057837, 110.441143)
- Lokasi 5 : Kos Banjarsari 35B (-7.060152, 110.439256)

Rumus *Haversine* :

$$2R.sin^{-1}\left(\sqrt{sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1).\cos(lat_2).sin^2\left(\frac{long_2 - long_1}{2}\right)}\right) \quad [14](1)$$

Dimana :

R = Jari jari bumi ≈ 63711 *km*
Lat 1 = *Latitude* titik uji
long1 = *Longitude* titik *landmark*
lat2 = *Latitude* titik uji
long2 = *Longitude* titik *landmark*

Perhitungan jarak pada :

- Titik Lokasi 1 (lat1, long1) : (-7.056782, 110.436396)
- Titik *Landmark* 0 (lat2, long2) : (-7.056973, 110.435308)

Sehingga dengan menerapkan persamaan (1) :

$$2 \times 63711.sin^{-1}\left(\sqrt{sin^2\left(\frac{-7.056973 - (-7.056782)}{2}\right) + \cos(-7.056782).\cos(-7.056973).sin^2\left(\frac{110.435308 - 110.436396}{2}\right)}\right)$$

Hasilnya : $= \mathbf{0.8293669407548993\ km}$

Tabel 5. Hasil Eksekusi Fungsi *Haversine*

Nama *Landmark*	*Latitude*	*Longitude*	Jarak (km)
Markoni Cak Eco	-7.056	110.435	0.82936
Nagoya Ramen & Sushi	-7.057	110.435	0.79407
Depot Sukses Tembalang	-7.058	110.434	0.87512
Calm Thai Tea Tembalang	-7.06	110.442	0.29588
Chicken Fighter Tembalang	-7.061	110.435	0.90544
Padang Murah Banjarsari Tembalang	-7.059	110.440	0.34067
Padang Murah Tembalang	-7.054	110.430	1.3699
Street Food Tembalang	-7.058	110.435	0.8339
Rumah Makan Padang Nusantara	-7.051	110.443	0.6887
The Salt Resto	-7.061	110.435	0.9175
Ayam Geprek Djogjakarta LPPU	-7.056	110.439	0.3868
Warung Makan Bos Gentong	-7.062	110.436	0.8311

Dengan mengeksekusi fungsi *haversine* terhadap *dataframe landmark* maka didapatkan hasil perhitungan jarak antara titik dan *landmark* pada tabel 5.

Kemudian sistem akan mem-*filter landmark* berdasarkan jarak dan jumlah pengunjung terbesar. Berikut *output dataframe landmark* rekomendasi dari uji titik Lokasi 1 hingga Lokasi 5 dapat dilihat pada gambar 7-11.

```
INPUT (lat/long): -7.057549365880778 110.44280587839107
Data Terhitung : 201
   index  Landmark  Latitude   Longitude                Nama_Tempat  Visit_Count  Distance
0    196       196 -7.057557  110.442797  Kost Putra Griya Sipodang          7.0  0.001296
1    194       194 -7.057092  110.443366          Cilor Milor Maranti          7.0  0.079993
2    198       198 -7.057861  110.441133                Graha Oriza          5.0  0.187712
3    197       197 -7.060123  110.439273          Kost Banjarsari 35B          6.0  0.483316
4    187       187 -7.056004  110.438544                Smooly Juice          7.0  0.500404
5     72        72 -7.060698  110.447004           OTI Fried Chicken          5.0  0.580324
6     57        57 -7.062562  110.441209        Nasi Kuning Pasundan 1          6.0  0.584205
7    195       195 -7.056802  110.436539                Kos Arjuna 6          8.0  0.696104
8     13        13 -7.059810  110.435675  Kembang Joyo Angkringan & Cafe         5.0  0.825563
9      0         0 -7.056973  110.435308              Markoni Cak Eco         22.0  0.829367
10    21        21 -7.055729  110.435316  MIE AYAM & BAKSO SALATIGA          6.0  0.850418
```

Gambar 7. Rekomendasi *Landmark* Lokasi 1

```
INPUT (lat/long): -7.056782 110.436396
Data Terhitung : 201
    index  Landmark  Latitude   Longitude                Nama_Tempat  Visit_Count  Distance
0     195       195 -7.056802  110.436539                Kos Arjuna 6          8.0  0.015926
1       0         0 -7.056973  110.435308              Markoni Cak Eco         22.0  0.121851
2      21        21 -7.055729  110.435316  MIE AYAM & BAKSO SALATIGA          6.0  0.166969
3     187       187 -7.056004  110.438544                Smooly Juice          7.0  0.252172
4      13        13 -7.059810  110.435675  Kembang Joyo Angkringan & Cafe         5.0  0.345754
5      15        15 -7.055971  110.433223             Penyet Vandem xm3          8.0  0.361348
6     197       197 -7.060123  110.439273          Kost Banjarsari 35B          6.0  0.488375
7     198       198 -7.057861  110.441133                Graha Oriza          5.0  0.535995
8     189       189 -7.054325  110.438782             KFC PALTROW             7.0  0.684745
9     196       196 -7.057557  110.442797  Kost Putra Griya Sipodang          7.0  0.711157
10    194       194 -7.057092  110.443366          Cilor Milor Maranti          7.0  0.769446
11     57        57 -7.062562  110.441209        Nasi Kuning Pasundan 1          6.0  0.833242
12     68        68 -7.052833  110.428999             Soto Ayam Kabul          6.0  0.926312
```

Gambar 8. Rekomendasi *Landmark* Lokasi 2

```
INPUT (lat/long): -7.0495413538904454 110.44004042030004
Data Terhitung : 201
   index  Landmark  Latitude   Longitude                Nama_Tempat  Visit_Count  Distance
0    187       187 -7.056004  110.438544                Smooly Juice          7.0  0.736880
1     21        21 -7.055729  110.435316  MIE AYAM & BAKSO SALATIGA          6.0  0.862711
2    195       195 -7.056802  110.436539                Kos Arjuna 6          8.0  0.894485
3    194       194 -7.057092  110.443366          Cilor Milor Maranti          7.0  0.915721
4    198       198 -7.057861  110.441133                Graha Oriza          5.0  0.932341
5    196       196 -7.057557  110.442797  Kost Putra Griya Sipodang          7.0  0.941189
6      0         0 -7.056973  110.435308              Markoni Cak Eco         22.0  0.976938
```

Gambar 9. Rekomendasi *Landmark* Lokasi 3

```
INPUT (lat/long): -7.05783734106022 110.44114355124218
Data Terhitung : 201
   index  Landmark  Latitude   Longitude                Nama_Tempat  Visit_Count  Distance
0    198       198 -7.057861  110.441133                Graha Oriza          5.0  0.002875
1    196       196 -7.057557  110.442797  Kost Putra Griya Sipodang          7.0  0.184989
2    194       194 -7.057092  110.443366          Cilor Milor Maranti          7.0  0.258715
3    197       197 -7.060123  110.439273          Kost Banjarsari 35B          6.0  0.327213
4    187       187 -7.056004  110.438544                Smooly Juice          7.0  0.351704
5    195       195 -7.056802  110.436539                Kos Arjuna 6          8.0  0.520675
6     57        57 -7.062562  110.441209        Nasi Kuning Pasundan 1          6.0  0.525078
7     13        13 -7.059810  110.435675  Kembang Joyo Angkringan & Cafe         5.0  0.641692
8      0         0 -7.056973  110.435308              Markoni Cak Eco         22.0  0.650691
9     21        21 -7.055729  110.435316  MIE AYAM & BAKSO SALATIGA          6.0  0.684855
10    72        72 -7.060698  110.447004           OTI Fried Chicken          5.0  0.720255
11    15        15 -7.055971  110.433223             Penyet Vandem xm3          8.0  0.897788
```

Gambar 10. Rekomendasi *Landmark* Lokasi 4

```
INPUT (lat/long): -7.060152071609383 110.43925663534353
Data Terhitung : 201
   index  Landmark  Latitude   Longitude                Nama_Tempat  Visit_Count  Distance
0    197       197 -7.060123  110.439273          Kost Banjarsari 35B          6.0  0.003701
1    198       198 -7.057861  110.441133                Graha Oriza          5.0  0.328084
2     57        57 -7.062562  110.441209        Nasi Kuning Pasundan 1          6.0  0.343624
3     13        13 -7.059810  110.435675  Kembang Joyo Angkringan & Cafe         5.0  0.396817
4    187       187 -7.056004  110.438544                Smooly Juice          7.0  0.467607
5    195       195 -7.056802  110.436539                Kos Arjuna 6          8.0  0.477928
6    196       196 -7.057557  110.442797  Kost Putra Griya Sipodang          7.0  0.485393
7      0         0 -7.056973  110.435308              Markoni Cak Eco         22.0  0.560745
8    194       194 -7.057092  110.443366          Cilor Milor Maranti          7.0  0.566585
9     21        21 -7.055729  110.435316  MIE AYAM & BAKSO SALATIGA          6.0  0.656087
10    15        15 -7.055971  110.433223             Penyet Vandem xm3          8.0  0.811567
11    72        72 -7.060698  110.447004           OTI Fried Chicken          5.0  0.856550
```

Gambar 11. Rekomendasi *Landmark* Lokasi 5

Dari hasil test ke-5 uji titik lokasi, program telah berhasil menampilkan rekomendasi *landmark* yang berbeda-beda dan mengurutkan kelimanya berdasarkan jarak terdekatnya. Berdasarkan ke-5 uji titik rekomendasi didapatkan *landmark* yang dirangkum berdasarkan *landmark* yang sering direkomendasikan beserta data pengunjungnya pada tabel 6 .

Tabel 6. *Landmark* Rekomendasi Uji Titik Lokasi

Nama Landmark	Pengunjung	Rekomendasi
Markoni Cak Eco	22	5
Kos Arjuna 6	8	5
Kost Putra Griya Sipodang	7	5
Cilor Milor Maranti	7	5
Smooly Juice	7	5
Kos Graha Oriza	5	5
MIE AYAM & BAKSO SALATIGA	6	5
Kost Banjarsari 35B	6	4
Kembang Joyo Angkringan & Café	5	4
Nasi Kuning Pasundan 1	1	4
Penyet Vandem xm3	8	3
OTI Fried Chicken	5	3
KFC PALTROW	7	1
Soto Ayam Kabul	6	1

Tabel 7. Hasil uji validasi sistem rekomendasi *landmark*

No.	Uji Validasi	Keterangan
1	menangkap titik-titik *landmark* dari database.	Berhasil
2	menangkap titik peminta rekomendasi.	Berhasil
3	menerapkan fungsi *haversine* pada titik-titik *landmark* untuk menentukan variabel jarak terdekat pada *landmark* rekome-ndasi.	Berhasil
4	mempertimbangkan variabel jumlah pengunjung terbanyak untuk memberikan rekomendasi *landmark*.	Berhasil
5	Mem-*filter* dan mengurutkan rekomendasi *landmark* berdasar-kan jarak terdekat dan jumlah pengunjungnya.	Berhasil

Dari tabel 6 dapat kita lihat secara seksama *landmark-landmark* yang direkomendasikan oleh sistem terhadap ke-5 titik uji. Dari 201 landmark sistem merekomendasikan 5 kali landmark MIE AYAM & BAKSO SALATIGA, Markoni Cak Eco, Kos Arjuna 6, Smooly Juice, Kos Graha Oriza, Cilor Milor Maranti, dan Kost Putra Griya Sipodang.

Jika diperhatikan dengan seksama ketujuh landmark yang sering direkomendasikan memiliki jumlah pengunjung lebih dari sama dengan 5 (lima) sedangkan landmark Nasi Kuning Pasundan 1 yang memiliki jumlah pengunjung 1 (satu) yang hanya direkomendasikan 4 kali. Hal ini membuktikan bahwa program berhasil menentukan *landmark* rekomendasi berdasarkan jarak terdekat dan jumlah pengunjungnya.

Dapat dilihat dari tabel 7 diatas sistem berhasil dalam semua kategori uji validasi sehingga dapat kita simpulkan bahwa sistem telah memenuhi standar validasi algoritma dan program dapat segera diimplementasikan pada sistem rekomendasi *landmark.*

4. Kesimpulan

Kesimpulan yang dapat diambil dalam membuat sistem Rekomendasi *Landmark* pada Objek Bergerak dari tugas akhir ini antara lain adalah sistem pemberi rekomendasi terdiri atas 3 tahap, yaitu proses *parsing* data lalu sistem penghitung pengunjung *landmark* dan sistem pemberi rekomendasi. Pertama data asli pengguna harus melewati proses *parsing* data karena pada tipe data ID tidak terdeteksi sebagai data numerik alias terdeteksi sebagai *object* sehingga harus diubah. Pada sistem penghitung jumlah pengunjung program melakukan proses filter 1 dan filter untuk menghilangkan titik-titik yang berdekatan serta membedakan aktifitas dengan mengimplementasikan fungsi *haversine* dan menghitung selisih waktunya. Pada pengujian uji titik sistem rekomendasi *landmark* menampilkan *landmark* yang berbeda-beda karena sistem berhasil mengurutkan jarak terdekatnya serta mereko-mendasikan *landmark* berdasarkan jumlah pengunjung *landmark* tersebut.

Referensi

[1] Neufert, Ernst. "BAUENTWURFSLEHRE : Data Arsitek". alih bahasa, Sunarto Tjahjadi, Ferryanto Chaidir. Jakarta. Penerbit Erlangga. 2002.

[2] Grus, Joel. "Data Science from Scratch : First principles with Python". 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc. 2015.

[3] Ozdemir, Sinan. "Principles of Data Science". 35 Livery Street Birmingham B3 2PB, UK. ISBN 978-1-78588-791-8. 2016.

[4] McKinney, William. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython". 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc. 2018

[5] McKinney, Wes. "pandas: powerful Python data analysis toolkit (Release 1.1.0)". pandas Development Team. 2020

[6] VanderPlas, Jake. "Python Data Science Handbook: Essential Tools for Working with Data". 1005 Gravenstein Highway North, Sebastopol, CA 95472 : O'Reilly Media, Inc. 2017.

[7] Darren, John Hunter. Firing, Dale Eric. Droettboom, Michael. "Mathplotlib Realease 3.3.1".

[8] Qassim, Ahmed. "Easy Steps To Plot Geographic Data on a Map Python" Tersedia : https://towardsdatascience.com-/easy-steps-to-plot-geographic-data-on-a-map-python-11217859a2db. Diakses: Agustus. 20, 2020.

[9] AdityaGogoi. "Notes-on-Python-for-Data-Analysis-2nd-Edition". Tersedia : https://github.com/AdityaGogoi-/Notes-on-Python-for-Data-Analysis-2nd-Edition. Diakses : Agustus. 10, 2020.

[10] Gboeing. "Data visualization with pandas" https://github.com/jmportilla /urban-data-science/blob/ma-ster/12-Python-Data-Visualization/pandas-data-visualization.ipynb. Diakses : Agustus. 21, 2020.

[11] Datadave. "Instructional Set - Plotting with matplotlib and pandas". https://github.com/jmportilla/data-science-course/blob/master/lessons/lesson06_matplotlib_and_ED-A/Visualization_Instructional_Set.ipynb. Diakses : Agustus. 21, 2020.

[12] Sullivan, John. "Data Cleaning with Python and Pandas: Detecting Missing Values" Tersedia : https://towards-datascience.com/data-cleaning-with-python-and-pandas-detecting-missing-values-3e9c6ebcf78b. Diakses: Agustus. 20, 2020.

[13] Rosa A.S. dan M. Shalahuddin. "Rekayasa Perangkat Lunak : Terstruktur dan Berorientasi Objek". Jl. Buah Batu, Burangrang, Kec. Lengkong, Kota Bandung, Jawa Barat 40264, Penerbit Informatika Bandung, 2018.

[14] Yulianto, Yulianto & Ramadiani, Ramadiani & harsa kridalaksana, Awang. "Penerapan Formula Haversine Pada Sistem Informasi Geografis Pencarian Jarak Terdekat Lokasi Lapangan Futsal". Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer. 13. 14. 2018. 10.3087-2/jim.v13i1.1027.

[15] Anaconda Inc . "Anaconda Documentation," 2016. Tersedia : https://docs.anaconda.com/anaconda/ Diakses : September, 11, 2020.

[16] Boeing, G. 2017. "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks." Computers, Environment and Urban Systems 65, 126-139. doi:10.1016/j.compenv-urbsys.2017.05.004