# APLIKASI PENCARIAN RUTE TERCEPAT KENDARAAN BERMOTOR DI KOTA MALANG MENGGUNAKAN ALGORITMA *DIJKSTRA* DAN *MULTIPLE REGRESSION*

**SKRIPSI**

**OLEH:**
**M. DHOFIR ALIBI**
**NIM. 13650098**

**JURUSAN TEKNIK INFORMATIKA**
**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM**
**MALANG**
**2019**

# APLIKASI PENCARIAN RUTE TERCEPAT KENDARAAN BERMOTOR DI KOTA MALANG MENGGUNAKAN ALGORITMA *DIJKSTRA* DAN *MULTIPLE REGRESSION*

**SKRIPSI**

**Diajukan kepada:**
**Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang**
**Untuk Memenuhi Salah Satu Persyaratan Dalam**
**Memperoleh Gelar Sarjana Komputer (S.Kom)**

**Oleh :**
**M. DHOFIR ALIBI**
**NIM. 13650098**

**JURUSAN TEKNIK INFORMATIKA**
**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG**
**2019**

## HALAMAN PERSETUJUAN

# APLIKASI PENCARIAN RUTE TERCEPAT KENDARAAN BERMOTOR DI KOTA MALANG MENGGUNAKAN ALGORITMA *DIJKSTRA* DAN *MULTIPLE REGRESSION*

**SKRIPSI**


**OLEH:**
**M. DHOFIR ALIBI**
**NIM. 13650098**


Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : Mei 2019

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Irwan Budi Santoso, M.Kom | M. Imamudin, Lc., MA |
| NIP. 19770103 201101 1 004 | NIP. 19740602 200901 1 010 |

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

## HALAMAN PENGESAHAN

# APLIKASI PENCARIAN RUTE TERCEPAT KENDARAAN BERMOTOR DI KOTA MALANG MENGGUNAKAN ALGORITMA *DIJKSTRA* DAN *MULTIPLE REGRESSION*

**SKRIPSI**


**Oleh :**
**M. DHOFIR ALIBI**
**NIM. 13650098**


Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan untuk
Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal : Juni 2019

| Susunan Dewan Penguji | | Tanda Tangan |
|---|---|---|
| Penguji Utama | : Roro Inda Melani, MT., M.Sc<br>NIP. 19780925 200501 2 008 | ( ) |
| Ketua Penguji | : Dr. Cahyo Crysdian<br>NIP. 19740424 200901 1 008 | ( ) |
| Sekretaris Penguji | : Irwan Budi Santoso, M.Kom<br>NIP. 19770103 201101 1 004 | ( ) |
| Anggota Penguji | : M. Imamudin, Lc., MA<br>NIP. 19740602 200901 1 010 | ( ) |

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

## HALAMAN PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini :

Nama : M. Dhofir Alibi

Nim : 13650098

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Skripsi : **APLIKASI PENCARIAN RUTE TERCEPAT KENDARAAN BERMOTOR DI KOTA MALANG MENGGUNAKAN ALGORITMA *DIJKSTRA* DAN *MULTIPLE REGRESSION*.**

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar – benar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan Skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, Mei 2019

Yang membuat pernyataan


M. Dhofir Alibi

NIM. 13650098

**MOTTO**

ليس الفتى من يقول كان أبى,  ولكن الفتى ها أنا ذا

*Bukanlah seorang pemuda yang mengatakan itu Ayahku, akan tetapi seorang pemuda adalah yang mengatakan inilah Saya.*

*- Ali bin Abi Thalib*

*Sejatinya manusia hidup adalah untuk menjadi lebih baik.*

*- M. Dhofir Alibi*

## HALAMAN PERSEMBAHAN

Alhamdulillah puji syukur ke Hadirat Allah SWT yang telah memberikan nikmat *dhohiriyah* dan *bathinyah* sehingga penulis mampu untuk menyelesaikan studi S1 di kampus UIN Malang ini. Salawat serta salam selalu tercurahkan kepada Baginda Nabi Muhammad SAW, yang telah membimbing umatnya menuju jalan yang benar.

Terima kasih kepada kedua orang tua, sang Ayah tercinta, Bapak Sumarwan yang selalu mendidik dan memberikan contoh kehidupan terutama hal agama. Ibu Sumarni yang tak lelah untuk menyayangi, rela berkorban, sehingga dapat merasakan kehidupan sampai saat ini. Tak lupa saudara saya, semoga seluruh tujuan tercapai dan diberikan yang terbaik.

Teruntuk seluruh guru, ustad, kiai dan dosen mulai Sekolah Dasar, Pondok Pesantren hingga Perguruan Tinggi. Pembimbing skripsiku Irwan Budi Santoso, M. Kom. dan M. Imamudin, Lc., MA yang dengan tulus, sabar, dan ikhlas membimbing serta menyalurkan pengetahuannya. Nasehat-nasehat bapak akan selalu diingat dan kita akan terus terhubung melalui sambung doa sampai akhir hayatku.

Teman seperjuangan *Fortinity* TI'13 UIN Maliki Malang, adik-adik angkatan, Majelis Al-Fataa serta keluarga Kontrakan70an yang telah meluangkan waktunya. Rekan-rekan dan semua pihak yang tak bisa disebutkan satu persatu, terima kasih. Semoga terus terhubung meskipun dalam untaian doa yang mengiringi kesuksesan kita.

## KATA PENGANTAR

*Assalamualaikum Warahmatullahi Wabarokatuhu.*

*Alhamdulillahi Robbil 'Alamiin*, segala puji bagi Allah yang selalu memberikan nikmat *dhohiriyah* dan nikmat *bathiniyah* dalam proses penyelesaian skripsi ini. Sholawat serta salam selalu tercurahkan kepada junjungan, baginda dan pusaka umat islam, Nabi Muhammad SAW yang telah memberikan teladan, bimbingan dan petunjuk, sehingga umat manusia menjadi lebih beradab.

Dalam menyelesaikan skripsi ini, banyak pihak yang telah memberikan bantuan baik secara moril, nasihat dan semangat maupun materiil. Atas segala bantuan yang telah diberikan, penulis ingin menyampaikan doa dan ucapan terimakasih yang sedalam-dalamnya kepada:

1. Iwan Budi Santoso, M. Kom., selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, mengarahkan dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.
2. M. Imamudin, Lc, M.A, selaku dosen pembimbing II yang telah membimbing serta memberikan masukan kepada penulis dalam pengerjaan skripsi ini.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika yang telah memberikan motivasi untuk terus berjuang.
4. Segenap dosen teknik informatika yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
5. Teman-teman seperjuangan teknik informatika Fortinity 2013.

Berbagai kekurangan dan kesalahan mungkin pembaca temukan dalam penulisan skripsi ini, untuk itu penulis menerima segala kritik dan saran yang membangun dari pembaca sekalian. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya dan semoga karya ini senantiasa dapat memberi manfaat.

*Wassalamualaikum Warahmatullahi.Wabarokatuhu.*

Malang, Mei 2019

Penulis

## DAFTAR ISI

## DAFTAR GAMBAR

## DAFTAR TABEL

# ABSTRAK

Alibi, M. Dhofir. **Aplikasi Pencarian Rute Tercepat Kendaraan Bermotor di Kota Malang Menggunakan Algoritma *Dijkstra* dan *Multiple Regression***. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing (I) Irwan Budi Santoso, M. Kom, (II) M. Imamudin, Lc., MA.

**Kata Kunci** : Kemacetan, Rute Tercepat, Algoritma *Dijkstra, Multiple Regression*.

Kota Malang termasuk salah satu kota yang memiliki tingkat kepadatan penduduk yang sangat tinggi. Kemacetan lalu lintas sering menjadi masalah besar bagi masyarakat. Berdasarkan data hasil riset INRIX pada tahun 2017, lembaga survei asal Amerika Serikat yang berfokus pada riset lalu lintas jalan menunjukkan bahwa kemacetan di Malang Raya mengalahkan Kota Surabaya yang berada di posisi kesembilan. Dibutuhkan sebuah petunjuk arah agar masyarakat dapat terhindar dari kemacetan dengan estimasi waktu yang cepat. Berdasarkan beberapa pertimbangan tersebut, maka perlu adanya suatu penelitian untuk menentukan jalur tercepat menuju stasiun Kotabaru di Kota Malang. Sebuah aplikasi pencarian rute tercepat kendaraan bermotor di kota Malang dibangun dengan menggunakan algoritma *dijkstra* dan *multiple regression*. Penelitian ini bertujuan untuk mengetahui akurasi rute tercepat yang dihasilkan dari aplikasi yang dibangun menggunakan algoritma *dijkstra* dan *multiple regression*. Data yang digunakan dalam penelitian ini adalah data jumlah kendaraan pada setiap jalan dari rute yang telah ditentukan dan data kecepatan yang diperoleh dari kecepatan rata-rata kendaraan bermotor yang diambil secara langsung. Kemudian diproses menggunakan algoritma *multiple regression* agar menghasilkan sebuah bobot. Bobot tersebut digunakan untuk data masukan pada penentuan rute tercepat menggunakan algoritma *dijkstra*. Kemudian dihitung persentase *error*-nya. Sehingga akurasi aplikasi dapat dihitung berdasarkan persentase *error* yang dihasilkan. Proses pengujian dilakukan dengan membandingkan data waktu tempuh yang dihasilkan aplikasi dengan data *testing* rute secara langsung. Dari 10 kali percobaan yang telah dilakukan, persentase *error* yang terkecil yang dihasilkan adalah sebesar 3,67% dan persentase *error* terbesar yang dihasilkan adalah sebesar 20,98%. Nilai rata-rata persentase *error* adalah sebesar 11,21% dan nilai akurasi yang dihasilkan adalah 88,79%. Sebagai perbandingan, dilakukan perhitungan persentase *error* terhadap aplikasi yang sudah ada yaitu Google *maps*. Nilai persentase *error* yang dihasilkan oleh aplikasi lebih kecil jika dibandingkan dengan nilai persentase *error* yang dihasilkan oleh Google maps, yaitu sebesar 38,30%.

# ABSTRACT

Alibi, M. Dhofir. **The Fastest Vehicle Route Search Application in Malang City Uses Dijkstra Algorithm and Multiple Regression**. Undergradute Thesis. Informatics Engineering Department of Science and Technology Faculty Islamic State University Maulana Malik Ibrahim Malang. Supervisor: (I) Irwan Budi Santoso, M. Kom, (II) M. Imamudin, Lc., MA.

**Keyword :** Congestion, Fastest Route, Dijkstra Algorithm, Multiple Regression.

Malang City is one of the cities that has a very high population density. Congestions are often a big problem for the community. Based on INRIX's research data in 2017, a survey institute from the United States that focuses on road traffic research shows that congestion in Malang City beats Surabaya City which is in ninth position. A direction is needed so that people can avoid congestion with a fast estimate of time. Based on these considerations, it is necessary to have a research to determine the fastest route to Kotabaru station in Malang City. A search application for the fastest route of vehicles in Malang City was built using the dijkstra algorithm and multiple regression. This research aims to determine the accuracy of the fastest route generated from the application built using the dijkstra algorithm and multiple regression. The data used in this research is the data on the number of vehicles on each road from a predetermined route and speed data obtained from the average speed of vehicles taken directly. Then it is processed using a multiple regression algorithm to produce a weight. This weight is used for input data in determining the fastest route using the dijkstra algorithm. Then the error percentage is calculated. So that the accuracy of the application can be calculated based on the percentage of errors generated. The testing process is done by comparing the travel time data generated by the application with route testing data directly. Of the 10 experiments that have been carried out, the smallest percentage error produced is 3.67% and the largest percentage of errors generated is 20.98%. The average value of error percentage is 11.21% and the resulting accuracy value is 88.79%. For comparison, a percentage error is calculated on an existing application, namely Google maps. The percentage value of errors generated by the application is smaller when compared to the percentage value of errors generated by Google maps, which is equal to 38.30%.

## ملخص


عليبي ، محمد ظفير. **أسرع تطبيق للبحث عن طريق المركبات في مدينة مالانج باستخدام خوارزمية ديكسترا والانحدار المتعدد** . أطروحة الجامعية . قسم هندسة المعلوماتية لكلية العلوم والتكنولوجيا في جامعة الحكومة الإسلامية مولانا مالك إبراهيم مالانج . المشرف : (الأول) اروان بودي سانتوسو، ماجيستير، (الثانى) محمد إمام الدين، ما جيستير

**الكلمات الرئيسية** : الازدحام ، أسرع الطرق ، خوارزمية ديكسترا ، الانحدار المتعدد

"ملانج" هي مدينة من المدن ذات الكثافة السكانية العالية. غالبًا ما تشكل مشكلة ازدحام المرور مشكلة كبيرة للمجتمع. بناءً علي بيانات أبحاث "INRIX" في عام ٢٠١٧ ، تُظهر مؤسسات المسح من الولايات المتحدة التي تركز على أبحاث حركة المرور على الطرق أن الاحتقان في ملانج يتفوق على مدينة سورابايا التي تحتل المرتبة التاسعة. هناك حاجة إلى اتجاه حتى يتمكن الناس من تجنب الازدحام مع تقدير سريع للوقت. إستنادا على تلك الاعتبارات، لابدا أن تكون أبحاث لتحديد طريق أسرع إلى محطة كوتابارو في مدينة ملانج. تم بناء تطبيق بحث عن أسرع طريق للسيارات الآلية في مدينة مالانج باستخدام خوارزمية dijkstra والانحدار المتعدد. تهدف هذا البحث إلى تحديد دقة أسرع مسار تم إنشاؤه من التطبيق المصمم باستخدام خوارزمية ديكسترا والانحدار المتعدد. البيان المستعمل في هذا البحث هو البيان المتعلق بعدد المركبات على كل طريق من المسار المعين وبيان السرعة التي تحصل عليها من متوسط سرعة المركبات الآلية المأخوذة مباشرةً. ثم تعالج باستخدام خوارزمية انحدار متعددة لإنتاج الوزن. يستعمل هذا الوزن لإدخال البيان في تحديد أسرع مسار باستخدام خوارزمية .dijkstra ثم يحسب نسبة الخطأ. بحيث يمكن حساب دقة التطبيق على أساس النسبة المئوية للأخطاء المحصولة. تتم عملية الاختبار عن طريق مقارنة بيانات وقت السفر الناتجة عن التطبيق مع بيانات اختبار الطريق مباشرة. من بين 10 تجارب تم إجراؤها ، كان أصغر خطأ تم إنتاجه هو 3.67% وأكبر نسبة مئوية من الأخطاء الناتجة 20.98%. متوسط قيمة النسبة المئوية للخطأ هو 11.21% والقيمة الناتجة هي 88.79%. للمقارنة ، يتم حساب نسبة الخطأ في تطبيق موجود ، وهو خرائط .Google تكون النسبة المئوية لقيمة الأخطاء الناتجة عن التطبيق أصغر مقارنةً مع النسبة المئوية لقيمة الأخطاء الناتجة عن خرائط Google ، والتي تساوي 38.30%.

# BAB I

# PENDAHULUAN

Bab ini berisi penjelasan mengenai alasan dari diangkatnya topik dan dilakukannya penelitian ini, identifikasi permasalahan berdasarkan alasan dari penelitian ini, fokus atau tujuan dilakukannya penelitian ini, manfaat yang diharapkan, dan batasan penelitian agar cakupannya tidak meluas dari fokus penelitian ini.

## 1.1 Latar Belakang

Stasiun kereta api adalah sebuah prasarana transportasi sebagai tempat untuk menurunkan atau menaikkan penumpang, perpindahan intra atau antar moda transportasi, serta mengatur kedatangan atau pemberangkatan kendaraan umum. Di sebuah daerah atau kota tentunya terdapat salah satu prasarana transportasi, tidak terkecuali di Kota Malang. Di kota Malang, terdapat dua prasarana transportasi yang berupa stasiun, salah satunya adalah stasiun Kotabaru.

Kota Malang merupakan salah satu dari beberapa kota yang mempunyai tingkat kepadatan penduduk sangat tinggi. Kemacetan lalu lintas sering menjadi masalah besar bagi masyarakat sekitar yang mayoritasnya adalah mahasiswa dan pekerja. Hal tersebut akan menimbulkan masalah bagi sebagian masyarakat yang hendak bepergian menggunakan sarana transportasi tersebut. Salah satu masalahnya adalah ketika masyarakat tersebut telah membeli tiket dari sarana transportasi yang telah disediakan oleh stasiun Kotabaru.

Berdasarkan data hasil riset INRIX, lembaga survei asal Amerika Serikat yang berfokus pada riset lalu lintas jalan menunjukkan bahwa kemacetan di Malang Raya mengalahkan Kota Surabaya yang berada di posisi kesembilan (Ratri, 2017).

Dengan mempertimbangkan hal tersebut, dibutuhkan sebuah petunjuk arah agar masyarakat dapat terhindar dari kemacetan dengan estimasi waktu yang cepat. Maka diperlukan sebuah penelitian untuk menentukan jalur tercepat menuju stasiun Kotabaru di Kota Malang. Penelitian ini ditujukan agar dapat memudahkan dan menolong masyarakat agar terhindar dari kemacetan dan menemukan rute tercepat. Karena sebagai sesama manusia kita harus saling tolong menolong dalam kebaikan. Firman Allah surat Al-Maidah ayat 2 pada baris terakhir berbunyi:

وَتَعَاوَنُواْ عَلَى ٱلۡبِرِّ وَٱلتَّقۡوَىٰۖ وَلَا تَعَاوَنُواْ عَلَى ٱلۡإِثۡمِ وَٱلۡعُدۡوَٰنِۚ وَٱتَّقُواْ ٱللَّهَۖ إِنَّ ٱللَّهَ شَدِيدُ ٱلۡعِقَابِ ٢

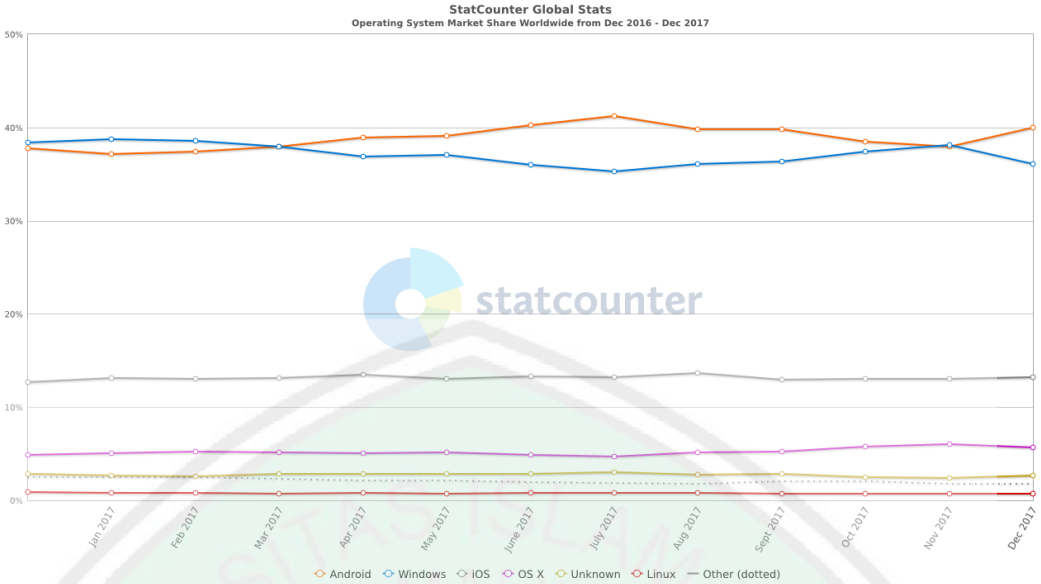
Artinya : *(Bertolong-menolonglah kamu dalam kebaikan) dalam mengerjakan yang dititahkan (dan ketakwaan) dengan meninggalkan apa-apa yang dilarang (dan janganlah kamu bertolong-menolong) pada ta'awanu dibuang salah satu di antara dua ta' pada asalanya (dalam berbuat dosa) kayau maksiat (dan pelanggaran) artinya melampaui batas-batas ajaran Allah. (Dan bertakwalah kamu kepada Allah) takutlah kamu kepada azab siksa-Nya dengan menaati-Nya (sesungguhnya Allah amat berat siksa-Nya) bagi orang yang menentang-Nya. (Tafsir Jalalain).*

Di dalam surat tersebut, disebutkan secara jelas bahwa kita harus tolong-menolong dalam hal kebaikan dan janganlah bertolong menolong dalam hal keburukan. Menurut Prof. Dr. M. Quraish Shihab dalam kitab Tafsir Al-Misbah, maksud dari tolong menolong ini adalah tidak hanya kepada orang islam, namun juga kepada setiap kalangan masyarakat.

Pada era yang serba canggih dengan perkembangan teknologi yang sangat pesat seperti sekarang ini, perangkat *mobile* saat ini telah membawa kemajuan yang sangat berarti dalam berbagai aspek, terutama bagi kebutuhan masyarakat dalam kehidupan sehari-hari. Perangkat *mobile* dapat mempermudah pekerjaan atau aktivitas bagi para penggunanya, karena dengan ukuran yang mudah untuk dibawa ke mana-mana. Seiring dengan pesatnya perkembangan teknologi perangkat *mobile* saat ini, banyak aplikasi-aplikasi berbasis *mobile* yang telah dibangun oleh para developer yang berguna bagi masyarakat, perusahaan, maupun institusi pemerintah.

Perangkat *mobile* yang berkembang saat ini mempunyai berbagai macam sistem operasi, diantaranya adalah Android, iOS, Blackberry, dan Windows Phone. Dibandingkan dengan sistem operasi lainnya, sistem operasi Android adalah sistem operasi yang bersifat terbuka dan dapat dikembangkan oleh siapa pun, sehingga penulis menginginkan membangun sebuah aplikasi berbasis *mobile* menggunakan sistem operasi Android.

Sistem operasi Android adalah sebuah sistem operasi berbasis Linux yang dirancang untuk perangkat *mobile* layar sentuh seperti telepon pintar dan komputer tablet yang bersifat *open source*.




Gambar 1.1 Grafik pengguna Sistem Operasi di dunia (sumber gs.statcounter.com).

Dari grafik pengguna sistem operasi di seluruh dunia, dapat dilihat bahwa sistem operasi Android menduduki peringkat pertama pada bulan Desember 2017. Dengan dukungan teknologi yang mumpuni, sistem operasi Android dapat merangsek melalui berbagai segmen pasar, baik dari segmen kalangan bawah, segmen kalangan menengah, maupun segmen kalangan atas.

Sistem operasi Android merupakan sebuah sistem yang dimiliki oleh Google. Fitur-fitur Google juga terdapat di dalam sistem operasi Android salah satunya adalah Google *maps*. Google *maps* berguna untuk memberikan layanan berupa citra satelit, peta jalan, dan perencanaan rute untuk bepergian dengan berjalan kaki, mobil, sepeda, dan angkutan umum. Selain itu, Google juga menyediakan API (*Application Program interface*) untuk para *developer*, agar para *developer* dapat memanfaatkan layanan Google untuk aplikasi yang mereka kembangkan. Seiring berkembangnya sistem operasi Android, para *developer* dapat menggunakan fungsi dari fitur lokasi yaitu GPS (*Global Positioning System*) atau disebut dengan LBS (*Location Based Services*).

*Location based services* merupakan sebuah layanan berbasis lokasi yang berguna untuk menemukan lokasi perangkat yang digunakan oleh pengguna. Dengan adanya layanan *location based services*, pengguna dapat menemukan posisi seseorang, dan rute untuk menuju lokasi yang dituju.

Dengan pertimbangan-pertimbangan yang telah disebutkan, dibutuhkan sebuah cara atau metode untuk menentukan rute tercepat kendaraan bermotor menuju prasarana transportasi di Kota Malang. Terdapat beberapa algoritma yang dapat digunakan untuk penentuan rute tercepat. Salah satunya adalah algoritma *dijkstra*. Penelitian ini akan menggunakan algoritma tersebut dalam pada penentuan rute tercepat dan *multiple regression* yang digunakan sebagai metode untuk mengestimasi atau memprediksi nilai rata-rata (populasi) satu variabel dependen berdasarkan dua atau lebih variabel independen.

Algoritma *dijkstra* adalah salah satu metode untuk memecahkan masalah pencarian rute terpendek/tercepat. Algoritma ini menentukan rute atau jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik yang lainnya. Pada algoritma *dijkstra*, *node* digunakan karena algoritma *dijkstra* menggunakan *graph* berarah untuk penentu rute terpendek.

*Multiple regression* adalah salah satu teknik multivariat yang digunakan untuk mengestimasi hubungan antara satu variabel dependen dengan dua atau lebih himpunan variabel independen.

Sesuai dengan latar belakang yang sudah dipaparkan diatas, maka penulis mengangkat judul **"Aplikasi Pencarian Rute Tercepat Kendaraan Bermotor di Kota Malang Menggunakan Algoritma *Dijkstra* dan *Multiple Regression*"**.

## 1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dijabarkan pada sub bab sebelumnya, masalah yang diidentifikasi adalah seberapa akurat rute tercepat yang dihasilkan dari aplikasi yang dibangun menggunakan algoritma *dijkstra* dan *multiple regression*?

## 1.3 Tujuan Penelitian

Fokus dari penelitian ini berdasarkan identifikasi masalah yang telah disebutkan adalah untuk mengetahui akurasi rute tercepat yang dihasilkan dari aplikasi yang dibangun menggunakan algoritma *dijkstra* dan *multiple regression*.

## 1.4 Manfaat Penelitian

Secara umum, manfaat yang didapatkan dari penelitian ini adalah kemudahan bagi masyarakat di Kota Malang untuk mengetahui rute tercepat menuju stasiun Kotabaru di Kota Malang.

## 1.5 Batasan Penelitian

Pembahasan-pembahasan dalam penelitian ini diberikan sebuah batasan agar tidak meluas dan tidak menyimpang dari fokus penelitian ini. Batasan-batasan tersebut adalah

1. Titik awal keberangkatan dimulai dari Kampus Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prasarana Transportasi yang menjadi titik akhir adalah stasiun Kotabaru.
3. Parameter untuk mengukur adalah jumlah kendaraan, jarak, waktu, hari, dan kecepatan berkendara.
4. Kendaraan bermotor yang digunakan adalah sepeda motor.
5. Simulasi menggunakan *software* Android Studio.

# BAB II

# TINJAUAN PUSTAKA

Pada bab ini, dibahas mengenai tinjauan pustaka dari penelitian. Tinjauan pustaka berisi tentang penelitian yang terkait dari penelitian ini, referensi dan konsep teori yang digunakan pada penelitian ini.

## 2.1 *Location Based Service* (LBS)

*Location based service* (LBS) atau layanan lokasi adalah sebuah layanan informasi yang dapat diakses dengan perangkat bergerak melalui jaringan dan mampu menampilkan posisi secara geografis keberadaan perangkat bergerak tersebut. *location based service* dapat berfungsi sebagai layanan untuk mengidentifikasi lokasi dari seseorang atau suatu objek tertentu. Menurut Qusay H. Mahmoud, *location based service* adalah sebuah layanan yang digunakan untuk mengetahui posisi pengguna, kemudian mengunakan informasi tersebut untuk menyediakan jasa dan aplikasi yang personal. Terdapat dua unsur pertama dari *location based service*, yaitu :

1. *Location Manager* (API Maps)

   Menyediakan *tool* atau *source* untuk LBS, *application programming interface* (API) Maps menyediakan fasilitas untuk menampilkan, memanipulasi peta beserta fitur-fitur lainnya seperti tampilan satelit, jalan, maupun gabungannya. Paket ini berada pada com.google.android.maps.

2. *Location Provider* (API *Location*)

   Menyediakan teknologi pencarian lokasi yang digunakan oleh perangkat. API *location* berhubungan dengan data GPS (*Global Positioning System*)

dan data lokasi *real*-time. Pengguna dapat menentukan lokasinya, melacak gerakan atau perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan (Safaat, 2011). API *location* berada pada paket Android, yaitu dalam paket android.locaion. Lokasi, perpindahan, serta kedekatan dengan lokasi tertentu dapat ditentukan melalui *location manager*.

## 2.2 Google *Maps*

Google *Maps* adalah jasa peta gratis dan *online* disediakan oleh Google yang dapat ditemukan di https://maps.google.com. Di dalam situs tersebut terdapat informasi geografis dari hampir seluruh pemukaan bumi, kecuali daerah dua kutub yaitu utara dan selatan. Layanan ini dirancang oleh Google agar interaktif, karena di dalam layan tersebut, peta dapat digeser sesuai keinginan pengguna, diubah level *zoom*-nya, serta diubah tampilan jenis peta-nya.

Google *maps* API memungkinkan *developer* untuk mengintegrasikan layanan *maps* ke dalam aplikasi yang dibuat. Google *maps* API memungkinkan layanan Google *maps* untuk ditanamkan dalam situs eksternal, dan di dalam situs tersebut dapat dilakukan *overlay*.

Pada awalnya, Google *maps* API hanya dalam bentuk JavaScript API. Kemudian Google *maps* API dikembangkan sehingga dapat disertakan API ke dalam Adobe Flash, layanan untuk mengambil gambar dari peta statis dan layanan dari *website* untuk *geocoding* yang hasilnya adalah sebuah petunjuk rute.

Android memiliki sebuah *keyClass* dalam *library maps* yaitu *MapView*. *MapView* merupakan sebuah *subClass* dari *view group* pada standar *library* Android. Sebuah *MapView* dapat menampilkan *maps* dengan data yang diambil dari

layanan Google *maps*. Layanan *maps* interaktif juga disediakan seperti *zoom* peta secara otomatis, termasuk permintaan jaringan untuk *maps*. Semua elemen UI yang diperlukan pengguna juga disediakan agar *maps* dapat dikendalikan secara interaktif. Aplikasi tersebut juga dapat menggunakn *method* kelas *MapView* untuk mengontrol secara terprogram dan menarik sejumlah tampilan di atas *maps*.

### 2.3 Regresi Linear

Regresi linear adalah alat statistik yang dipergunakan untuk mengetahui pengaruh antara satu variabel atau beberapa variabel terhadap satu buah variabel. Variabel yang mempengaruhi sering disebut variabel bebas, variabel independen atau variabel penjelas. Variabel yang dipengaruhi sering disebut dengan variabel terikat atau variabel dependen.

Soegyarto Mangkuatmodjo (2004) menyatakan dalam analisis regresi dikenal 2 jenis variabel yaitu :

1. Variabel bebas disebut juga variabel independen atau *predictor* merupakan variabel yang berubah-ubah tanpa adanya pengaruh variabel lain. Tetapi sebaliknya, suatu perubahan yang terjadi pada variabel bebas akan mengakibatkan terjadinya perubahan variabel lain.
2. Variabel tidak bebas disebut juga variabel dependen atau *respon* merupakan variabel yang hanya akan berubah manakala terjadi perubahan pada variabel lain.

Untuk mempelajari hubungan-hubungan antara variabel bebas maka regresi linear terdiri dari dua bentuk yaitu :

1. Regresi linear sederhana (*simple linear regression*).
2. Regresi linear berganda (*multiple linear regression*).

Tujuan utama regresi adalah untuk memprediksi nilai dari suatu variabel dalam hubungannya dengan variabel lain berdasarkan persamaan regresi yang diperoleh.

Prinsip dasar yang harus dipenuhi dalam membangun suatu persamaan regresi adalah bahwa antara variabel dependen dengan variabel independennya mempunyai sifat hubungan sebab akibat (hubungan kausalitasi), baik yang didasarkan pada teori (*theoritical*), hasil penelitian sebelumnya (*prior research*), atau pun yang didasarkan pada penjelasan logis (*logical explaination*) tertentu (Algifari, 2000).

### 2.3.1 Regresi Linear Sederhana

Regresi linear sederhana adalah hubungan secara linear antara satu variabel independen (X) dengan variabel dependen (Y). Bentuk umum persamaan linear berganda adalah :

$$Y = a + bX \qquad (2.1)$$

yang menunjukkan bahwa :

*Y* adalah sebagai variabel dependen.

*a* adalah intersep (titik potong kurva terhadap sumbu Y).

*b* adalah kemiringan (*slope*) kurva linear.

*X* adalah variabel independen.

Persamaan diatas dapat digunakan untuk menaksir nilai *Y* jika nilai *a*, *b*, dan *X* diketahui. Nilai *a* merupakan nilai *Y* yang dipotong oleh kurva linear pada sumbu vertikal *Y*. Atau dengan kata lain, *a* adalah nilai *Y* jika $X = 0$. Nilai *b* adalah kemiringan (*slope*) kurva linear yang menunjukkan besarnya perubahan nilai *Y* sebagai akibat dari perubahan setiap unit nilai *X*. Besarnya *a* dan *b* konstan sepanjang kurva linear.

Persamaan tersebut merupakan model matematis deterministik (*deterministic mathematical model*), sebab apabila nilai variabel *X* diketahui, maka nilai variabel *Y* dapat ditentukan tanpa mengandung faktor kesalahan (*error*).

### 2.3.2 Regresi Linear Berganda

Analisis regresi linear berganda adalah suatu metode statistik umum yang digunakan untuk meneliti hubungan antara sebuah variabel dependen dengan beberapa variabel independen. Tujuan analisis regresi berganda adalah menggunakan nilai-nilai variabel independen yang diketahui untuk meramalkan nilai variabel dependen (Sulaiman, 2004).

1. Model Regresi Linear Berganda

   Perbedaan antara regresi linear berganda dan regresi linear sederhana adalah jika dalam regresi sederhana hanya ada satu peubah bebas *X* yang dihubungkan dengan satu peubah tak bebas *Y* linear (pangkat satu) dalam *X* sehingga terbentuk model $Y = a + bX$, maka dalam regresi linear berganda terdapat sejumlah (sebut *k* buah, $k \geq 2$) peubah bebas yang dihubungkan dengan *Y* linear atau berpangkat satu dalam semua peubah bebas. Jika peubah bebas itu $X_1, X_2, \ldots, X_k (k \geq 2)$ dan seperti biasa peubah tak bebasnya *Y*, maka bentuk umum regresi linear berganda adalah sebagai berikut:

   $$\hat{Y} = b_0 + b_1X_1 + b_2X_2 + \ldots + b_kX_k \qquad (2.2)$$

   yang menyatakan bahwa:

   $\hat{Y}$ = nilai estimasi *Y*

   $X_1, X_2$ = nilai variabel independen $X_1$ dan $X_2$

   $b_0, b_1, b_2$ = *slope* yang berhubungan dengan variabel $X_1$ dan $X_2$

   Bentuk hubungan regresi linear berganda diilustrasikan dalam bagan berikut

$$\begin{bmatrix} Y_1 \\ Y_2 \\ - \\ - \\ - \\ Y_n \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} & - & - & - & X_{1k} \\ X_{21} & X_{22} & - & - & - & X_{2k} \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ X_{n1} & X_{n2} & - & - & - & X_{nk} \end{bmatrix} \quad (2.3)$$

Data dari regresi linear berganda ditunjukkan pada Tabel 2.1 berikut.

Tabel 2.1 Tabel Regresi Linear Berganda.

| Nomor Observasi | Responden ($Y_i$) | Variabel Bebas | | | |
|---|---|---|---|---|---|
| | | $X_{1i}$ | $X_{2i}$ | - | $X_{ki}$ |
| 1 | $Y_1$ | $X_{11}$ | $X_{21}$ | - | $X_{k1}$ |
| 2 | $Y_2$ | $X_{12}$ | $X_{22}$ | - | $X_{k2}$ |
| - | - | - | - | - | - |
| N | $Y_n$ | $X_{1n}$ | $X_{2n}$ | - | $X_{kn}$ |

2. Estimasi Parameter Linear Berganda

Bila diketahui n pasangan data X dan Y dengan variabel X sebanyak p, serta estimasi parameter regresi berganda dimisalkan b untuk b, maka dengan melakukan ekpektasi persamaan 2.2 atau 2.3 dapat ditentukan nilai $\hat{Y}$ dari model regresi dan secara matematis ditulis

$$\hat{Y} = Xb \quad (2.4)$$

Dengan $\hat{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ - \\ Y_n \end{bmatrix}$ dan $b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ - \\ b_n \end{bmatrix}$ (2.5)

Estimasi parameter dari model regresi linear berganda juga dilakukan dengan metode *ordinary least squares* (OLS). Metode ini meminimumkan jumlah kuadrat dari *error*. Kesalahan (*error*) dalam regresi linear berganda ditulis dengan $\mathcal{E}$ yang diperoleh berdasarkan selisih antara nilai sesungguhnya Y dengan nilai estimasi model $\hat{Y}$.

$$\mathcal{E} = Y - \hat{Y} \quad (2.6)$$

Sesuai prinsip OLS, masing-masing dari *error* dikuadratkan. Dalam bentuk perkalian matriks ditulis sebagai berikut

$$\mathcal{E}'\mathcal{E} = (Y - \hat{Y})' (Y - \hat{Y})$$

$$= (Y - Xb)' (Y - Xb) \quad (2.7)$$

Jumlah kuadrat kesalahan pada persamaan 2.7 selanjutnya disebut SSE (*sum square of error*). Kemudian dengan metode OLS, persamaan 2.7 tersebut diminimumkan dengan menurunkan atau mendifferensialkan SSE terhadap parameter yang diestimasi dan disamadengankan dengan nol. Secara matematis dapat ditulis

$$b = [X'X]^{-1}X'Y \quad (2.8)$$

## 2.4 Algoritma *Dijkstra*

Algortima *dijkstra* ditemukan oleh Edsger W. Dijkstra dan di publikasikan pada tahun 1959 pada sebuah jurnal *Numerische Mathematik* yang berjudul "*A Note on Two Problems in Connecxion with Graphs*". Algortma ini sering digambarkan sebagai algoritma greedy (tamak).

Dijkstra merupakan salah satu variasi bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi pencarian lintasan terpendek pada sebuah lintasan yang mempunyai panjang minimum dari *vertex* a ke z dalam *graph* berbobot, bobot tersebut adalah bilangan positif, jadi tidak dapat dilalui oleh *node* negatif. Namun jika terjadi demikian, maka penyelesaian yang diberikan adalah *infinity* (tak hingga). Pada algoritma *dijkstra*, *node* digunakan karena algoritma *dijkstra* menggunakan *graph* berarah untuk penentuan rute lintasan terpendek.

Proses untuk mendapatkan solusi optimum jalur terpendek dengan menghitung berdasarkan bobot terkecil dari satu titik ke titik yang lainnya. Perhitungan dilakukan terhadap sisi *graph* yang memiliki jalur awal dan jalur akhir. Misalnya, bila *vertices* dari sebuah *graph* melambangkan kota-kota dan bobot sisi (*edge weights*) melambangkan jarak antara kota tersebut, maka algoritma *dijkstra* dapat digunakan untuk menemukan jarak antara dua kota.

*Input* algoritma ini adalah sebuah *graph* berarah yang berbobot G dan sebuah sumber *vertex* s dalam G dan V adalah himpunan semua *vertices* dalam graf G. Setiap sisi dari *graph* ini adalah pasangan *vertices* (u,v) yang melambangkan hubungan dari *vertex* u ke *vertex* v. Himpunan semua tepi disebut E. Ongkos (*cost*) dari sebuah sisi dapat dianggap sebagai jarak antara dua *vertex*, yaitu jumlah jarak semua sisi dalam jalur tersebut. Untuk sepasang *vertex* s dan t dalam V, algoritma ini menghitung jarak terpendek dari s ke t. Maka fungsi untuk menentukan jalur terpendek dari *graph* berarah adalah G = (V,E).

Misalkan G adalah *graph* berarah yang mempunyai label dengan titik-titik V(G) = $\{v_1, v_2, ..., v_n\}$ dan *path* terpendek yang akan dicari adalah dari $v_1$ ke $v_n$. Algoritma *dijkstra* bermula dari titik $v_1$. Kemudian dalam iterasinya, algoritma *dijkstra* ini akan mencari satu titik dengan jumlah bobotnya dari titik satu terkecil. Titik-titik tersebut yang telah terpilih dipisahkan, dan titik-titik yang telah dipilih tidak akan diperhatikan kembali pada terasi selanjutnya.

Misalnya :

| | | |
|---|---|---|
| V(G) | = | $\{v_1, v_2, ..., v_n\}$. |
| L | = | Himpunan titik-titik $\in$ V(G) yang telah terpilih pada alur *path* terpendek. |

D(j) = Jumlah bobot dari *path* terkecil $v_1$ ke $v_j$.

w(i, j) = Bobot garis dari titik $v_i$ ke titik $v_j$.

w*(1, j) = Jumlah bobot dari *path* terkecil $v_1$ ke $v_j$.

Algoritma *dijkstra* yang digunakan untuk mencari *path* terpendek adalah sebagai berikut :

1. L = { };

   V = {$v_2$, $v_3$, ..., $v_n$}

2. Pada i = 2, ..., n, lakukan D(i) = w(1, i)
3. Selama $v_n \notin$ L lakukan :

   a. Pilih titik $v_k \in$ V – L dengan D(k) terkecil.

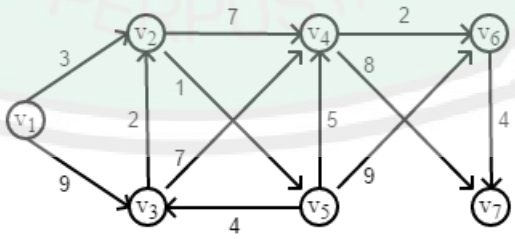
      $L = L \cup \{v_k\}$

   b. Pada setiap $v_j \in V - L$ lakukan :

      Jika D(j) > D(k) + W(k, j) maka gantikan D(j) dengan D(k) + W(k, j)

4. Untuk setiap $v_j \in V$, $w*(1, j) = D(j)$

Berdasarkan penjabaran algoritma *dijkstra* tersebut, *path* terpendek dari titik $v_1$ ke $v_n$ adalah melalui titik-titik pada L secara berurutan, dan jumlah bobot *path* terkecilnya adalah D(n).




Gambar 2.2 *Path* algoritma *dijkstra*

Matriks penghubung W untuk menyatakan *graph* pada Gambar 2.2 adalah sebagai berikut :

$$W = \begin{array}{c} \cdot \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} \infty & 3 & 9 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 7 & 1 & \infty & \infty \\ \infty & 2 & \infty & 7 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 2 & 8 \\ \infty & \infty & 4 & 5 & \infty & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 4 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix} \end{array} \quad (2.9)$$

Mula-mula L = { } dan V = {$v_2$, $v_3$, ..., $v_7$}

$$D(2) = W(1,2) = 3 \quad ; \quad D(3) = W(1,3) = 9$$

$$D(4) = W(1,4) = \infty \quad ; \quad D(5) = W(1,5) = \infty$$

$$D(6) = W(1,6) = \infty \quad ; \quad D(7) = W(1,7) = \infty$$

V-L = {$v_2$, $v_3$, ..., $v_7$} – { } = {$v_2$, $v_3$, ..., $v_7$}

$v_n = v_7 \notin L$, sehingga langkah 3(a) – 3(b) dilakukan.

3(a) : D(k) terkecil adalah D(2), sehingga $v_k = v_2$

$L = L \cup \{v_k\} = \{\ \} \cup \{v_2\} = \{v_2\}$

3(b) : V – L = {$v_2$, $v_3$, $v_4$, $v_5$, $v_6$, $v_7$} – {$v_2$} = {$v_3$, $v_4$, $v_5$, $v_6$, $v_7$}

k = 2 (dari langkah 3(a))

Diselidiki tiap titik dalam V-L

Untuk j = 3 :

D(j) = D(3) = 9 ; D(k) + W(k, j) = D(2) + W(2, 3) = 3 + ∞ = ∞

Karena D(3) ≯ D(2) + W(2, 3) maka D(3) tetap = 9

Untuk j = 4 :

D(j) = D(4) = ∞ ; D(k) + W(k, j) = D(2) + W(2,4) = 3 + 7 = 10

Karena D(4) > D(2) + W(2, 4), maka harga D(4) diubah menjadi 10

Hal ini berarti bahwa untuk mencapai titik $v_4$ dari $v_1$, jalur melalui $v_2$, yaitu : $v_1$ $v_2$ $v_4$ (D(2) + W(2, 4)) mempunyai bobot yang lebih kecil jika dibandingkan jalur langsung $v_1$ $v_4$ (D(4)).

Untuk j = 5 :

D(j) = D(5) = ∞ :

D(k) + W(k, j) = D(2) + W(2, 5) = 3 + 1 = 4

Karena D(5) > D(2) + W(2, 5), maka harga D(5) diubah menjadi 4.

Untuk j = 6 :

D(j) = D(6) = ∞;

D(k) + W(k, j) = D(2) + W(2, 6) = 3 + ∞ = ∞

Karena D(6) ≯ D(2) + W(2, 6), maka harga D(6) tetap seperti sebelumnya, yaitu ∞.

Untuk j = 7

D(j) = D(7) = ∞

D(k) + W(k, j) = D(2) + W(2, 7) = 3 + ∞ = ∞

Karena D(7) ≯ D(2) + W(2, 7), maka harga D(6) tetap seperti sebelumnya, yaitu ∞.

Langkah selanjutnya kembali ke langkah 3(a). Karena $v_7 \notin L$.

$V - L = \{v_3, v_4, v_5, v_6, v_7\} \neq \emptyset$.

Di antara D(k) (k = 3, 4, ..., 7) hasil iterasi langkah 3(b), D(k) yang terkecil adalah D(5), sehingga $v_k = v_5$.

Maka sekarang, $L = L \cup \{v_5\} = \{v_2\} \cup \{v_5\} = \{v_2, v_5\}$

3(b) : $V - L = \{v_2, v_3, v_4, v_5, v_6, v_7\} - \{v_2, v_5\} = \{v_3, v_4, v_6, v_7\}$

Langkah 3(b) untuk mengecek setiap titik dalam V – L diulangi lagi.

Langkah 3(a) dan 3(b) diulang-ulang terus hingga $v_7 \in L$. Hasil iterasi selengkapnya adalah sebagai berikut :

Tabel 2.2 Hasil Iterasi.

| Indeks k shg D(k) minimum | L | V – L | D(2) | D(3) | D(4) | D(5) | D(6) | D(7) |
|---|---|---|---|---|---|---|---|---|
| - | ø | {$v_2$, $v_3$, $v_4$, $v_5$, $v_6$, $v_7$} | W(1,2) = 3 | W(1,3) = 9 | W(1,4) = ∞ | W(1,5) = ∞ | W(1,6) = ∞ | W(1,7) = ∞ |
| 2 | {$v_2$} | {$v_3$, $v_4$, $v_5$, $v_6$, $v_7$} | 3 (tetap) | Min (D(3), D(3)+W(2,3)) = Min (9, 3 + ∞ ) = 9 | Min (D(4), D(2)+W(2,4)) = Min (∞, 3 + 7) = 10 | Min (D(5), D(2)+W(2,5)) = Min (∞, 3 + 1) = 4 | Min (D(6), D(2)+W(2,6)) = Min (∞, 3 + ∞) = ∞ | Min (D(7), D(2)+W(2,7)) = Min (∞, 3 + ∞) = ∞ |
| 5 | {$v_2$, $v_5$} | {$v_3$, $v_4$, $v_6$, $v_7$} | 3 (tetap) | Min (D(3), D(5)+W(5,3)) = Min (9, 4 + 4) = 8 | Min (D(4), D(5)+W(5,4)) = Min (10, 4 + 5) = 9 | 4 (tetap) | Min (D(6), D(5)+W(5,6)) = Min (∞, 4 + 9) = 13 | Min (D(7), D(5)+W(5,7)) = Min (∞, 4 + ∞) = ∞ |
| 3 | {$v_2$, $v_5$, $v_3$} | {$v_4$, $v_6$, $v_7$} | 3 (tetap) | 8 (tetap) | Min (D(4), D(3)+W(3,4)) = Min (9, 8 + 7) = 9 | 4 (tetap) | Min (D(6), D(3)+W(3,6)) = Min (13, 8 + ∞) = 13 | Min (D(7), D(3)+W(3,7)) = Min (∞, 8 + ∞) = ∞ |
| 4 | {$v_2$, $v_5$, $v_3$, $v_4$} | {$v_6$, $v_7$} | 3 (tetap) | 8 (tetap) | 9 (tetap) | 4 (tetap) | Min (D(6), D(4)+W(4,6)) = Min (13, 9 + 2) = 11 | Min (D(7), D(4)+W(4,7)) = Min (∞, 9 + 8) = 17 |
| 6 | {$v_2$, $v_5$, $v_3$, $v_4$, $v_6$} | {$v_7$} | 3 (tetap) | 8 (tetap) | 9 (tetap) | 4 (tetap) | 11 (tetap) | Min (D(7), D(6)+W(6,7)) = Min (17, 11 + 4) = 15 |
| 7 | {$v_2$, $v_5$, $v_3$, $v_4$, $v_6$, $v_7$} | | | | | | | |

Karena kondisi dari $v_n = v_7 \in L$, maka iterasi dihentikan. *Path* terpendek yang dihasilkan dari $v_1$ ke $v_7$ adalah 15, dengan jalur secara mundur sebagai berikut :

Pada iterasi indeks k = 6, diperoleh penurunan jarak pada kolom D(7) dari 17 ke 15.

Ini berarti titik yang terpilih pada baris tersebut ($v_6$) adalah jalur path (jalur $v_6 \rightarrow v_7$).

Berikutnya pada kolom D(6) diperoleh penurunan jarak dari 13 ke 11. Hal ini menunjukkan bahwa titik yang terdapat pada indeks k = 4 ($v_4$) adalah jalur *path* (jalur $v_4 \rightarrow v_6 \rightarrow v_7$).

Penurunan jarak tidak terjadi pada Kolom D(4) (dari 9 tetap 9). Hal ini menunjukkan bahwa titik yang terdapat pada indeks k = 3 ($v_3$) bukanlah jalur *path*. Naik satu baris di atasnya, terjadi penurunan jarak dari 10 ke 9 yaitu baris tersebut sesuai indeks k = 5. Jadi $v_5$ adalah jalur *path* (jarak $v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$).

Terjadi penurunan jarak dari ∞ ke 4 pada kolom D(5). Baris tersebut sesuai indeks k = 2. Jadi $v_2$ adalah jalur *path* (jalur $v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$).

*Path* terpendek yang dihasilkan adalah $v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7$ dengan total panjang = 15.

Hasil akhir yang didapatkan dari algoritma *dijkstra* adalah *path* terpendek dari titik $v_1$ kesemua titik lain. Untuk menemukan *path* terpendek dari semua titik, dilakukan sebuah iterasi dengan dimulai dari semua titik menggunakan algoritma ini.

**2.5 Penelitian Terkait**

Pada sub bab ini akan dijelaskan beberapa penelitian yang digunakan sebagai referensi dalam melakukan penelitian ini. Penelitian-penelitian yang terkait dengan penelitian ini akan di-*review* sehingga dapat menunjang dan sebagai acuan dalam melakukan penelitian ini.

Penelitian yang dilakukan (Yulia R, Istiadi, & Roqib, 2015) mengenai pencarian SPBU (Stasiun Pengisian Bahan Bakar) terdekat dan penentuan jarak terpendek di Kabupaten Jember menggunakan algoritma *dijkstra*. Penelitian ini membahas mengenai pemetaan lokasi, pencarian lokasi dan penentuan jalur terpendek menuju lokasi SPBU tersebut. Pencarian SPBU terdekat dipengaruhi oleh kriteria, *cost*, dan *reverse cost*. Di mana untuk jalan satu arah diberikan nilai *reverse cost* sebesar 1000000, sehingga jalan ini tidak akan pernah dipilih. Algoritma *dijkstra* sangat sesuai dan mudah digunakan pada studi kasus di penelitian ini. Penerapan algoritma *dijkstra* telah merekomendasikan jalur terpendek jarak tempuh guna memberikan efisiensi penggunaan bahan bakar kendaraan bermotor.

Penelitian mengenai pencarian rute terpendek juga dilakukan oleh (Purwananto, Purwitasari, & Wibowo, 2005), dalam implementasi dan analisis

algoritma pencarian rute terpendek di kota Surabaya. Pada pencarian rute terpendek, terdapat tiga algoritma yang digunakan dalam penelitian ini yakni algoritma *floyd warshall*, *dijkstra* dan *two queues*. Tiga algoritma ini akan diuji dan kemudian dievaluasi tingkat kecepatan dalam proses eksekusi berdasarkan perhitungan dengan faktor-faktor tertentu. Data uji coba yang dignuakan adalah data jaringan jalan di kota Surabaya. Kemudian disimpan dalam *database* menggunakan DBMS MySQL. Ketiga algoritma tersebut diterpakan dalam sebuah aplikasi dengan bentuk WEB dan WAP. Hal ini berguna untuk memudahkan pengguna dalam mendapatkan informasi dari rute terpendek yang diperoleh. Berdasarkan hasil uji coba, ketiga algoritma rute terpendek mempunyai lama waktu eksekusi yang dipengaruhi oleh jumlah *node* dalam jaringan dan *node* yang diperiksa. Algoritma *dijkstra* memiliki keunggalan dalam eksekusi waktu dengan catatan jumlah *node*-nya masih kurang dari 1000. Dengan hasil eksekusi yang mencapai kurang dari 1 detik. Sedangkan dengan jumlah *node* yang lebih dari 1000, algoritma *two queues* memiliki waktu eksekusi yang tercepat jika dibandingkan dengan algoritma lainnya.

Penelitian yang dilakukan (Saputro, 2013) mengenai pencarian rute terpendek peta wisata di Kota Manado berbasis *mobile web* menggunakan algoritma *dijkstra*. Tujuan dari penelitian ini adalah membangun sebuah Sistem Informasi Geografis, yang diharapkan dapat membantu wisatawan dari luar daerah kota Manado dalam menampilkan rute dari satu tempat wisata ke tempat wisata lain. Metode yang menjadi dasar dalam perancangan SIG ini adalah dengan Metode Graf, dengan menerapkan algoritma pencarian rute terpendek, algoritma *dijkstra*. Selain itu, sistem ini akan menggunakan *haversine* formula dalam mengalkulasikan jarak, baik

jarak antar satu tempat wisata ke tempat wisata lain, maupun jarak antara posisi *user* berada ke hotel-hotel yang berada di kota Manado. Hasil dari penelitian ini adalah aplikasi berbasis *mobile web* yang dapat menampilkan rute antar satu tempat wisata dengan tempat wisata lain, serta rute dari posisi *user* menuju posisi hotel-hotel di kota Manado.

Algoritma *dijkstra* juga digunakan oleh (Pugas, Somantri, & Satoto, 2011) dalam penelitiannya yang berjudul "Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra dan Astar (A*) pada SIG Berbasis Web untuk Pemetaan Pariwisata Kota Sawahlunto". Tujuan dari penelitian ini adalah untuk mengembangkan SIG beserta visualisasi data spasial sebagai promosi wisata Kota Sawahlunto. Algoritma *dijkstra* menggunakan prinsip *greedy*, di mana pada setiap langkah dipilih sisi dengan bobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Sedangkan algoritma *astar* (A*) merupakan sebuah format heuristik untuk menghitung efisiensi solusi optimal. Dari hasil pengujian melalui *host to host* dengan 5 kali percobaan titik awal dan tujuan, disimpulkan bahwa pencarian rute terpendek menggunakan algoritma *dijkstra* dan *astar* menghasilkan rute jalan yang sama. Perhitungan jarak rute jalan terpendek menggunakan algoritma *dijkstra* dan *astar* menghasilkan jarak yang sama. Perhitungan jarak rute jalan terpendek secara manual menggunakan peta analog berskala 1:40.000 sedikit berbeda dengan perhitungan menggunakan algoritma *dijkstra* dan *astar* yaitu memiliki selisih jarak rata-rata 0,016 km. Kecepatan pencarian rute terpendek menggunakan algoritma *dijkstra* berbeda dengan algoritma *astar* di mana *astar* lebih cepat untuk proses pencarian rute terpendek dengan selisih waktu rata-rata 40 ms.

Terdapat beberapa literatur/penelitian yang telah dipelajari mengenai *multiple regression*, salah satunya adalah penelitian yang dilakukan oleh (Rozikin & Solichin, 2017) tentang implementasi algoritma genetika dan regresi linear berganda untuk prediksi bahan makanan pada restoran cepat saji. Tujuan penelitian ini adalah menerapkan algoritma genetika untuk memprediksi seberapa banyak makanan yang terjual pada bulan selanjutnya, sehingga pihak restoran cepat saji dapat memperkirakan jumlah bahan makanan yang perlu dipersiapkan. Analisis regresi berganda pada penelitian ini digunakan untuk membuat prediksi dari perkiraan nilai dan intensitas hubungan dua variabel atau lebih. Hasil pengujian menunjukkan bahwa tingkat akurasi sistem prediksi yang dihasilkan sebesar 89,9% dan tingkat kesalahan sebesar 10,1%.

Penelitian mengenai regresi linier berganda untuk melakukan prediksi juga dilakukan oleh (Mustafar, 2011). Penelitian ini dilakukan untuk memprediksi keluaran pada ladang minyak. Tujuan penelitian ini adalah untuk menemukan variabel yang tepat dan akurat untuk memprediksi *output* pada ladang minyak karena banyak faktor yang mempengaruhi *output*-nya. Ada 8 variabel bebas yang digunakan di antaranya adalah jumlah total sumur, jumlah *startup* sumur, jumlah sumur baru, volume air yang disuntikkan tahun sebelumnya, kadar air minyak tahun sebelumnya, pemulihan persen tahun sebelumnya, produksi minyak tahun sebelumnya, pemulihan persen tahun sebelumnya, produksi minyak tahun sebelumnya. Karena terlalu banyak variabel yang mempengaruhi kinerja ladang *output* minyak, penulis menggunakan metode kuadrat terkecil untuk bisa meminimalkan jumlah variabel dengan menghilangkan variabel yang paling tidak penting. Dari hasil dan pembahasan itu empat faktor yang paling penting yang

mempengaruhi *output* ladang minyak yang ditentukan dan menunjukkan bahwa persentase kesalahan nilai prediksi dari *output* aktual hanya 4,57%.

Selain digunakan untuk prediksi, regresi linear berganda juga digunakan untuk sistem penentuan. Penelitian yang telah dilakukan (Wicaksono, 2016), yang menggunakan metode *multipe linear regression* dan *euclidean distance* untuk menentukan harga buah berdasarkan tingkat kematangan dan berat buah. Tujuan dari penelitian ini adalah untuk mengetahui tingkat akurasi yang dapat dihasilkan dengan metode *euclidean distance* dan *multiple linear regression* dalam menentukan tingkat kematangan dan berat buah sebagai dasar penentuan harga. Data yang digunakan adalah tiga jenis buah dengan tingkat kematangan yang berbeda. Kemudian data tersebut diolah saat proses *training*. Pada saat proses *training*, citra diambil dari data buah, ekstraksi fitur dilakukan pada dimensi, warna dan bentuk. Fitur tersebut disimpan bersamaan dengan jenis buah dan tingkat kematangannya. Bentuk *output* dari proses ini adalah sebuah angka taksiran yang digunakan sebagai nilai estimasi berat pada saat proses *testing*. Pada proses *testing* dilakukan perbandingan antara fitur warna RGB dan bentuk dengan data yang telah terdapat di dalam *database* dengan menggunakan metode *euclidean distance*. Kemudian berat buah dapat dihitung estimasinya menggunakan metode *multiple linear regression* dengan *input* berupa angka taksiran hasil dari proses *training*. Hasilnya, akurasi yang diperoleh pada saat identifikasi jenis buah dari 70 data adalah sebesar 80% dan akurasi sistem yang dihasilkan dari estimasi berat dengan keseluruhan sebesar 91,13%.

# BAB III

# METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai beberapa hal, yaitu analisis kebutuhan data, analisis perhitungan regresi linear berganda untuk menentukan bobot lintasan, penyelesaian masalah pencarian rute tercepat menggunakan algoritma *dijkstra* dengan bobot yang diperoleh dari perhitungan menggunakan metode analisis statistik *multiple linear regression*, desain sistem dan desain *interface*.

## 3.1 Analisis

Dalam sub bab ini terdapat dua analisis yang dilakukan yaitu analisis kebutuhan data dan analisis perhitungan regresi linear berganda.

### 3.1.1 Kebutuhan Data

Data yang didapatkan pada penelitian ini yaitu data jumlah kendaraan, jarak, waktu, hari, dan kecepatan yang didapatkan dari penelitian secara langsung pada jalan yang telah ditentukan adalah sebagai berikut :

1. Data nama jalan

   Pengambilan data dari nama jalan yang akan dilalui menuju Stasiun Kotabaru adalah didasarkan pada opsi rute dari Google *maps*.

2. Data jumlah kendaraan

   Pengambilan data jumlah kendaraan dilakukan dengan menghitung secara langsung jumlah kendaraan yang terdapat pada nama-nama jalan pada berdasarkan rute, waktu dan hari yang telah ditentukan. Kendaraan yang dihitung meliputi kendaraan bermotor maupun tidak bermotor.

3. Data jarak

Pengambilan data jarak didapatkan dari data yang terdapat pada Google *maps*. Jarak yang dimaksud adalah jarak dari setiap nama jalan yang telah diberikan oleh Google *maps*.

4. Data waktu

   Pengambilan data waktu berdasarkan data waktu yang telah ditentukan oleh penulis. Asumsi yang digunakan pada parameter waktu adalah rentang waktu dari pukul 06.00 WIB sampai pukul 18.00 WIB. Rentang waktu tersebut dibagi menjadi lima bagian. Rentang waktu pertama dimulai dari jam 06.00-08.00 WIB. Kemudian, rentang waktu kedua dimulai dari pukul 08.01-11.00 WIB. Rentang waktu ketiga dimulai dari pukul 11.01-13.00 WIB. Rentang waktu keempat dimulai dari pukul 13.01-16.00 WIB. Dan rentang waktu terakhir dimulai dari pukul 16.01-18.00 WIB.

5. Data hari

   Untuk parameter hari meliputi jumlah hari dalam seminggu seperti pada umumnya yaitu Senin, Selasa, Rabu, Kamis, Jumat, Sabtu, dan Minggu.

6. Data kecepatan

   Pengambilan data kecepatan dilakukan dengan survei yang dilakukan secara langsung dengan berkendara melalui setiap jalan pada rute yang telah ditentukan. Data kecepatan diambil dari kecepatan rata-rata sepeda motor yang digunakan ketika melalui jalan pada rute yang telah ditentukan.

Pengambilan data jumlah kendaraan dan kecepatan dilakukan oleh minimum dua orang. Orang pertama bertugas untuk mengambil data jumlah kendaraan. Sedangkan orang kedua bertugas sebagai *driver*, yakni berkendara melalui jalan pada rute yang telah ditentukan dengan tujuan untuk mendapatkan data kecepatan

rata-rata dari sepeda motor tersebut. Pengambilan dua data tersebut dilakukan secara bersamaan pada waktu yang sama. Sehingga diperoleh data estimasi waktu. Data estimasi waktu ini merupakan waktu tempuh yang dibutuhkan oleh orang kedua dari titik awal menuju ke titik tujuan dengan kecepatan rata-rata dan jumlah kendaraan yang telah didapatkan.

Pada pengumpulan data yang dilakukan secara langsung (data primer) yakni data jumlah kendaraan dan kecepatan, dibuat sebuah *tools* sederhana untuk memudahkan dalam pengumpulan data tersebut. *Tools* tersebut berguna untuk mencatat jumlah kendaraan, waktu keberangkatan dari sepeda motor yang digunakan oleh *driver*, waktu ketika *driver* sudah sampai tujuan dan estimasi waktu yang dibutuhkan oleh *driver*.

### 3.1.2 Perhitungan Regresi Linear Berganda

Data yang digunakan pada perhitungan ini adalah data yang diambil dari data *training* dari *database*. Data yang akan dihitung berupa data jumlah kendaraan, kecepatan, dan estimasi waktu.

Tabel 3.1 Tabel Data

| Data ke | Jumlah Kendaraan | Kecepatan (km/jam) | Estimasi Waktu (Menit) | |
|---|---|---|---|---|
| | X1 | X2 | Y (desimal) | Y (jam) |
| 1 | 41 | 45 | 0,7833333 | 00:00:47 |
| 2 | 32 | 40 | 0,9666667 | 00:00:58 |
| 3 | 28 | 40 | 0,9333333 | 00:00:56 |
| 4 | 51 | 40 | 0,9 | 00:00:54 |
| 5 | 31 | 40 | 0,9666667 | 00:00:58 |
| 6 | 53 | 40 | 0,7666667 | 00:00:46 |
| 7 | 46 | 45 | 0,7833333 | 00:00:47 |
| 8 | 38 | 40 | 0,8166667 | 00:00:49 |
| 9 | 44 | 45 | 0,8166667 | 00:00:49 |
| | 364 | 375 | 7,7333333 | |

Data estimasi waktu dikonversikan menjadi desimal. Hal ini bertujuan agar data estimasi waktu dapat dijadikan parameter untuk membuat sebuah model regresi linear berganda.

Langkah pertama untuk melakukan perhitungan pada regresi linear berganda adalah mendeklarasikan data yang dibutuhkan.

Tabel 3.2 Deklarasi Data.

| **RUMUS = (X'.X)-1(X'.Y)** | | | |
|---|---|---|---|
| | **X** | | **Y** |
| 1 | 41 | 45 | 0,7833333 |
| 1 | 32 | 40 | 0,9666667 |
| 1 | 28 | 40 | 0,9333333 |
| 1 | 51 | 40 | 0,9 |
| 1 | 31 | 40 | 0,9666667 |
| 1 | 53 | 40 | 0,7666667 |
| 1 | 46 | 45 | 0,7833333 |
| 1 | 38 | 40 | 0,8166667 |
| 1 | 44 | 45 | 0,8166667 |

Selanjutnya pada langkah kedua adalah mengubah variabel X menjadi X transpose (X').

Tabel 3.3 Tabel X *transpose*.

| **X'** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 41 | 32 | 28 | 51 | 31 | 53 | 46 | 38 | 44 |
| 45 | 40 | 40 | 40 | 40 | 40 | 45 | 40 | 45 |

Selanjutnya pada langkah ketiga adalah menghitung perkalian antara X transpose (X') dan X.

Tabel 3.4 Perkalian X *transpose* dan X.

| **X'.X** | | |
|---|---|---|
| 9 | 364 | 375 |
| 364 | 15356 | 15215 |
| 375 | 15215 | 15675 |

Selanjutnya pada langkah keempat adalah menghitung invers dari hasil perkalian X *transpose* dan X.

Tabel 3.5 Hasil *inverse*.

| $(X'.X)^{-1}$ | | |
|---|---|---|
| 34,83338 | -0,00028 | -0,8330591 |
| -0,00028 | 0,001702 | -0,0016454 |
| -0,83306 | -0,00165 | 0,0215905 |

Selanjutnya pada langkah kelima adalah mensubstitusikan hasil perhitungan ke rumus awal $(X'.X)^{-1}(X'.Y)$.

Tabel 3.6 Tabel substitusi.

| $(X'.X)^{-1}$ | | | X'Y |
|---|---|---|---|
| 34,83338 | -0,00028 | -0,8330591 | 7,7333333 |
| -0,00028 | 0,001702 | -0,0016454 | 308,68333 |
| -0,83306 | -0,00165 | 0,0215905 | 321,25 |

Pada langkah terakhir adalah melakukan perkalian matriks antara $(X'.X)^{-1}$ dan X'Y. Dari hasil perkalian tersebut maka akan didapatkan hasil estimasi parameter regresi linear berganda.

Tabel 3.7 Tabel estimasi parameter regresi.

| Hasil | | |
|---|---|---|
| **a** | 1,6703373 | **Intercept** |
| **b1** | -0,005357 | **Koefisien 1** |
| **b2** | -0,014266 | **Koefisien 2** |

Kemudian setelah diketahui estimasi parameternya, tinggal memasukkan data yang digunakan sebagai *testing*. Misalkan data jumlah kendaraan adalah 30 dan data kecepatannya adalah 45. Dengan perhitungan bobot menggunakan regresi linear berganda berdasarkan persamaan 3.1 hasilnya dapat dilihat pada Tabel 3.8.

$$\hat{Y}=a+b_1x_1+b_2x_2 \qquad (3.1)$$

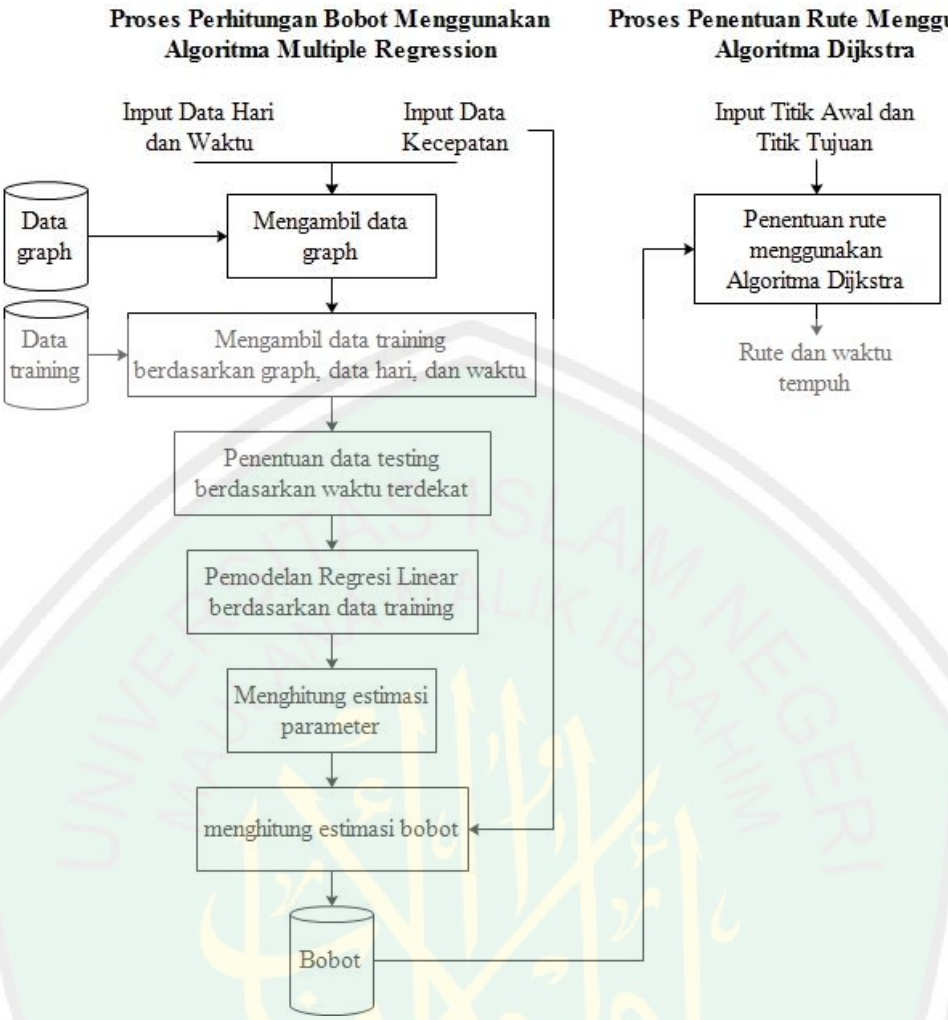
Tabel 3.8 Hasil estimasi bobot.

| a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|
| 1,670337 | -0,005357 | -0,014266 | 30 | 45 |
| **Desimal** | | | 0,867656422 | |
| **Jam** | | | 00:00:52 | |

### 3.2 Desain Sistem

Desain dari sistem yang dirancang ini, terdapat dua proses yang akan dilakukan dalam penentuan rute tercepat, yaitu proses perhitungan bobot dan proses penentuan rute tercepat. Proses *training* dilakukan untuk mendapatkan nilai-nilai parameter yang akan dijadikan acuan untuk membentuk sebuah persamaan regresi. Hal ini memungkinkan untuk memprediksi bobot (estimasi waktu) dari *vertex* satu ke *vertex* yang lainnya. Kemudian setelah mendapatkan sebuah persamaan regresi akan dilakukan sebuah perhitungan menggunakan persamaan regresi linier berganda yang akan menghasilkan bobot (estimasi waktu). *Output* proses *training* akan disimpan ke dalam *database*. Kemudian proses testing digunakan untuk membandingkan *input*-an dengan data yang ada dalam *database* sehingga menghasilkan sebuah *graph*. Setelah *graph* didapatkan, maka selanjutnya akan dilakukan penentuan rute tercepat dari *vertex* awal ke *vertex* terakhir menggunakan algoritma *dijkstra* sehingga akan menghasilkan rute dan waktu tempuh.

Blok diagram berikut adalah alur dari sistem, baik dari proses perhitungan bobot maupun proses penentuan rute mulai dari *input* sampai proses *output* yang ditunjukkan pada Gambar 3.1.




Gambar 3.1 Blok diagram sistem.

*Input* dari proses penentuan bobot berupa data kecepatan, hari dan waktu diperoleh dari pengguna. Data kecepatan diperoleh dari pengguna yang menentukan kecepatan berkendara menuju ke stasiun Kotabaru. Data hari dan waktu diambil dari hari dan waktu ketika penggunaan aplikasi. Proses selanjutnya adalah mengambil data *graph* dari *database*. Data *graph* digunakan sebagai acuan pengambilan data *training*. Kemudian, data *training* yaitu hasil pengumpulan data, diambil dari *database*. Selain menggunakan data *graph* sebagai acuan, data *training* juga diambil berdasarkan *input* data hari dan waktu. Kemudian data *training* yang akan digunakan dalam pemodelan regresi linear berganda ditentukan berdasarkan waktu terdekat dari *input* data waktu.

Pada pemodelan regresi linear berganda, data *training* yang diambil adalah jumlah kendaraan, kecepatan dan estimasi waktu. Jumlah kendaraan dan kecepatan sebagai parameter X1 dan X2 sedangkan estimasi waktu sebagai parameter Y. Kemudian dilakukan proses perhitungan estimasi parameter. Setelah estimasi parameter diperoleh, data *testing* dimasukkan yaitu berupa *input* data kecepatan. Terakhir, dihitung estimasi bobot tersebut kemudian disimpan dalam *database*.

Pada proses penentuan rute tercepat menggunakan algoritma *dijkstra* data yang digunakan sebagai *input* adalah data titik awal dan tujuan. Kemudian rute tercepat dihitung berdasarkan bobot dari titik dan tujuan. *Output* yang dihasilkan berupa rute tercepat dan prediksi waktu yang harus ditempuh.

### 3.2.1 Proses Penentuan Bobot Menggunakan Algoritma *Multiple Regression*.

Data yang diperoleh dari hasil penelitian secara langsung meliputi jumlah kendaraan, waktu, hari, dan kecepatan. Adapun pengolahan data yang akan dilakukan adalah sebagai berikut :

1. Membuat Model Regresi Linier Berganda

   Pada tahap yang pertama diperlukan adanya sebuah model dari regresi linier berganda untuk mengetahui variabel-variabel yang akan digunakan. Pada regresi linier berganda terdapat sejumlah variabel bebas yang dihubungkan dengan $Y$ linier atau berpangkat satu dalam semua variabel bebas. Jika variabel bebas itu adalah $X_1, X_2, \ldots, X_n$ dan variabel tak bebasnya adalah $Y$, maka model umum pada regresi linier berganda adalah sebagai berikut :

   $$\hat{Y}=\beta_0+\beta_1 X_1+\beta_2 X_2 \qquad (3.2)$$

   Di mana :

   $\hat{Y}$ = nilai estimasi waktu

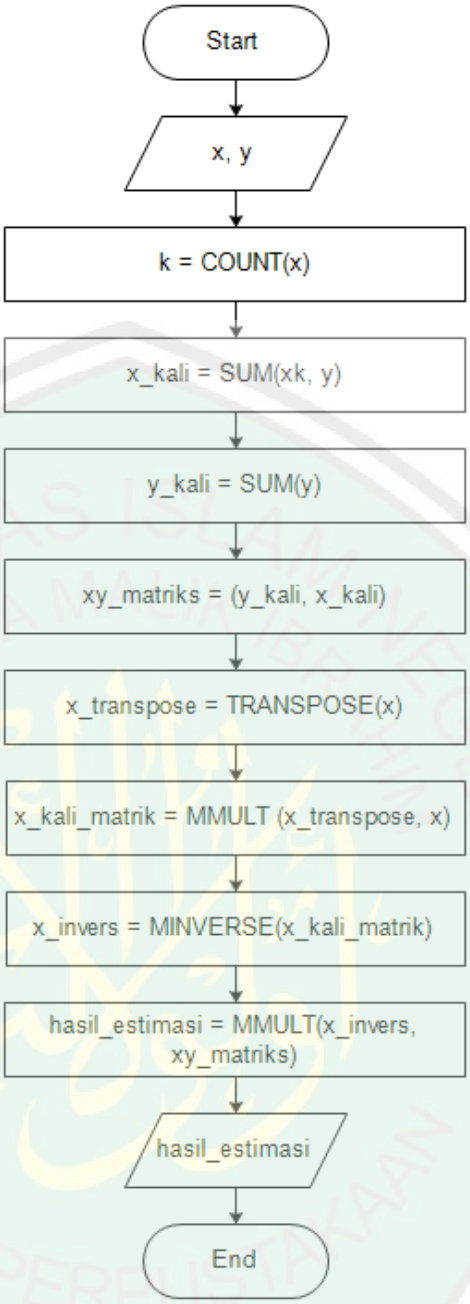
$\beta_0, \beta_1, \beta_1$ = nilai taksiran dari hasil estimasi parameter

$X_1, X_2$ = nilai variabel bebas di mana $X_1$ adalah jarak dan $X_2$ adalah kecepatan.

2. Menghitung Estimasi parameter

Pada dasarnya terdapat 2 jenis variabel yang saling berhubungan dan saling mempengaruhi, yaitu variabel bebas dan variabel tak bebas. Di mana data berupa jumlah kendaraan dan kecepatan adalah sebagai variabel bebas, sedangkan estimasi waktu adalah sebagai variabel tak bebas.

*Output* dari proses estimasi parameter adalah angka taksiran yang akan di jadikan sebuah persamaan regresi, yaitu sebuah persamaan yang memungkinkan untuk memprediksi nilai-nilai variabel tak bebas dari nilai-nilai variabel bebas. Tahap estimasi parameter dijelaskan pada *flowchart* yang ditunjukkan pada Gambar 3.2.



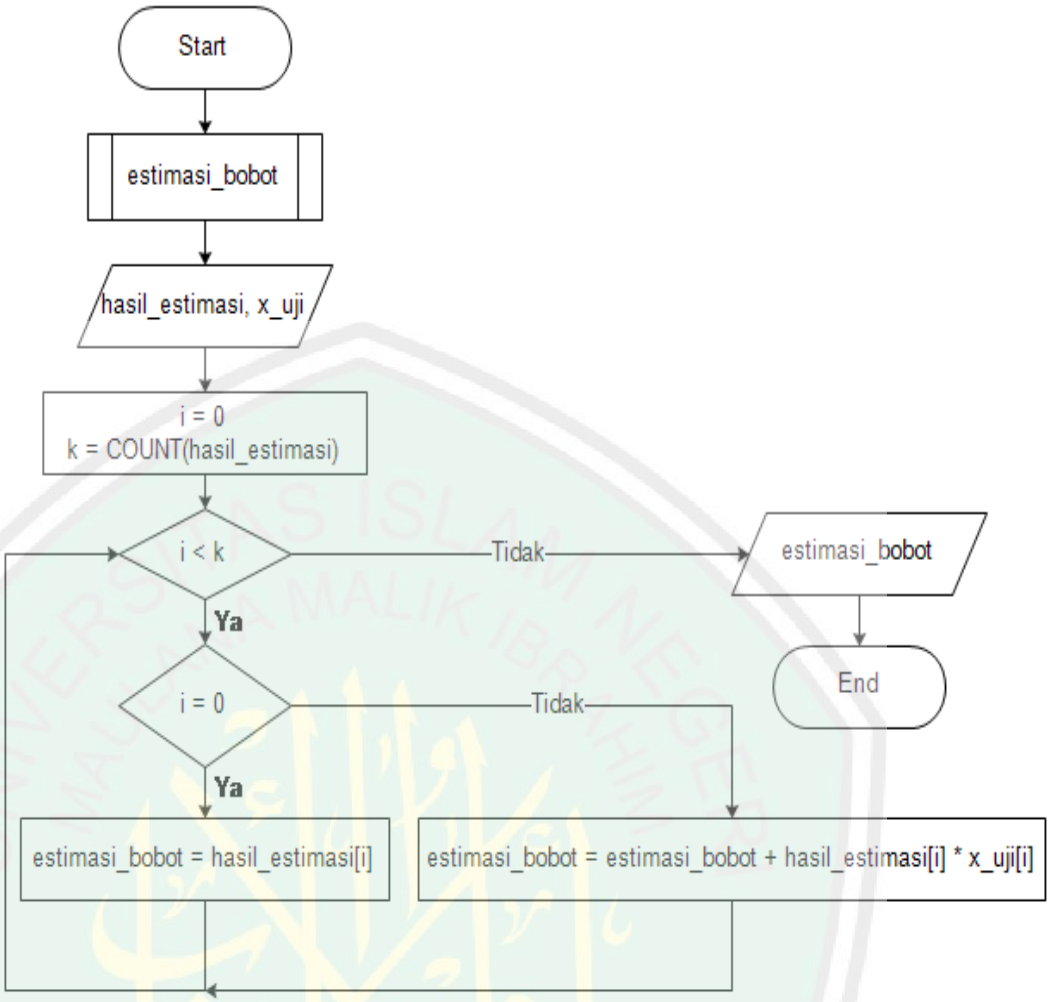

Gambar 3.2 *Flowchart* tahap estimasi parameter

Berdasarkan Gambar 3.2 tahap estimasi parameter dimulai dari *input* x berupa variabel bebas dan y sebagai variabel tak bebas. Karena menggunakan regresi linier berganda di mana variabel bebas yang digunakan lebih dari satu, maka *input*-an x berupa matriks. k merupakan variabel yang berisi jumlah variabel bebas. x_kali merupakan variabel yang berisi hasil penjumlahan dari seluruh data. Di mana data yang dijumlahkan

adalah hasil perkalian dari setiap variabel bebas dengan variabel tak bebas. y_kali merupakan hasil penjumlahan seluruh data pada variabel tak bebas. xy_matriks adalah matriks dari hasil penggabungan x_kali dan y_kali. x_transpose merupakan penukaran elemen matriks dari baris menjadi kolom dan kolom menjadi baris. x_kali_matrik merupakan perkalian dua buah matrik x_transpose dan x. x_invers merupakan hasil dari pencarian invers matriks dari x_kali_matriks. Sedangkan hasil_estimasi merupakan variabel yang berisi perkalian dua buah matriks x_invers dan xy_matriks. *Output* atau keluaran dari proses estimasi parameter ini berupa variabel hasil_estimasi, dimana variabel ini berisi angka taksiran berupa matriks untuk digunakan pada persamaan regresi yang akan diproses pada tahap perhitungan estimasi bobot lintasan.

3. Menghitung Estimasi Bobot Lintasan

   Dalam menghitung estimasi bobot lintasan digunakan parameter pada data berupa jumlah kendaraan, estimasi waktu dan kecepatan. Bobot lintasan dapat di estimasi dengan memasukkan variabel dari hasil proses estimasi parameter sebelumnya. Dengan persamaan regresi yang telah diproses pada tahap pemodelan regresi linier berganda, estimasi bobot lintasan dapat di estimasi. Berikut ini merupakan *flowchart* tahap estimasi bobot lintasan dengan model regresi linier berganda.




Gambar 3.3 *Flowchart* tahap estimasi bobot

Estimasi_bobot merupakan variabel yang didefinisikan yang nantinya akan digunakan untuk menampung hasil dari tahap ini. *Input* pada proses ini adalah hasil_estimasi dan x_uji, dimana hasil_estimasi didapatkan dari proses estimasi parameter sebelumnya dan x_uji adalah data yang akan diuji. Variabel i adalah variabel untuk iterasi dan k adalah variabel yang berisi jumlah variabel bebas. Kemudian estimasi bobot lintasan akan di estimasi menggunakan persamaan regresi yang telah ditentukan pada proses sebelumnya. *Output* pada proses ini berupa hasil estimasi bobot yang berada pada variabel estimasi_bobot. Berikut persamaan regresi yang dimaksud :

$$estimasi\_bobot = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \qquad (3.3)$$

Dimana :

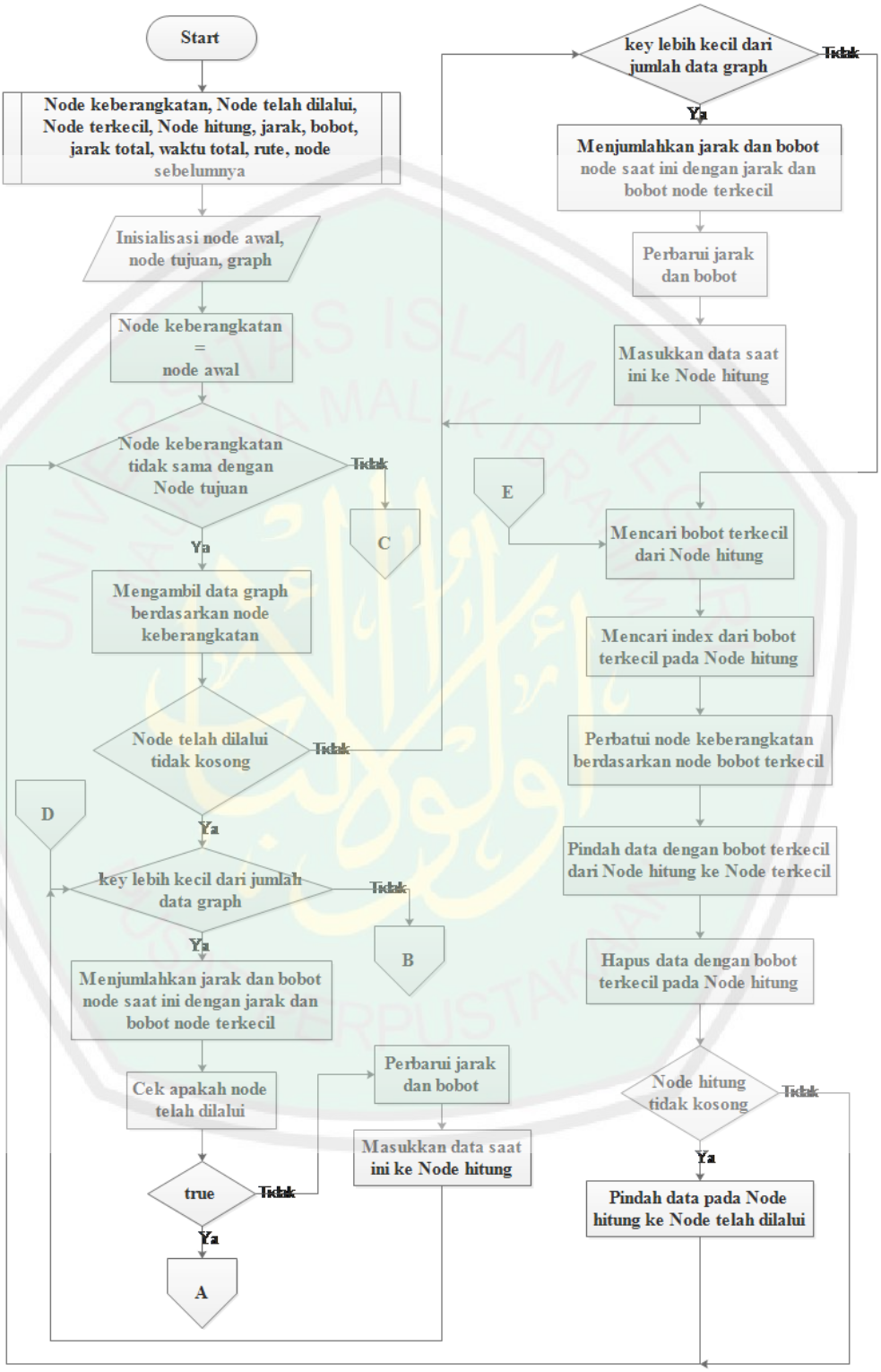
| | | |
|---|---|---|
| *estimasi_bobot* | = | nilai estimasi waktu |
| $\beta_0, \beta_1, \beta_1$ | = | nilai taksiran dari hasil estimasi parameter yang terdapat pada variabel hasil_estimasi |
| $X_1, X_2$ | = | nilai variabel bebas pada variabel x_uji, dimana $X_1$ adalah jarak dan $X_2$ adalah kecepatan |

### 3.2.2 Proses Penentuan Rute Menggunakan Algoritma Dijkstra

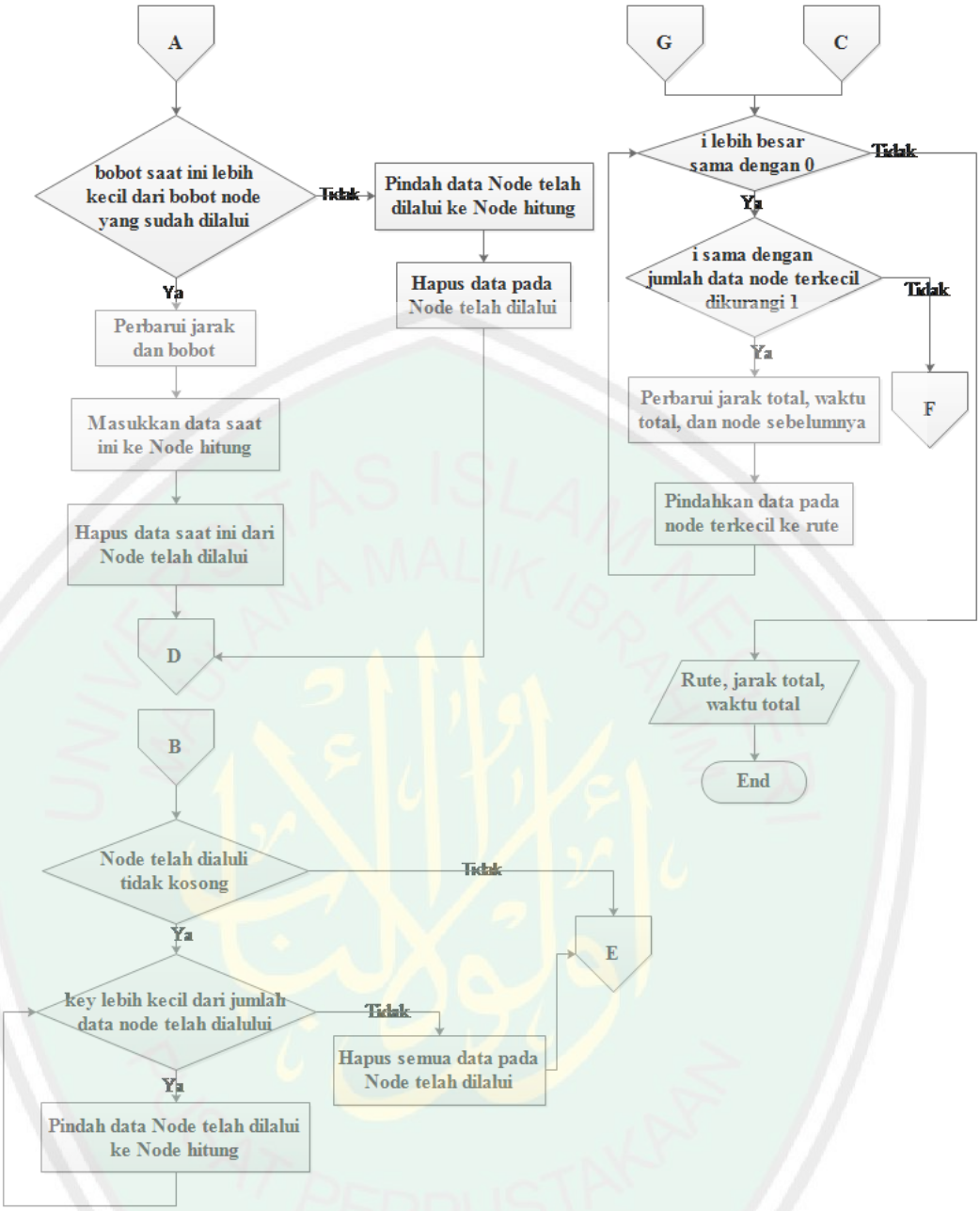
Pada proses selanjutnya akan dilakukan penentuan rute dan waktu tempuhnya. Penentuan rute dengan menggunakan algoritma *dijkstra* didasari oleh *cost* atau bobot antar pasangan *vertices*. Bobot dari setiap pasangan *vertices* diperoleh dari data pada proses estimasi waktu tempuh yang tersimpan dalam database. Dengan algoritma *dijkstra* dapat diperoleh solusi optimum rute tercepat berdasarkan waktu tempuh yang menjadi bobot dari setiap pasangan *vertices*. Berikut *flowchart* pada tahap proses penentuan rute dengan algoritma *dijkstra* seperti diperlihatkan pada Gambar 3.4 berikut ini.

Pada tahap ini *input*-an berupa *node* awal, *node* akhir, kecepatan, dan waktu. Di mana *node* awal adalah lokasi awal *user* dan *node* akhir adalah lokasi tujuan *user*. Pada proses selanjutnya dilakukan pencarian bobot berdasarkan kecepatan, hari, dan waktu pada *database*. Setelah bobot ditemukan maka akan dilakukan pelabelan sementara pada bobot antar *node*. Lalu akan dilakukan pencarian bobot minimum yang dilakukan dari *node* awal. Setelah bobot minimum didapatkan, maka akan dilakukan pelabelan bahwa *node* sudah dilewati. Selanjutnya dilakukan pencarian bobot minimum dari *node* yang sudah diberi label dan membandingkan dengan *node* lainnya. Jika *node* sudah dilewati maka akan dilakukan pelabelan pada

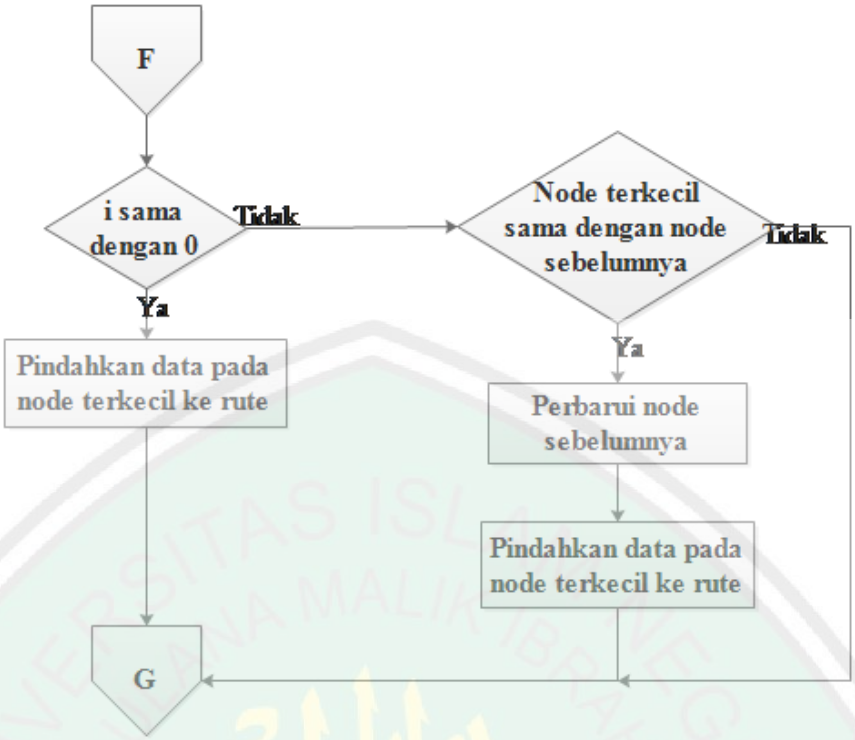
*node*, jika *node* belum dilewati maka akan dilakukan pemeriksaan bobot pada *node* tersebut.




Gambar 3.4 *Flowchart* penentuan rute dengan algoritma *dijkstra*.




Gambar 3.4 *Flowchart* penentuan rute dengan algoritma *dijkstra* (sambungan).




Gambar 3.4 *Flowchart* penentuan rute dengan algoritma *dijkstra* (sambungan).

Jika bobot baru lebih kecil dari bobot lama maka akan dilakukan pembaruan pada bobot, namun jika bobot baru lebih besar dari bobot lama maka akan dilakukan perbandingan bobot pada tiap *node*. Setelah itu dilakukan penerapan bobot minimum pada tiap *node* maka akan dilakukan pemeriksaan apakah *node* tersebut adalah *node* terakhir. Jika *node* bukan *node* terakhir maka kembali pada proses pencarian bobot minimum berikutnya dan membandingkan antar *node*, namun jika *node* tersebut adalah *node* terakhir maka rute tercepat telah didapatkan dan proses telah selesai.

### 3.2.3 *Inteface*

Aplikasi yang dibuat ini berbasis *mobile* Android. Desain *interface* yang digunakan berikut sesuai dengan tampilan *interface* Android. Terdapat 2 *activity* yang digunakan, yaitu *activity* untuk *input* data dan *activity* untuk memproses dan menampilkan rute. Berikut ini adalah desain *interface* yang telah dibuat.

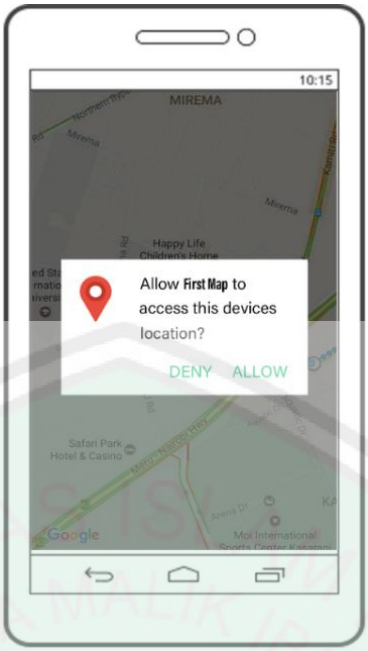
### 3.2.3.1 Desain *Activity Input*

Sebelum mendapatkan fitur rute tercepat, pengguna harus memilih beberapa *input* yang dibutuhkan untuk menentukan rute. *Input* tersebut berupa lokasi tujuan, dan kecepatan. Tampilan desain *activity input* ditunjukkan pada Gambar 3.5 berikut ini.


Gambar 3.5 *User interface activity input.*

### 3.2.3.2 Desain *Activity* Rute

Tampilan desain *activity* rute adalah tempat di mana *output* rute akan ditampilkan. Pada *activity* ini proses *training* akan berjalan. Berikut tampilan desain *activity* rute.




Gambar 3.6 *User interface activity* rute

Pada Gambar 3.6 tersebut terlihat bahwa jika *user* belum mengaktifkan fitur lokasi maka akan muncul permintaan untuk menghidupkannya. Lalu setelah lokasi hidup maka akan muncul rute tercepat menuju titik akhir yang dimasukkan seperti terlihat pada Gambar 3.7 berikut ini.


Gambar 3.7 *User interface* activity rute terpendek.

# BAB IV

# UJI COBA DAN PEMBAHASAN

Bab ini membahas tentang implementasi dari sistem yang telah dijabarkan pada sub bab sebelumnya dan langkah uji coba yang akan dilakukan . Kemudiann dijelaskan mengenai hasil dari uji coba tersebut dan pembahasannya. Serta, pemaparan integrasi antara penelitian ini dengan kajian Alquran dan Hadis.

## 4.1 Implementasi sistem

Sebelum dilakukan uji coba sistem, terlebih dahulu dijelaskan implementasi dari sistem yang telah dirancang. Aplikasi dibangun dengan menggunakan *algoritma dijsktra* dan metode *multiple regression*. Implementasi sistem tersebut dapat diuraikan sebagai berikut ini :

### 4.1.1 Proses Pengumpulan dan Pengolahan Data

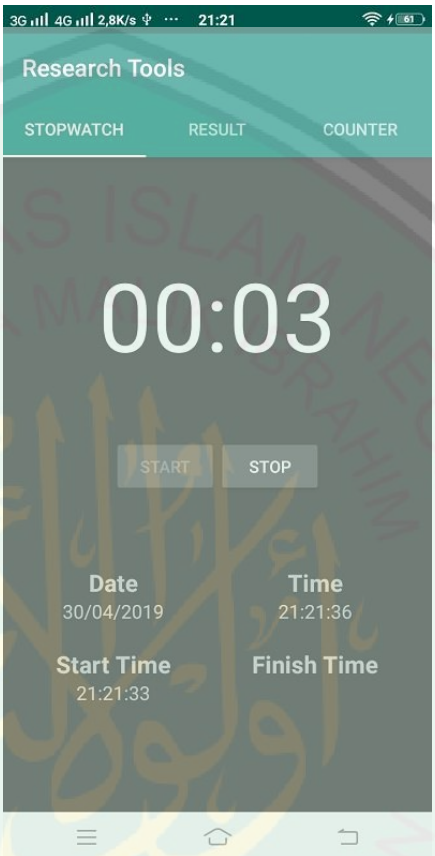
Proses pengumpulan data yang dibutuhkan untuk penelitian ini dilakukan dengan menggunakan sebuah *tools* sederhana yang dirancang untuk memudahkan proses tersebut. Kemudian setelah data didapatkan, data tersebut dimasukkan ke dalam sebuah *database* melalui halaman pengolahan data. Dalam hal ini, proses pengumpulan data dapat diuraikan sebagai berikut ini :

#### 4.1.1.1 *Tools* pengumpulan data

Pengumpulan data secara langsung jika dilakukan dengan cara yang manual akan memperlama waktu penelitian. Hal ini dikarenakan data yang dibutuhkan seperti waktu *start driver*, waktu *driver* sampai ke tujuan dan estimasi waktu jika dilakukan pencatatan manual akan membutuhkan waktu yang cukup lama. Maka dari itu, dirancang sebuah *tools* untuk memudahkan pengumpulan data tersebut.

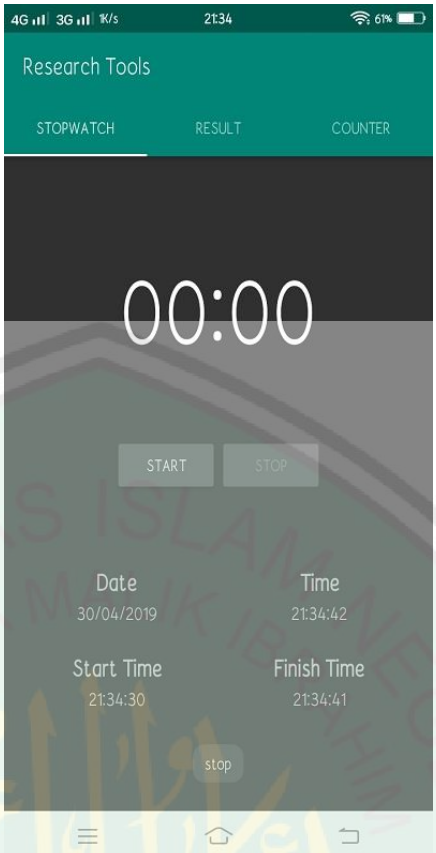
*Tools* ini dirancang dengan berbasis *mobile*. Sehingga waktu pencatatan data lebih cepat daripada pencatatan secara manual. Berikut ini adalah tampilan dari *tools* berserta penjelasannya.

a. Tampilan untuk *driver*




Gambar 4.1 Tampilan *driver* ketika *start*.

Pada gambar 4.1 terdapat sebuah *stopwatch* yang berguna untuk menghitung estimasi waktu yang dibutuhkan *driver* dari lokasi mulai sampai ke tujuan. Ketika *driver* menekan tombol start, maka otomatis *stopwatch* akan berjalan, tombol *start* otomatis *disable* dan waktu mulai akan dicatat di *start time*. Kemudian ketika sampai di tujuan *driver* hanya harus menekan tombol *stop* agar *stopwatch* berhenti, otomatis *stopwatch* di *reset* kembali ke awal, tombol *stop* otomatis *disable* dan data waktu *finish* akan terlihat di *finish time*. Seperti terlihat pada gambar 4.2 berikut ini.




Gambar 4.2 Tampilan *driver* ketika *finish*

b. Tampilan untuk pencatat jumlah kendaraan

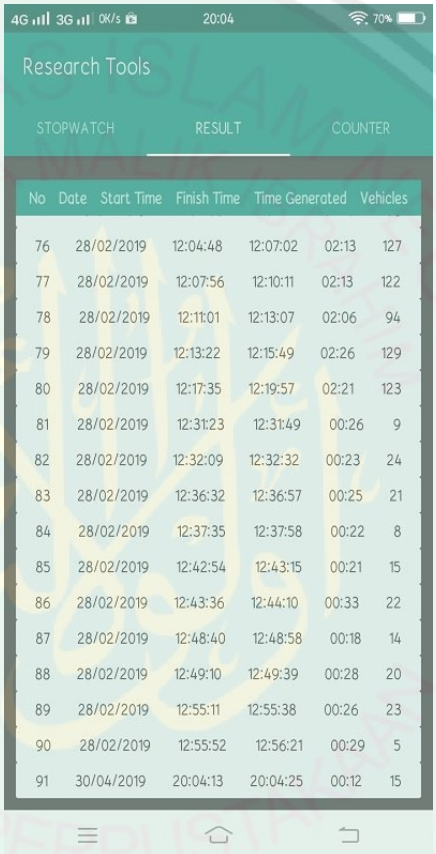
Pada tampilan ini, terdapat sebuah tombol *counter* untuk menghitung jumlah kendaraan yang melintasi jalan tersebut. Ketika *driver* menekan tombol *start* maka otomatis akan mengirim sebuah notifikasi dan otomatis hasil *counting* akan berwarna biru yang menandakan bahwa pencatat kendaraan bisa menghitung jumlah kendaraan. Seperti terlihat pada gambar 4.3 berikut ini. Kemudian ketika *driver* menekan tombol *stop* maka otomatis *tools* akan mengirimkan sebuah notifikasi kepada pencatat kendaraan dan otomatis proses *counting* akan berhenti. Hal ini ditandai dengan tulisan jumlah kendaraan yang berwarna merah, seperti yang terlihat pada Gambar 4.4.


Gambar 4.3 Tampilan pencatat kendaraan ketika *counting*.


Gambar 4.4 Tampilan pencatat kendaraan ketika *stop counting*.

c. Tampilan tabel pengumpulan data

Setelah *driver* menekan tombol *stop* dan pada tampilan pencatat kendaraan *counter* berhenti, maka pencatat kendaraan diharuskan untuk menekan tombol *upload* agar data tersebut masuk ke dalam *database*. Kemudian hasil dari pengumpulan data tersebut dapat dilihat di tampilan *result* yang ditunjukkan pada Gambar 4.5 berikut.




Gambar 4.5 Tampilan hasil pengumpulan data

*Database* pengumpulan data ini akan sementara disimpan dalam *fire**base***. Kemudian data tersebut akan diekspor ke dalam *file excel* untuk memudahkan dalam pengelompokan data berdasarkan hari dan waktu dalam satu jalan. Seperti pada Tabel 4.1 berikut ini yang merupakan hasil dari pengumpulan data pada hari kamis di jalan Sumbersari.

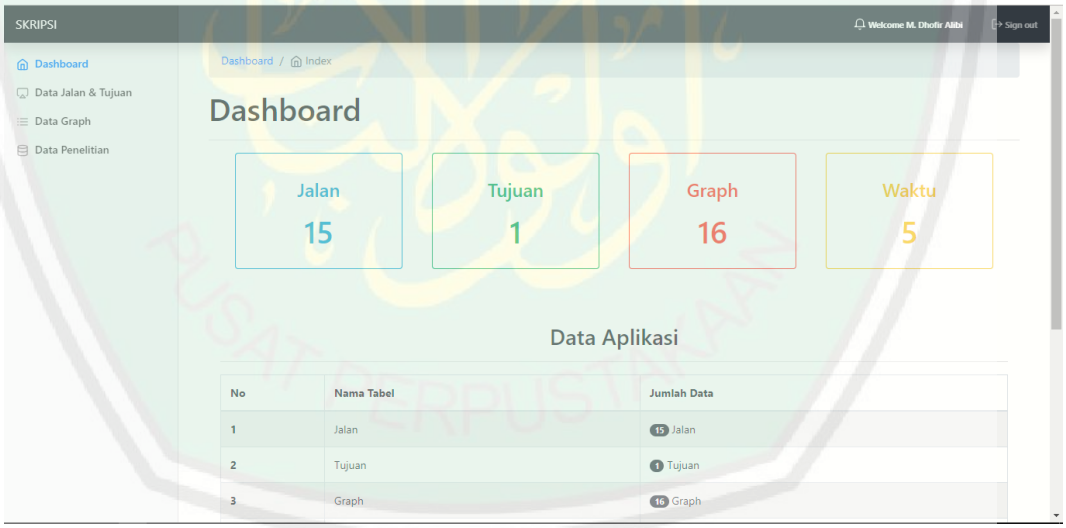
Tabel 4.1 Hasil pengumpulan data hari Kamis jalan Sumbersari

| Jam | No | Jam | | Jumlah Kendaraan | Kecepatan | Waktu Tempuh |
|---|---|---|---|---|---|---|
| | | Start | Finish | | | |
| 6 - 7 | 1 | 06.00.59 | 06.02.03 | 45 | 45 | 00.01.04 |
| | 2 | 06.02.12 | 06.03.33 | 43 | 40 | 00.01.21 |
| | 3 | 06.03.41 | 06.05.11 | 27 | 40 | 00.01.30 |
| | 4 | 06.05.18 | 06.06.54 | 57 | 35 | 00.01.36 |
| | 5 | 06.07.03 | 06.08.30 | 39 | 35 | 00.01.27 |
| | 6 | 06.08.36 | 06.10.05 | 32 | 40 | 00.01.29 |
| | 7 | 06.10.12 | 06.11.19 | 41 | 40 | 00.01.07 |
| | 8 | 06.11.28 | 06.13.01 | 47 | 35 | 00.01.33 |
| | 9 | 06.13.15 | 06.14.43 | 30 | 40 | 00.01.28 |
| | 10 | 06.14.50 | 06.15.57 | 41 | 40 | 00.01.07 |
| 8 - 9 - 10 | 1 | 09.10.42 | 09.12.27 | 65 | 25 | 00.01.44 |
| | 2 | 09.15.07 | 09.16.41 | 48 | 35 | 00.01.34 |
| | 3 | 09.19.36 | 09.21.35 | 92 | 25 | 00.01.58 |
| | 4 | 09.23.43 | 09.24.51 | 73 | 55 | 00.01.08 |
| | 5 | 09.27.30 | 09.30.09 | 94 | 15 | 00.02.39 |
| | 6 | 09.33.43 | 09.35.35 | 81 | 30 | 00.01.51 |
| | 7 | 09.38.40 | 09.40.34 | 77 | 30 | 00.01.53 |
| | 8 | 09.41.59 | 09.44.12 | 81 | 25 | 00.02.13 |
| | 9 | 09.47.09 | 09.48.56 | 70 | 25 | 00.01.46 |
| | 10 | 09.51.38 | 09.53.12 | 63 | 35 | 00.01.33 |
| 11 - 12 | 1 | 11.09.31 | 11.11.17 | 69 | 40 | 00.01.46 |
| | 2 | 11.15.09 | 11.16.44 | 57 | 40 | 00.01.34 |
| | 3 | 11.20.15 | 11.22.03 | 76 | 40 | 00.01.48 |
| | 4 | 11.30.13 | 11.31.43 | 39 | 45 | 00.01.29 |
| | 5 | 11.38.17 | 11.39.47 | 74 | 40 | 00.01.30 |
| | 6 | 11.43.04 | 11.44.36 | 66 | 40 | 00.01.32 |
| | 7 | 11.48.20 | 11.49.39 | 56 | 45 | 00.01.18 |
| | 8 | 11.53.02 | 11.54.45 | 62 | 40 | 00.01.42 |
| | 9 | 11.58.00 | 11.59.29 | 62 | 40 | 00.01.28 |
| | 10 | 12.03.23 | 12.05.04 | 57 | 40 | 00.01.41 |
| 13 - 14 - 15 | 1 | 13.00.12 | 13.01.42 | 36 | 40 | 00.01.29 |
| | 2 | 13.02.00 | 13.03.32 | 75 | 40 | 00.01.31 |
| | 3 | 13.03.51 | 13.05.22 | 49 | 40 | 00.01.31 |
| | 4 | 13.05.34 | 13.07.55 | 90 | 25 | 00.02.20 |
| | 5 | 13.08.12 | 13.09.45 | 67 | 40 | 00.01.32 |
| | 6 | 13.10.00 | 13.11.44 | 70 | 30 | 00.01.43 |
| | 7 | 13.11.55 | 13.13.27 | 68 | 40 | 00.01.32 |
| | 8 | 13.13.33 | 13.15.37 | 85 | 20 | 00.02.03 |
| | 9 | 13.15.56 | 13.17.58 | 82 | 20 | 00.02.01 |
| | 10 | 13.18.07 | 13.19.33 | 58 | 45 | 00.01.25 |
| 16 - 17 | 1 | 16.17.22 | 16.19.30 | 83 | 35 | 00.02.08 |
| | 2 | 16.19.35 | 16.21.48 | 91 | 30 | 00.02.13 |
| | 3 | 16.21.54 | 16.24.19 | 106 | 30 | 00.02.25 |
| | 4 | 16.24.24 | 16.27.13 | 124 | 20 | 00.02.49 |
| | 5 | 16.27.25 | 16.29.14 | 73 | 40 | 00.01.49 |
| | 6 | 16.29.21 | 16.31.59 | 124 | 25 | 00.02.38 |
| | 7 | 16.32.06 | 16.34.17 | 109 | 25 | 00.02.11 |
| | 8 | 16.34.28 | 16.37.03 | 121 | 25 | 00.02.35 |
| | 9 | 16.37.10 | 16.39.12 | 93 | 35 | 00.02.02 |
| | 10 | 16.39.17 | 16.41.30 | 124 | 30 | 00.02.13 |

Data yang telah didapatkan dikelompokkan berdasarkan nama jalan pada rute yang telah ditentukan. Rute menuju ke stasiun Kotabaru ditentukan berdasarkan rute yang diambil dari Google *maps* yang terdiri dari dua rute. Data dalam satu jalan dikelompokkan berdasarkan hari diambilnya data tersebut. Data pada hari tersebut dikelompokkan lagi berdasarkan rentang waktu dari diambilnya data tersebut. Data ini digunakan sebagai data *training* untuk aplikasi ini.

#### 4.1.1.2 Halaman pengolahan data

Setelah data *Training* didapatkan, data tersebut kemudian di simpan ke dalam *database* MySQL, karena aplikasi ini dibangun menggunakan *database* tersebut. Halaman pengolahan data berguna memudahkan untuk proses *input* dan pengolahan data. Tampilan halaman *dashboard* pengolahan data dapat dilihat pada Gambar 4.6.




Gambar 4.6 Halaman *dashboard* pengolahan data.

Halaman pengolahan data memiliki beberapa menu yaitu data jalan dan tujuan, data *graph* dan data penelitian. Data jalan dan tujuan berisi nama-nama jalan yang dilewati oleh *driver* pada rute yang ditentukan dan data nama lokasi tujuan rute tersebut. Di dalam menu tersebut, terdapat tabel dari data nama jalan dan nama

tujuan disertai dengan tombol untuk edit dan hapus, juga terdapat tombol untuk memasukkan data baru. Pada menu data graph berisi tabel yang memuat beberapa kolom sebagai berikut ini.

a. Simpul awal

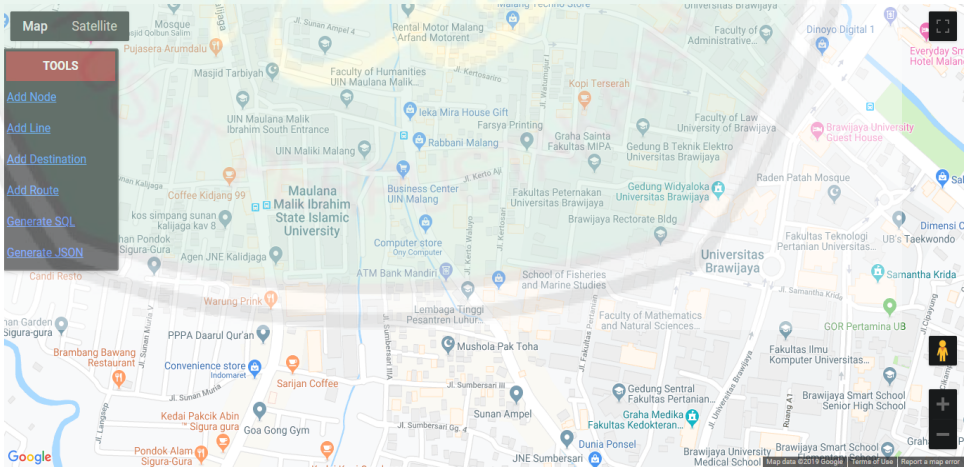
   Kolom ini memuat data nama jalan dimana *driver* memulai.

b. Simpul tujuan

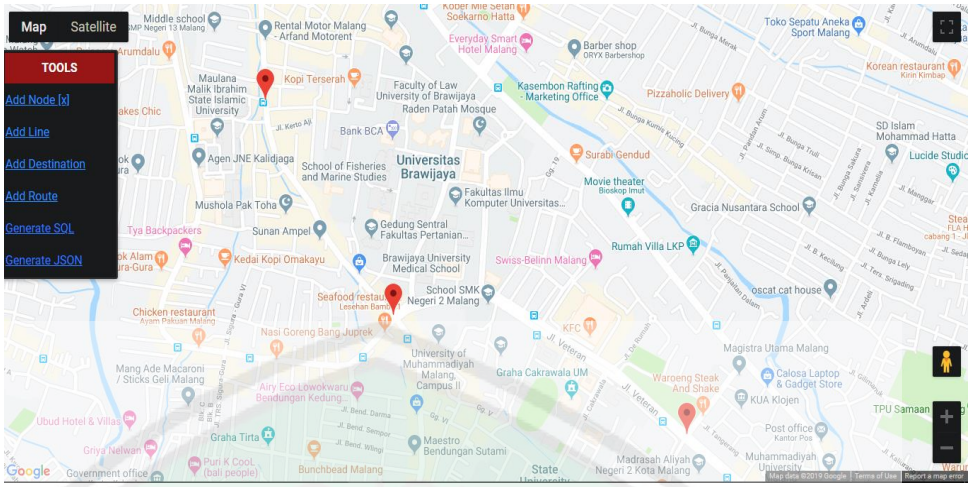
   Kolom ini memuat data nama jalan tujuan *driver*.

c. Jalur

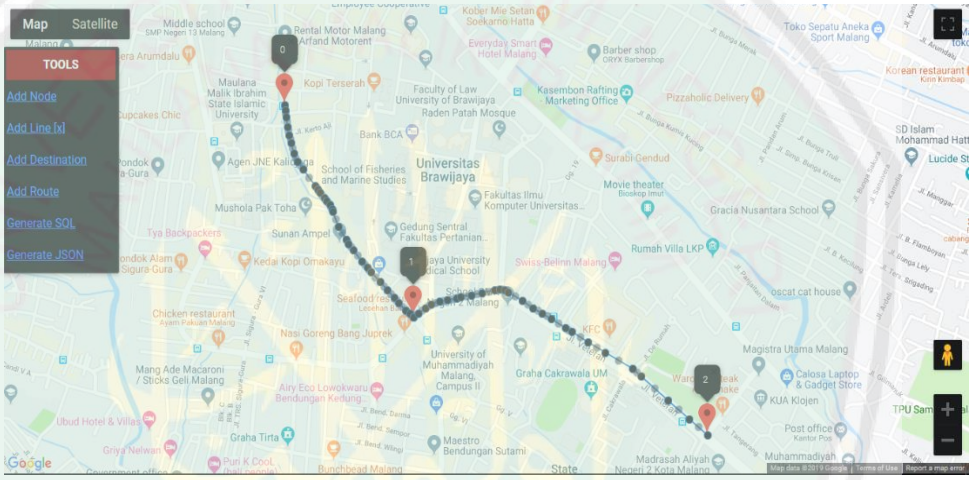
   Kolom ini berisi data koordinat dari jalur yang dilalui oleh *driver* dari simpul awal (*nodes* awal) sampai simpul tujuan (*nodes* tujuan). Koordinat tersebut didapatkan dari http://graph.latcoding.com/. Langkah-langkah untuk mendapatkan koordinat, yang pertama adalah menambahkan *node* yang dibutuhkan, kedua menambahkan *line* atau jalur antara pasangan *node*, dan langkah terakhir adalah melakukan *generate* json sehingga didapatkan seluruh koordinat dari setiap pasangan *node* yang dibutuhkan.



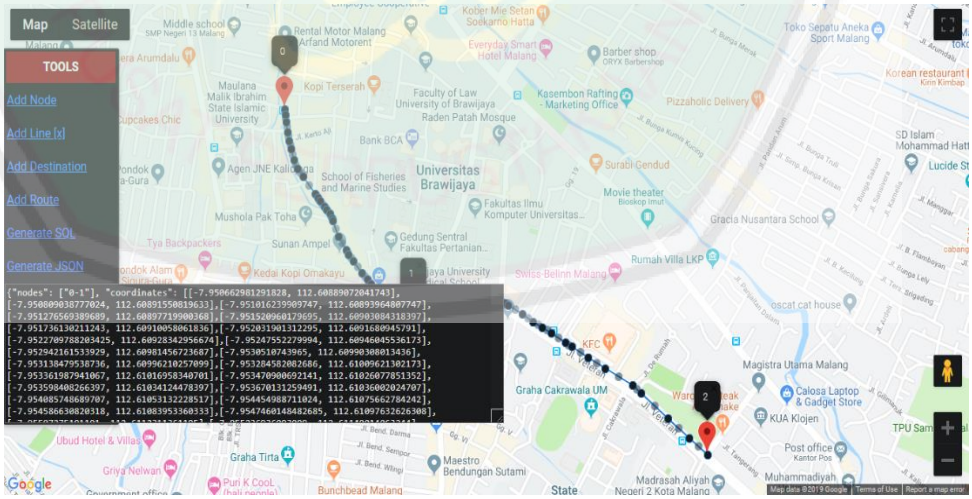

Gambar 4.7 Halaman awal graph.latcoding.com.




Gambar 4.8 Menambahkan *node*.




Gambar 4.9 Menambahkan *line*.




Gambar 4.10 *Generate* Json.

d. Jarak

Kolom ini berisi jarak yang harus dilalui *driver* untuk mencapai *nodes* tujuan dari *nodes* awal dalam satuan meter. Data jarak diambil dari Google Maps.

e. Bobot

Kolom ini berisi bobot yang dihasilkan berdasarkan perhitungan algoritma *multiple regression* berdasarkan parameter yang telah ditentukan. Kolom ini tidak bisa diisi dan diedit secara manual, karena langsung diisi oleh hasil perhitungan dari hasil perhitungan algoritma *multiple regression*.

Pada tabel data *graph* juga terdapat tombol untuk *edit* data dan hapus data. Juga terdapat tombol untuk menambahkan data baru.

Pada bagian menu data penelitian berisi hasil pengumpulan data yang telah dilakukan. Terdapat sebuah tabel yang terdiri dari beberapa kolom sebagai berikut ini.

- Simpul awal dan Simpul Tujuan

  Simpul awal merupakan nama jalan dimana *driver* mulai dan simpul tujuan merupakan nama jalan tujuan *driver*.

- Waktu

  Kolom ini berisi rentang waktu dilakukannya pengumpulan data.

- Start dan Finish

  *Start* merupakan waktu dimulainya *driver* pada simpul awal dan *finis* merupakan waktu ketika sampai tujuan pada simpul tujuan.

- Jumlah kendaraan

Kolom ini berisi jumlah kendaraan yang melintasi jalur tersebut mulai dari simpul awal sampai tujuan dengan batas waktu *start* sampai waktu *finish*.

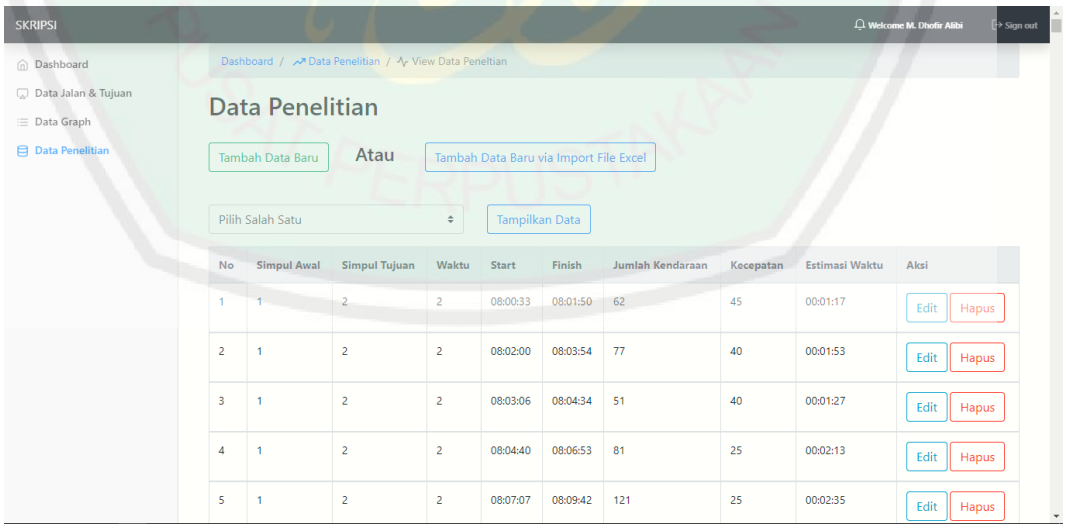
- Kecepatan

  Kecepatan rata-rata dari sepeda motor yang digunakan *driver*.

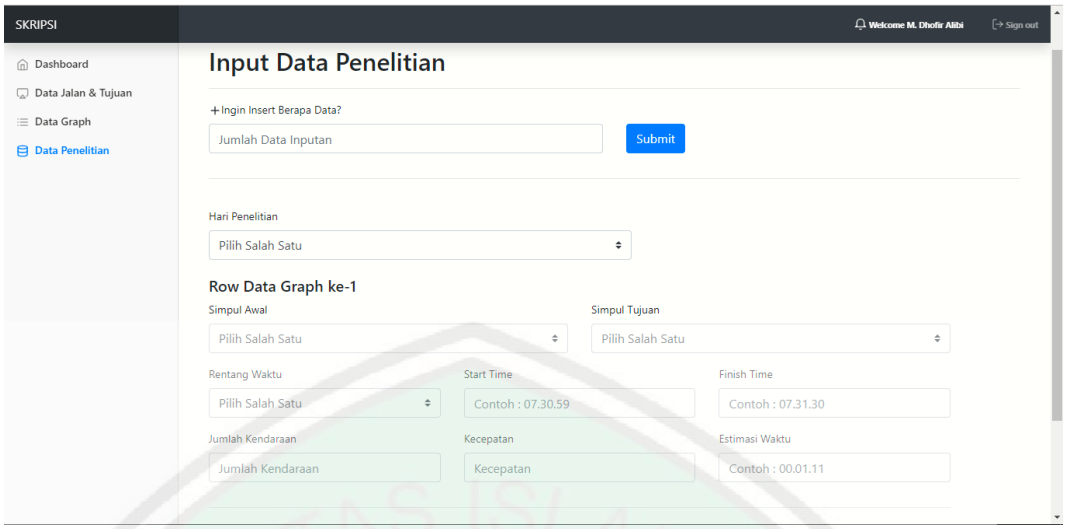
- Estimasi waktu

  Waktu yang dibutuhkan *driver* dalam menempuh perjalanan dari simpul awal sampai simpul tujuan.

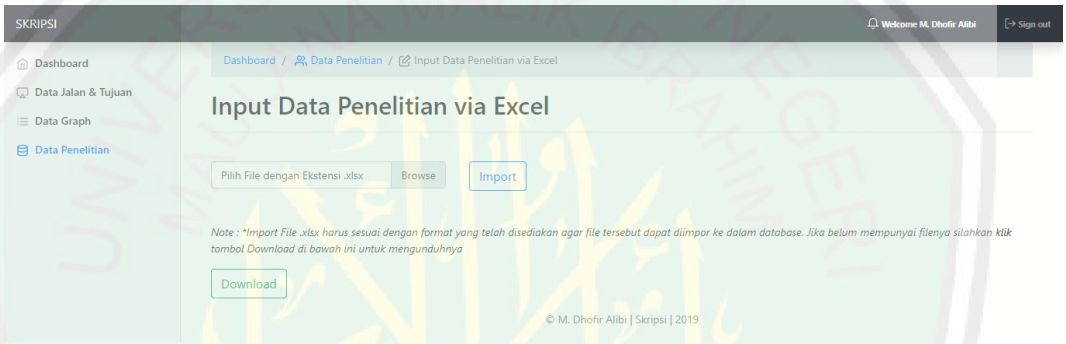
Tampilan olah data penelitian dapat dilihat pada Gambar 4.11. Di dalam tampilan tersebut sebuah *dropdown* yang berisi hari diambilnya data tersebut. Tabel di bawahnya berisi data yang ditampilkan berdasarkan hari yang dipilih. Untuk menambah data baru, pada menu data penelitian ini disediakan dua cara. Yang pertama adalah dengan cara menambahkan data melalui *form input* manual yang ditunjukkan pada Gambar 4.12 dan yang kedua adalah dengan cara impor data dari *file* excel yang ditunjukkan pada Gambar 4.13. Juga terdapat tombol untuk *edit* data dan hapus data.




Gambar 4.11 Halaman data penelitian.




Gambar 4.12 Halaman *input* data penelitian secara manual.




Gambar 4.13 Halaman *input* data penelitian dengan *import file* excel.

### 4.1.2 Penerapan Algoritma *Multiple Regression*

Setelah data *training* didapatkan dan disimpan di dalam *database*. Proses selanjutnya adalah mencari bobot dengan menggunakan algoritma *multiple regression*. *Input* data pertama kali adalah data kecepatan, hari dan waktu. Data kecepatan adalah 40 km/jam, data hari adalah hari Senin dan waktu adalah jam 08:01:30 WIB. Kemudian data *graph* diambil dari *database*. Titik awal adalah depan jalan Sumbersari, maka data *graph* yang diambil berdasarkan *node* awal adalah jalan Sumbersari dengan *node* tujuan adalah jalan Veteran.

Kemudian data *training* diambil berdasarkan data *graph* yakni *node* awal dan *node* tujuan, hari senin dan pada rentang waktu antara jam 08.01-11.00 WIB. Data *training* yang diambil dari *database* ditunjukkan pada Tabel 4.2 berikut.

Tabel 4.2 Data *training* yang diambil dari *database*.

| ID Data | Start | Finish | Jumlah Kendaraan | Kecepatan | Estimasi Waktu |
|---|---|---|---|---|---|
| 1 | 08:00:33 | 08:01:50 | 62 | 45 | 00:01:17 |
| 2 | 08:02:00 | 08:03:54 | 77 | 40 | 00:01:53 |
| 3 | 08:03:06 | 08:04:34 | 51 | 40 | 00:01:27 |
| 4 | 08:04:40 | 08:06:53 | 81 | 25 | 00:02:13 |
| 5 | 08:07:07 | 08:09:42 | 121 | 25 | 00:02:35 |
| 6 | 08:09:50 | 08:11:45 | 65 | 25 | 00:01:44 |
| 7 | 08:11:55 | 08:13:56 | 85 | 50 | 00:02:01 |
| 8 | 08:14:05 | 08:16:35 | 124 | 30 | 00:02:30 |
| 9 | 08:16:42 | 08:18:44 | 70 | 35 | 00:02:01 |
| 10 | 08:18:57 | 08:20:18 | 62 | 45 | 00:01:17 |

Proses selanjutnya adalah menentukan data *testing* berdasarkan waktu terdekat dari *input* data waktu. Dalam hal ini waktu yang terdekat adalah pada data yang pertama dengan jumlah kendaraan 62. Karena *input* data waktu adalah 08:01:30 WIB maka *range* data start dan finish yang mendekati adalah data yang pertama. Setelah data *testing* ditentukan, proses selanjutnya adalah pemodelan regresi linear berganda. Langkah pertama adalah deklarasi data. *Source code* yang digunakan untuk deklarasi data ditunjukkan pada Gambar 4.14 dan Hasilnya ditunjukkan pada Tabel 4.3.

```
function getEstimasiWaktu($training, $testing, $kecepatan){
$x = array();
$y = array();
// Langkah 1 : mendeklarasikan data yg dibutuhkan
foreach ($training as $key => $value) {
array_push($x, array(1, $value["jumlah_kendaraan"], $value["kecepatan"]
));
array_push($y, timeToDecimal($value["estimasi_waktu"]));
}
```

Gambar 4.14 *Source code* deklarasi data

Tabel 4.3 Deklarasi data.

| | X | | Y |
|---|---|---|---|
| 1 | 77 | 40 | 1,8833333 |
| 1 | 51 | 40 | 1,45 |

Tabel 4.3 Deklarasi data (lanjutan).

| | | | |
|---|---|---|---|
| 1 | 81 | 25 | 2,2166667 |
| 1 | 121 | 25 | 2,5833333 |
| 1 | 65 | 25 | 1,7333333 |
| 1 | 85 | 50 | 2,0166667 |
| 1 | 124 | 30 | 2,5 |
| 1 | 70 | 35 | 2,0166667 |
| 1 | 62 | 45 | 1,2833333 |

Langkah kedua adalah mengubah variabel X menjadi X transpose (X'). *Source code* yang digunakan dapat dilihat pada Gambar 4.15 dan hasilnya ditunjukkan pada Tabel 4.4.

```
$x_transpose = array();
foreach ($x as $key_baris => $value_baris) {
  foreach ($value_baris as $key_kolom => $value_kolom) {
    $x_transpose[$key_kolom][$key_baris] = $value_kolom;
  }
}
```

Gambar 4.15 *Source code* untuk mengubah X menjadi X' (X *transpose*).

Tabel 4.4 Hasil pengubahan X menjadi X' (*transpose*).

| **X'** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 77 | 51 | 81 | 121 | 65 | 85 | 124 | 70 | 62 |
| 40 | 40 | 25 | 25 | 25 | 50 | 30 | 35 | 45 |

Kemudian pada langkah ketiga adalah menghitung perkalian antara X transpose (X') dan X. *Source code* yang digunakan dapat dilihat pada Gambar 4.16 dan hasilnya ditunjukkan pada Tabel 4.5.

```
$jml_baris_kolom = count($x_transpose);
$perkalian = array();
for ($i=0; $i < $jml_baris_kolom; $i++) {
  for ($j=0; $j < $jml_baris_kolom; $j++) {
    $total = 0;
    for ($k=0; $k < count($x_transpose[$j]); $k++) {
       $total += $x_transpose[$i][$k] * $x[$k][$j];
    }
    $perkalian[$i][$j] = $total;
  }
}
```

Gambar 4.16 *Source code* perkalian antara X'(*transpose*) dengan X.

Tabel 4.5 Hasil perkalian X'(*transpose*) dengan X.

| **X'.X** | | |
|---|---|---|
| 9 | 736 | 315 |
| 736 | 65302 | 25005 |
| 315 | 25005 | 11725 |

Pada langkah keempat dilakukan perhitungan *invers* dari perkalian antara variabel X dengan X' (*transpose*). *Source code* dari perhitungan tersebut dapat dilihat pada Gambar 4.17 dan hasilnya dapat dilihat pada Tabel 4.6.

```
$invers_langkah4 = array();
$dm1 = $perkalian[0][0] * $perkalian[1][1] * $perkalian[2][2];
$dm2 = $perkalian[0][1] * $perkalian[1][2] * $perkalian[2][0];
$dm3 = $perkalian[0][2] * $perkalian[1][0] * $perkalian[2][1];
$dm4 = $perkalian[0][1] * $perkalian[1][0] * $perkalian[2][2];
$dm5 = $perkalian[0][0] * $perkalian[1][2] * $perkalian[2][1];
$dm6 = $perkalian[0][2] * $perkalian[1][1] * $perkalian[2][0];

$determinan = ($dm1 + $dm2 + $dm3) - ($dm4 + $dm5 + $dm6);
if ($determinan == 0) {
  echo "Matriks Tidak memiliki invers";
} else {
  // menentukan kofaktor matriks ordo 3x3
$k11 = ($perkalian[1][1] * $perkalian[2][2]) - ($perkalian[2][1] *
$perkalian[1][2]);
$k12 = ($perkalian[1][0] * $perkalian[2][2]) - ($perkalian[2][0] *
$perkalian[1][2]);
$k13 = ($perkalian[1][0] * $perkalian[2][1]) - ($perkalian[2][0] *
$perkalian[1][1]);
$k21 = ($perkalian[0][1] * $perkalian[2][2]) - ($perkalian[2][1] *
$perkalian[0][2]);
$k22 = ($perkalian[0][0] * $perkalian[2][2]) - ($perkalian[2][0] *
$perkalian[0][2]);
$k23 = ($perkalian[0][0] * $perkalian[2][1]) - ($perkalian[2][0] *
$perkalian[0][1]);
$k31 = ($perkalian[0][1] * $perkalian[1][2]) - ($perkalian[1][1] *
$perkalian[0][2]);
$k32 = ($perkalian[0][0] * $perkalian[1][2]) - ($perkalian[1][0] *
$perkalian[0][2]);
$k33 = ($perkalian[0][0] * $perkalian[1][1]) - ($perkalian[1][0] *
$perkalian[0][1]);
$adj11 = $k11 * 1;
$adj12 = $k21 * -1;
$adj13 = $k31 * 1;
$adj21 = $k12 * -1;
$adj22 = $k22 * 1;
$adj23 = $k32 * -1;
$adj31 = $k13 * 1;
$adj32 = $k23 * -1;
$adj33 = $k33 * 1;
$invers_langkah4 = array(array(($adj11 / $determinan), ($adj12 /
$determinan), ($adj13 / $determinan)), array(($adj21 / $determinan),
($adj22 / $determinan), ($adj23 / $determinan)), array(($adj31 /
$determinan), ($adj32 / $determinan), ($adj33 / $determinan)));
}
```

```
$sigma_x1_kali_y = array();
$sigma_x2_kali_y = array();

foreach ($x as $key => $value) {
 $x1_kali_y = $value[1] * $y[$key];
 $x2_kali_y = $value[2] * $y[$key];
 array_push($sigma_x1_kali_y, $x1_kali_y);
 array_push($sigma_x2_kali_y, $x2_kali_y);
}
$sigma_y = array_sum($y);
$sigma_x1_kali_y = array_sum($sigma_x1_kali_y);
$sigma_x2_kali_y = array_sum($sigma_x2_kali_y);

$x_aksen_y = array($sigma_y, $sigma_x1_kali_y, $sigma_x2_kali_y);
```

Gambar 4.17 *Source code* perhitungan *invers* perkalian X dengan X'(*transpose*).

Tabel 4.6 Hasil perhitungan *invers* perkalian X dengan X'(*tranpose*).

| $(X'.X)^{-1}$ | | |
|---|---|---|
| 5,184235 | -0,0278 | -0,0799866 |
| -0,0278 | 0,000233 | 0,0002509 |
| -0,07999 | 0,000251 | 0,0016992 |

Selanjutnya pada langkah kelima adalah substitusi hasil perhitungan *invers* ke dalam rumus $(X'.X)^{-1}(X'.Y)$. *Source code* dari perhitungan tersebut ditunjukkan pada Gambar 4.18 . Hasil perhitungannya ditunjukkan pada Tabel 4.7.

```
$subtitusi_langkah5 = array();
$jml_baris_kolom = count($invers_langkah4);
for ($i=0; $i < $jml_baris_kolom; $i++){
  for($j=0; $j < $jml_baris_kolomn; $i++){
    $total = 0;
    for ($k=0; $k < count($invers_langkah4[$j]); $k++) {
      $total += $invers_langkah4[$i][$k] * $x_aksen_y[$k];
    }
    $subtitusi_langkah5[$i] = $total;
  }
}
```

Gambar 4.18 *Source code* substitusi hasil perhitungan *invers*.

Tabel 4.7 Hasil substitusi perhitungan *invers*.

| $(X'.X)^{-1}$ | | | X'Y |
|---|---|---|---|
| 5,184235 | -0,0278 | -0,0799866 | 17,683333 |
| -0,0278 | 0,000233 | 0,0002509 | 1525,9167 |
| -0,07999 | 0,000251 | 0,0016992 | 600,83333 |

Pada langkah terakhir adalah menghitung estimasi parameter dengan melakukan perkalian matriks antara $(X'.X)^{-1}$ dan X'Y. Hasilnya ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil estimasi parameter regresi.

| Hasil | | |
|---|---|---|
| **a** | 1,1922697 | **Intercept** |
| **b1** | 0,0140278 | **Koefisien 1** |
| **b2** | -0,0107033 | **Koefisien 2** |

Kemudian setelah diketahui estimasi parameternya, tinggal memasukkan data yang digunakan sebagai *testing*. Data *testing* jumlah kendaraan adalah 62 dan data kecepatan yang merupakan *input* dari *user* adalah 40 km/jam. Perhitungan bobot menggunakan regresi linear berganda dengan rumus :

$$Y = a + b1x1 + b2x2 \quad (4.1)$$

*Source code* perhitungan bobot ditunjukkan pada Gambar 4.19. Hasil dari perhitungan bobot ditunjukkan pada Tabel 4.9 berikut.

```
$hasil = $subtitusi_langkah5[0] + ($subtitusi_langkah5[1] *
$testing["jumlah_kendaraan"]) + ($subtitusi_langkah5[2] * $kecepatan);
return $hasil;
}
```

Gambar 4.19 *Source code* perhitungan bobot.

Tabel 4.9 Perhitungan bobot menggunakan regresi linear berganda.

| a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|
| 1,1922697 | 0,0140278 | -0,0107033 | 62 | 40 |
| **Desimal** | | | 1,633859359 | |
| **Jam** | | | 00:01:38 | |

Hasil yang didapatkan dari perhitungan bobot yang telah dilakukan adalah 00:01:38 pada *node* awal yakni jalan Sumbersari menuju ke *node* tujuan yakni jalan Veteran. Hasil estimasi bobot ini nantinya akan menjadi masukan penentuan rute tercepat menggunakan algoritma *dijkstra*.

### 4.1.3 Penerapan Algoritma *Dijkstra*

Pada perhitungan bobot menggunakan algoritma *multiple regression* diperoleh hasil bobot yang digunakan sebagai masukan penentuan rute tercepat menggunakan algoritma *dijkstra*. Perhitungan bobot tidak hanya dilakukan pada *node* awal yakni jalan Sumbersari ke *node* tujuan yakni jalan Veteran, namun juga pada semua *node* dengan rute yang berbeda. Setelah semuanya dihitung, kemudian dibandingkan antara estimasi bobot pada rute pertama dan rute kedua. Kemudian dicari estimasi bobot yang paling terkecil antara keduanya. *Source code* algoritma *dijkstra* dapat dilihat pada Gambar 4.20.

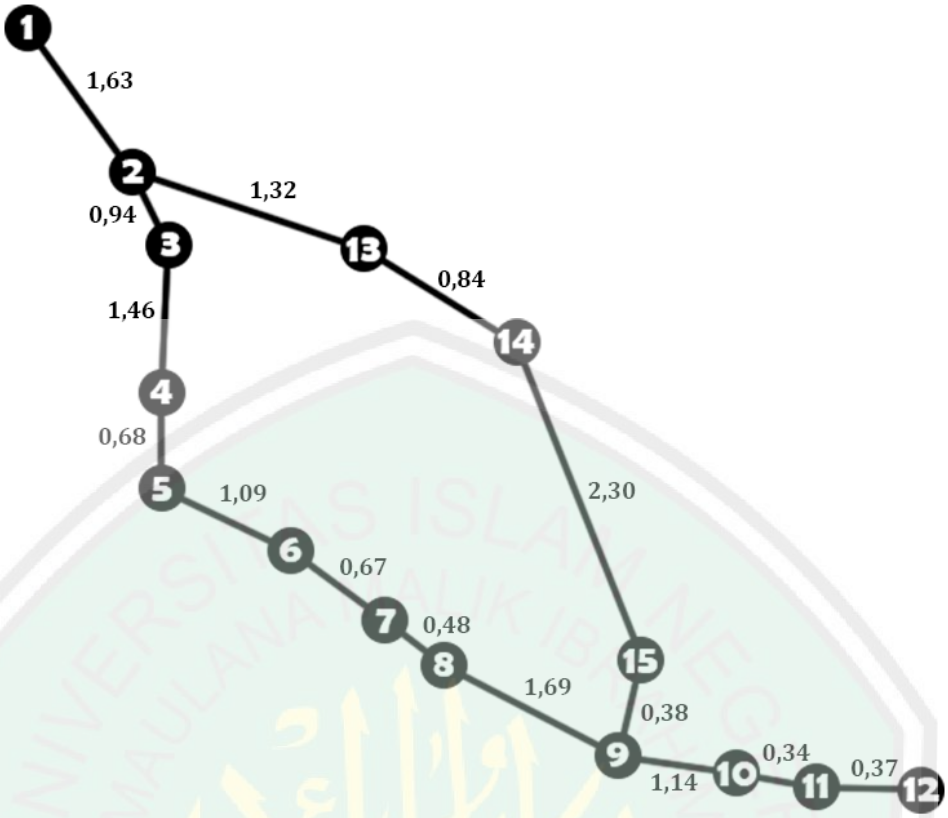
```
function getDijkstra($awal, $tujuan, $graph) {
 $node_keberangkatan = $awal;
 (float)$bobot = 0;
 (float)$jarak = 0;
 $node_telah_dilalui = array();
 $node_terkecil = array();

 while ($node_keberangkatan != $tujuan) {
  $temp = $graph[$node_keberangkatan];
  $node_hitung = array();

  if (!empty($node_telah_dilalui)) {
   foreach ($temp as $key => $value) {
     $jmlJarak = $value[2] + $jarak;
     $jmlBobot = $value[3] + $bobot;
     $cari = array_search($value[1], array_column($node_telah_dilalui,
     1));

     // cek apakah node sudah dilalui
     if (is_int($cari)) { // jika true
      //cek apakah bobot node saat ini lebih kecil dari bobot node
       sudah dilalui
       if ($jmlBobot < $node_telah_dilalui[$cari][3]) { // jika true
         // input data saat ini ke node hitung
         array_push($node_hitung,      array($value[0],      $value[1],
         $jmlJarak, $jmlBobot));
         // hapus data berdasarkan index cari pada node telah dilalui
         unset($node_telah_dilalui[$cari]);
       } else { // jika false
         // pindah data dari node telah dilalui ke node hitung
         array_push($node_hitung, $node_telah_dilalui[$cari]);
         // hapus data berdasarkan index cari pada node telah dilalui
         unset($node_telah_dilalui[$cari]);
       }
     } else { // jika else maka input data ke node hitung
       array_push($node_hitung, array($value[0], $value[1], $jmlJarak,
      $jmlBobot));
     }
  }
```

```
    // cek apakah masih ada data di node telah dilalui
     if (!empty($node_telah_dilalui)) {
     // pindahkan data node telah dilalui ke node hitung
      foreach ($node_telah_dilalui as $key => $value) {
        array_push($node_hitung, array($value[0], $value[1], $value[2],
        $value[3]));
      }
      // reset node dilalui
      $node_telah_dilalui = array();
     }
    } else {
      foreach ($temp as $key => $value) {
       $jmlJarak = $value[2] + $jarak;
       $jmlBobot = $value[3] + $bobot;
       array_push($node_hitung, array($value[0], $value[1], $jmlJarak,
        $jmlBobot));
      }
    }
    // cari nilai terkecil pada kolom bobot
    $min = min(array_column($node_hitung, 3));
    // cari index nilai terkecil
    $index = array_search($min, array_column($node_hitung, 3));
    // ubah node keberangkatan ke node terpilih
    $node_keberangkatan = $node_hitung[$index][1];
    // set jarak dari node terpilih
    $jarak = $node_hitung[$index][2];
    // set bobot dari node terpilih
    $bobot = $node_hitung[$index][3];

    // pindah node dg bobot terkecil ke node terkecil
    array_push($node_terkecil, $node_hitung[$index]);
    // hapus data yang telah dipindah
    unset($node_hitung[$index]);

    // pindah data pada node hitung ke node telah dilalui
    if (!empty($node_hitung)) {
     $node_telah_dilalui = $node_hitung;
    }
}
echo "=========================================================<br/>";
echo "Node Terkecil";
echo "<pre>";
print_r($node_terkecil);
echo "</pre>";
```

```
$jarakTotal = 0;
$waktuTotal = 0;
$rute = array();

$nodeSebelumnya = 0;
for ($i=(count($node_terkecil) - 1); $i >= 0; $i--) {
 if ($i == (count($node_terkecil) - 1)) {
   $jarakTotal = $node_terkecil[$i][2];
   $waktuTotal = $node_terkecil[$i][3];
   array_unshift($rute, $node_terkecil[$i][1]);
   // menambah elemen pertama array
   $nodeSebelumnya = $node_terkecil[$i][0];
 }elseif ($i == 0) {
   array_unshift($rute, $node_terkecil[$i][0], $node_terkecil[$i][1]);
 }else {
   if ($node_terkecil[$i][1] == $nodeSebelumnya) {
     array_unshift($rute, $node_terkecil[$i][1]);
     $nodeSebelumnya = $node_terkecil[$i][0];
   }
 }
}
$hasil = array($rute, $jarakTotal, $waktuTotal);
return $hasil;
}
```

Gambar 4.20 *Source code* penentuan rute menggunakan algoritam *dijkstra*.

Hasil dari perbandingan estimasi bobot antara rute pertama dan rute kedua ditunjukkan dengan ilustrasi *graph* dari kedua rute tersebut dengan masing-masing bobot dari setiap jalan. *Graph* kedua rute tersebut dapat dilihat pada Gambar 4.21. Kemudian dari *graph* tersebut, diperoleh rute tercepat dari lokasi *start* yaitu di depan kampus Universitas Islam Negeri Maulana Ibrahim Malang sampai lokasi finish yaitu Stasiun Kotabaru dan waktu tempuh yang dibutuhkan. Hasil tersebut dapat dilihat pada Gambar 4.22.

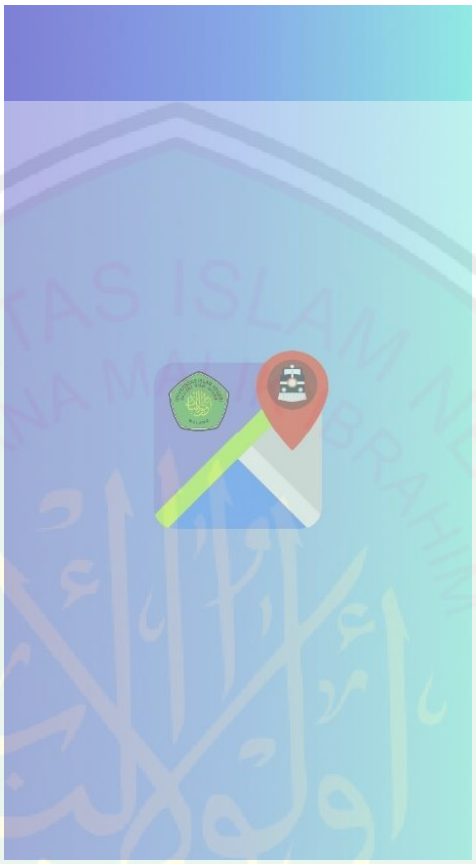



Gambar 4.21 *Graph* perbandingan estimasi bobot kedua rute.


Gambar 4.22 Hasil rute tercepat dan waktu tempuh.

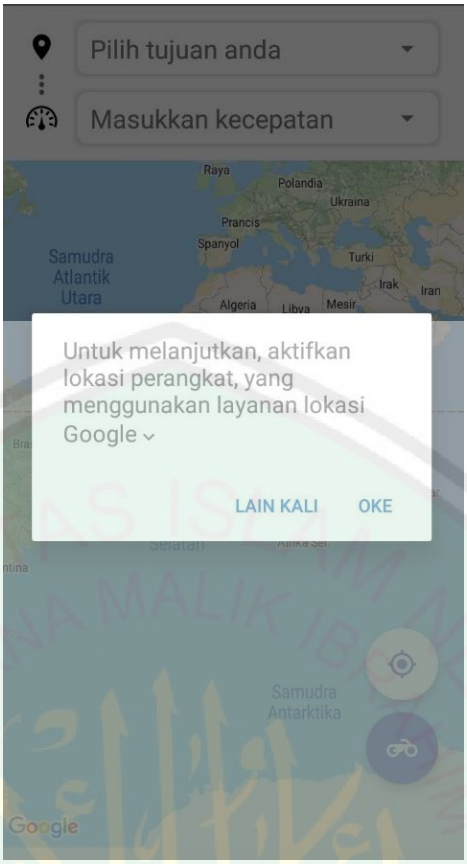
### 4.1.4 Implementasi Desain *Interface*

Ketika pertama kali membuka aplikasi akan muncul halaman *splash screen* seperti ditunjukkan pada Gambar 4.23.



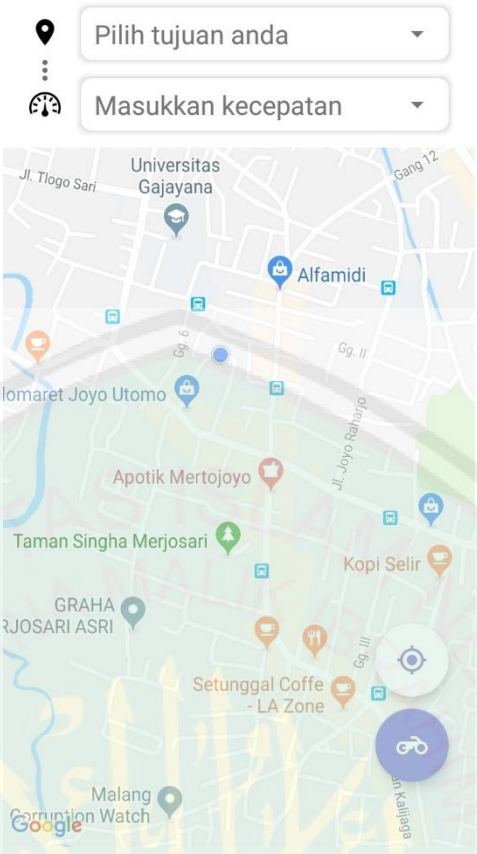
Gambar 4.23 *Splash screen* aplikasi

Kemudian akan masuk ke *interface* utama aplikasi. Karena aplikasi harus mendapatkan peta lokasi dari pengguna, jika pengguna belum mengaktifkan *Service* lokasinya, maka akan muncul sebuah pop-up untuk mengaktifkan *Service* lokasi dari *device* pengguna seperti ditunjukkan pada Gambar 4.24. Ketika pengguna menekan tombol "OK", otomatis akan masuk ke *interface* utama aplikasi. Gambar 4.25 merupakan *interface* utama aplikasi.




Gambar 4.24 *Pop-up* untuk mengaktifkan *service* lokasi.

Pada *interface* utama aplikasi terdapat *dropdown* untuk memilih tujuan dan *dropdown* untuk memilih kecepatan berkendara yang diinginkan. Pengguna tinggal memilih lokasi tujuan dan memilih kecepatan berkendara yang diinginkan pengguna. Rentang kecepatan rata-rata dimulai dari 30-60. Setelah itu pengguna tinggal menekan tombol dengan ikon sepeda motor untuk mengetahui rute tercepat dan waktu tempuh yang dibutuhkan. Kemudian akan muncul sebuah rute dengan waktu tempuh yang dibutuhkan seperti pada Gambar 4.22.




Gambar 4.25 *Interface* utama aplikasi.

## 4.2 Hasil Uji Coba

Berdasarkan implementasi sistem yang telah dijabarkan pada sub bab sebelumnya, proses pengujian dilakukan secara langsung dengan *testing* berkendara menggunakan sepeda motor dari lokasi awal menuju lokasi akhir. Sebelum itu, ditentukan terlebih dahulu *input* data yang dibutuhkan untuk percobaan. Tabel 4.10 merupakan daftar *input* yang digunakan untuk lima percobaan aplikasi.

Tabel 4.10 Daftar *input* yang digunakan.

| Percobaan Ke- | Waktu | Hari | Tanggal | Kecepatan (km/jam) |
|---|---|---|---|---|
| 1 | 14:49:41 | Rabu | 22/05/2019 | 40 |
| 2 | 15:14:59 | Rabu | 22/05/2019 | 35 |
| 3 | 14:06:48 | Kamis | 23/05/2019 | 40 |
| 4 | 14:33:06 | Kamis | 23/05/2019 | 35 |
| 5 | 14:58:03 | Kamis | 23/05/2019 | 35 |

Tabel 4.10 Daftar *input* yang digunakan (lanjutan).

| | | | | |
|---|---|---|---|---|
| 6 | 15:25:51 | Kamis | 23/05/2019 | 30 |
| 7 | 14:10:19 | Kamis | 23/05/2019 | 40 |
| 8 | 14:36:01 | Kamis | 23/05/2019 | 45 |
| 9 | 15:01:34 | Kamis | 23/05/2019 | 40 |
| 10 | 15:27:52 | Kamis | 23/05/2019 | 35 |

Keterangan :

- Waktu : waktu dimulainya *testing* rute dari titik awal secara langsung.
- Hari : hari dilakukannya *testing*.
- Tanggal : tanggal pelaksanaan *testing*.
- Kecepatan : kecepatan rata-rata dari sepeda motor.

Proses pengujian dilakukan dengan membandingkan data waktu tempuh yang dihasilkan aplikasi dengan data *testing* rute secara langsung. Rute yang digunakan adalah sama dengan rute tercepat yang dihasilkan aplikasi. Kemudian dihitung persentase *error*-nya. Sehingga akurasi dapat dihitung berdasarkan persentase *error* yang dihasilkan. Perhitungan *error* atau galat menggunakan rumus oleh (Capra, 2012). Rumus perhitungan tersebut adalah sebagai berikut:

$$\%\boldsymbol{error} = \boldsymbol{abs}\left(\frac{\boldsymbol{Exact\ value} - \boldsymbol{Approximate\ value}}{\boldsymbol{Exact\ value}}\right) \times \mathbf{100} \quad (4.2)$$

*Exact value* merupakan nilai sebenarnya dan *approximate value* merupakan nilai dugaan atau prediksi. Pada penelitian ini yang menjadi acuan nilai sebenarnya adalah waktu tempuh yang dihasilkan dari *testing* rute secara langsung. Sedangkan nilai dugaan adalah waktu tempuh yang dihasilkan oleh aplikasi.

Aplikasi yang telah dibuat ini bertujuan untuk mencari rute tercepat disertai dengan estimasi waktu yang dibutuhkan dari titik awal menuju ke titik tujuan. Untuk mengetahui tingkat keberhasilan dari sistem ini, dibutuhkan sebuah aplikasi dengan fungsi yang sama sebagai perbandingannya. Aplikasi yang dijadikan

perbandingan adalah Google *maps*. Maka dari itu, dihitung juga persentase *error* dari Google *maps*.

Proses perhitungannya sama dengan perhitungan *error* yang dilakukan pada aplikasi yang telah dibangun yaitu dengan menggunakan persamaan 4.2 diatas. Nilai acuan sebenarnya (*exact value*) adalah waktu tempuh yang dihasilkan dari *testing* rute secara langsung. Dan nilai dugaan atau prediksi (*approximate value*) yang digunakan adalah waktu tempuh estimasi waktu yang diberikan Google *maps*.

Pengambilan *approximate value* dari aplikasi yang telah dibuat dan *approximate value* dari Google *maps* dilakukan pada waktu yang bersamaan dengan waktu mulai dari *testing* rute secara langsung. Sehingga, *approximate value* dari aplikasi dan dari Google *maps* dapat dibandingkan. Hal ini dikarenakan waktu pengambilan nilai tersebut dilakukan pada waktu yang bersamaan.

Setelah dilakukan *testing* aplikasi secara langsung berdasarkan *input* yang ditentukan, hasil dari percobaan tersebut dapat dilihat pada Tabel 4.11

Tabel 4.11 Hasil percobaan

| Percobaan Ke- | *Exact value* | *Approximate value* Aplikasi | *Approsimate value* Google *maps* |
|---|---|---|---|
| 1 | 00:08:49 | 00:08:14 | 00:13:00 |
| 2 | 00:09:42 | 00:08:59 | 00:15:00 |
| 3 | 00:09:52 | 00:08:08 | 00:12:00 |
| 4 | 00:09:05 | 00:08:45 | 00:13:00 |
| 5 | 00:09:09 | 00:08:45 | 00:13:00 |
| 6 | 00:09:50 | 00:09:19 | 00:14:00 |
| 7 | 00:10:15 | 00:08:06 | 00:13:00 |
| 8 | 00:09:18 | 00:07:35 | 00:13:00 |
| 9 | 00:09:20 | 00:08:10 | 00:13:00 |
| 10 | 00:10:20 | 00:08:45 | 00:13:00 |

Keterangan :

- *Exact value* adalah waktu tempuh hasil testing rute secara langsung.

- *Approximate value* aplikasi adalah waktu tempuh yang dihasilkan dari aplikasi.
- *Approximate value* Google *maps* adalah waktu tempuh yang diberikan oleh Google maps.

Kemudian untuk menentukan *error* dari percobaan tersebut, data waktu tempuh pada *exact value* dan *approximate value* di konversi menjadi desimal agar memudahkan perhitungan *error* nya. Data desimal dari setiap *value* dapat dilihat pada Tabel 4.12.

Tabel 4.12 Konversi desimal hasil percobaan

| Percobaan Ke- | *Exact value* | *Approximate value* Aplikasi | *Approximate value* Google *maps* |
|---|---|---|---|
| 1 | 8,82 | 8,23 | 13 |
| 2 | 9,70 | 8,98 | 15 |
| 3 | 9,87 | 8,13 | 12 |
| 4 | 9,08 | 8,75 | 13 |
| 5 | 9,15 | 8,75 | 13 |
| 6 | 9,83 | 9,32 | 14 |
| 7 | 10,25 | 8,10 | 13 |
| 8 | 9,30 | 7,58 | 13 |
| 9 | 9,33 | 8,17 | 13 |
| 10 | 10,33 | 8,75 | 13 |

Setelah di konversi menjadi desimal, kemudian dihitung persentase *error* nya menggunakan rumus yang telah dijelaskan pada halaman sebelumnya. Hasilnya adalah sebagai berikut ini :

Percobaan 1 : - Aplikasi : $\boldsymbol{abs}(\frac{\mathbf{8{,}82 - 8{,}23}}{\mathbf{8{,}82}})$ x 100 = 6,62%

- Google *maps* : $\boldsymbol{abs}(\frac{\mathbf{8{,}82 - 13}}{\mathbf{8{,}82}})$ x 100 = 47,45%

Percobaan 2 : - Aplikasi : $\boldsymbol{abs}(\frac{\mathbf{9{,}70 - 8{,}98}}{\mathbf{9{,}70}})$ x 100 = 7,39%

- Google *maps* : $\boldsymbol{abs}(\frac{\mathbf{9{,}70 - 15}}{\mathbf{9{,}70}})$ x 100 = 54,64%

Percobaan 3 : - Aplikasi : $abs(\frac{9{,}87 - 8{,}13}{9{,}87})$ x 100 = 17,57%

- Google *maps* : $abs(\frac{9{,}87 - 12}{9{,}87})$ x 100 = 21,62%

Percobaan 4 : - Aplikasi : $abs(\frac{9{,}08 - 8{,}75}{9.08})$ x 100 = 3,67%

- Google *maps* : $abs(\frac{9{,}08 - 13}{9.08})$ x 100 = 43,12%

Percobaan 5 : - Aplikasi : $abs(\frac{9{,}15 - 8{,}75}{9{,}15})$ x 100 = 4,37%

- Google *maps* : $abs(\frac{9{,}15 - 13}{9{,}15})$ x 100 = 42,08%

Percobaan 6 : - Aplikasi : $abs(\frac{9{,}83 - 9{,}32}{9{,}83})$ x 100 = 5,25%

- Google *maps* : $abs(\frac{9{,}83 - 14}{9{,}83})$ x 100 = 42,37%

Percobaan 7 : - Aplikasi : $abs(\frac{10{,}25 - 8{,}10}{10{,}25})$ x 100 = 20,98%

- Google *maps* : $abs(\frac{10{,}25 - 13}{10{,}25})$ x 100 = 26,83%

Percobaan 8 : - Aplikasi : $abs(\frac{9{,}30 - 7{,}58}{9{,}30})$ x 100 = 18,46%

- Google *maps* : $abs(\frac{9{,}30 - 13}{9{,}30})$ x 100 = 39,78%

Percobaan 9 : - Aplikasi : $abs(\frac{9{,}33 - 8{,}17}{9{,}33})$ x 100 = 12,50%

- Google *maps* : $abs(\frac{9{,}33 - 13}{9{,}33})$ x 100 = 39,29%

Percobaan 10 : - Aplikasi : $abs(\frac{10{,}33 - 8{,}75}{10{,}33})$ x 100 = 15,32%

- Google *maps* : $abs(\frac{10{,}33 - 13}{10{,}33})$ x 100 = 25,81%

### 4.3 Analisa Hasil dan Pembahasan

Berdasarkan hasil uji coba yang telah dilakukan, percobaan dari aplikasi yang memiliki persentase *error* terkecil adalah percobaan yang ke empat dengan nilai

3,67%. Dan percobaan ke tujuh merupakan percobaan aplikasi dengan nilai persentase *error* yang terbesar dengan nilai 20,98%. Pada google *maps*, percobaan yang memiliki nilai persentase *error* terkecil adalah percobaan ke 3 sebesar 21,62% dan persentase *error* terbesar terdapat pada percobaan ke 2 dengan nilai 54,64%. Tabel 4.13 berikut merupakan tabel persentase dari setiap percobaan.

Tabel 4.13 Nilai persentase *error* setiap percobaan.

| Percobaan Ke- | Persentase *error* Aplikasi | Persentase *error* Google *maps* |
|---|---|---|
| 1 | 6,62% | 47,45% |
| 2 | 7,39% | 54,64% |
| 3 | 17,57% | 21,62% |
| 4 | 3,67% | 43,12% |
| 5 | 4,37% | 42,08% |
| 6 | 5,25% | 42,37% |
| 7 | 20,98% | 26,83% |
| 8 | 18,46% | 39,78% |
| 9 | 12,50% | 39,29% |
| 10 | 15,32% | 25,81% |

Pada aplikasi, percobaan yang ke tujuh dengan *input* kecepatan sebesar 40 km/jam dengan waktu mulai 14:10:19 menghasilkan persentase *error* terbesar. Hal ini dikarenakan variasi data kecepatan pada data *training* tidak terlalu banyak sehingga menghasilkan persentase *error* terbesar. Sedangkan pada percobaan ke empat dengan *input* kecepatan sebesar 35 km/jam pada waktu 14:33:06 menghasilkan persentase *error* terkecil. Hal ini dikarenakan variasi data *training* yang lebih banyak.

Kemudian dari 10 percobaan yang telah dilakukan, rata-rata persentase *error* dari aplikasi yang telah dibuat adalah 11,21%. Sedangkan rata-rata persentase *error* pada Google *maps* adalah 38,30%. Nilai rata-rata tersebut diambil dari jumlah persentase *error* setiap percobaan tersebut dibagi dengan jumlah percobaan.

Nilai akurasi dari aplikasi dengan didasarkan pada nilai rata-rata *error* yang dihasilkan adalah sebesar 88,79 %. Sedangkan pada Google *maps* nilai akurasi yang dihasilkan adalah 61,70%. Tabel 4.14 berikut ini merupakan perbandingan nilai rata-rata *error* dan akurasi antara aplikasi dengan Google *maps*.

Tabel 4.14 Perbandingan nilai rata-rata *error* dan akurasi antara aplikasi dengan google *maps*.

| | Nilai rata-rata *error* | Nilai Akurasi |
|---|---|---|
| Aplikasi | 11,21% | 88,79% |
| Google *maps* | 38,30% | 61,70% |

Berdasarkan perhitungan persentase *error* dari aplikasi yang telah dibuat dan aplikasi yang telah ada yaitu Google *maps*, diketahui persentase *error* yang dihasilkan oleh aplikasi yang dibuat lebih kecil dari pada persentase *error* yang dihasilkan oleh Goole *maps*. Penyebab besarnya persentase *error* yang dihasilkan oleh Google *maps* dikarenakan bergantungnya Google *maps* pada fitur lokasi yang aktif dari setiap *device*. Dikutip dari kumparan.com, Google *maps* memprediksi kondisi lalu lintas dari kumpulan data yang dikoleksi dari fitur *Global Positioning System* (GPS) di setiap *smartphone* pengguna, baik iPhone ataupun Android. Ketika pengguna Android ataupun iPhone mengaktifkan fitur lokasi, mereka akan mengirimkan data perjalanannya ke sistem Google secara *real-time*. Dengan data tersebut Google akan mengkalkulasi kepadatan jalan tersebut dan memperkirakan kemungkinan rata-rata kecepatan yang melewati rute atau jalan tersebut. Oleh karena itu, semakin banyak pengguna *smartphone* yang menyalakan GPS, maka semakin akurat pula prediksi kemacetan yang dihasilkan oleh Google *maps*. Begitu juga sebaliknya, semakin sedikit pengguna *smartphone* yang menyalakan GPS, maka semakin tidak akurat pula prediksi kemacetan yang dihasilkan oleh Google

*maps* (Putri, 2018). Penyebab kecilnya persentase *error* yang dihasilkan oleh aplikasi adalah adanya *input* kecepatan yang dilakukan oleh pengguna. *Input* kecepatan tersebut akan mempengaruhi perhitungan estimasi waktu yang akan dihasilkan oleh aplikasi. Dan juga variasi data dari jumlah kendaraan (data *training*) juga mempengaruhi estimasi waktu yang dihasilkan oleh aplikasi

### 4.4 Integrasi dengan Alquran dan Hadits

Manusia merupakan makhluk individual sekaligus makhluk sosial. Disebut sebagai makhluk sosial dikarenakan manusia merupakan makhluk yang tidak dapat hidup sendirian. Manusia pasti membutuhkan yang lain, baik yang berkenaan dengan manusia maupun yang berkenaan dengan alam sekitarnya. Karena itu, pastinya tolong-menolong kepada yang saling membutuhkan termasuk perbuatan yang ditekankan. Kata tolong-menolong saja banyak dijumpai dalam ayat-ayat Alquran. Terdapat sekitar tujuh ayat Alquran yang mengandung kata tolong-menolong yaitu pada surat Ash-Shaffat ayat 25, Al-Fath ayat 3, Ar-Rum ayat 5, Al-Hasyr ayat 12, dan Al-Maidah ayat 2 dan 80. Tujuh ayat tersebut didasarkan dari hasil pencarian menggunakan aplikasi hasil dari penelitian mengenai pencarian ayat Alquran dalam terjemah berbahasa Indonesia (Dzulfikri, 2019). Juga dalam hadis disampaikan mengenai tolong-menolong. Seperti pada hadis berikut ini.

عَنْ أَبِيْ هُرَيْرَةَ رَضِيَ اللَّهُ عَنْهُ عَنِ النَّبِيِّ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ قَالَ مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا ، نَفَّسَ اللهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ، وَمَنْ يَسَّرَ عَلَى مُعْسِرٍ ، يَسَّرَ اللهُ عَلَيْهِ فِي الدُّنْيَا وَالْآخِرَةِ ، وَمَنْ سَتَرَ مُسْلِمًا ، سَتَرَهُ اللهُ فِي الدُّنْيَا وَالْآخِرَةِ ، وَاللهُ فِي عَوْنِ الْعَبْدِ مَا كَانَ الْعَبْدُ فِي عَوْنِ أَخِيهِ

Artinya : "*Dari Abu Hurairah radliyallahu 'anhu, Nabi shallalahu 'alaihi wa salam bersabda 'Siapa yang melapangkan satu kesusahan dunia dari seorang Mukmin, maka Allah melapangkan darinya satu kesusahan di hari Kiamat. Siapa memudahkan (urusan) orang yang kesulitan, maka Allah memudahkan baginya*

*(dari kesulitan) di dunia dan akhirat. Siapa menutupi(aib0 seorang Muslim, maka Allah akan menutupi (aib)nya di dunia dan akhirat. Allah senantiasa menolong seorang hamba selama hamba tersebut menolong saudaranya...")*(HR. Muslim)

Hadis tersebut mempunyai makna yang cukup mendalam. Bila diperhatikan secara seksama, hadis tersebut mengandung makna bahwa Allah melibatkan secara langsung. Seolah-oleh berada di balik orang-orang yang kesusahan dan siap memberikan balasan yang setimpal bagi orang yang mau membantu atau tolong-menolong terhadap orang-orang yang dililit kesulitan itu. Secara verbal, Allah berjanji memudahkan segala urusan dan menolong orang yang mau menolong hamba-Nya. (Luhur, 2017)

Allah memberikan kedudukan istimewa bagi orang-orang yang gemar mengulurkan tangannya (menolong) orang lain. Terdapat sebuah penggalan kisah yang dapat dipelajari. Diceritakan kepada kami Muhammad bin Abdurrahman Asy-Syafi'i, berkata kepada kami Al-Qasim pun Hasyim As-Samsar, ia berkata : telah menceritakan kepada kami Abdurrahman bin Qais Adl-Dlibibi, ia berkata: telah menceritakan kepada kami Suakin bin Siraj, berkata kepada kami Amr bin Dinar, dari Ibnu Umar bahwa seorang laki-laki datang kepada Rasulullah SAW, maka ia bertanya: "Ya Rasulullah, siapakah orang yang paling dicintai Allah? Dan apakah amal yang paling dicintai Allah *asa wa jala*"? Rasulullah SAW pun menjawab :

أحبُّ الناسِ إلى اللهِ تعالى أنفعُهم للناسِ وأحبُّ الأعمالِ إلى اللهِ عزَّ وجلَّ سرورٌ يُدخلُه على مسلمٍ أو يكشفُ عنه كُربةً أو يقضي عنه دَينًا أو يطردُ عنه جوعًا ولأن أمشيَ مع أخٍ في حاجةٍ أحبُّ إليَّ من أن أعتكفَ في هذا المسجدِ ( يعني مسجدَ المدينةِ ) شهرًا

Artinya : "*Orang yang paling dicintai Allah adalah yang paling bermanfaat untuk orang lain. Dan perbuatan yang paling dicintai Allah adalah memberi kegembiraan seorang mukmin, menghilangkan salah satu kesusahannya, membayarkan hutangnya, atau menghilangkan rasa laparnya. Dan aku berjalan bersama saudaraku untuk memenuhi kebutuhannya itu lebih aku cintai daripada*

*beri'tikaf di masjid Nabawi selama sebulan."* (HR. Thabrani dalam Mu'jam Al-Kabir li Ath-Thabrani juz 11 hlm.84).

Tiada yang paling beruntung dalam kehidupan manusia ini selain dicintai oleh Sang Maha Mencintai yaitu Allah tuhan semesta alam. Kecintaan-Nya kepada manusia adalah anugerah terbesar, bahkan jikalau dibandingkan dengan kenikmatan surga ciptaan Allah sekalipun.

Dengan semakin berkembangnya teknologi informasi, manusia cenderung menjadi makhluk yang individualis. Hal itu bukan berarti tidak bisa untuk saling tolong menolong sesama manusia. Salah satu bentuk tolong menolong di bidang teknologi informasi adalah membuat sebuah aplikasi untuk memudahkan kebutuhan orang lain. Dengan dibangunnya sebuah aplikasi untuk mencari rute tercepat, dapat memudahkan pengguna untuk mendapatkan informasi mengenai rute tercepat pada hari dan waktu tersebut.

Aplikasi ini dibangun dengan menggunakan algoritma *dijkstra* dan *multiple regression*. Dengan tujuan agar diketahui akurasi penentuan rute tercepat dan waktu tempuh yang dihasilkan. Algoritma *multiple regression* yang memiliki ukuran tersendiri dalam perhitungan bobot berdasarkan parameter-parameternya. Disebutkan pada firman Allah surat Al-Furqon ayat 2 yang berbunyi:

ٱلَّذِى لَهُۥ مُلۡكُ ٱلسَّمَٰوَٰتِ وَٱلۡأَرۡضِ وَلَمۡ يَتَّخِذۡ وَلَدٗا وَلَمۡ يَكُن لَّهُۥ شَرِيكٞ فِي ٱلۡمُلۡكِ وَخَلَقَ كُلَّ شَيۡءٖ فَقَدَّرَهُۥ تَقۡدِيرٗا ٢

Artinya : *yang kepunyaan-Nya-lah kerajaan langit dan bumi, dan Dia tidak mempunyai anak, dan tidak ada sekutu bagi-Nya dalam kekuasaan(Nya), dan dia telah menciptakan segala sesuatu, dan Dia menetapkan ukuran-ukurannya dengan serapi-rapinya*.

Algoritma *multiple regression* mempunyai ukuran tersendiri dalam proses perhitungan bobotnya, begitu juga algoritma *dijkstra* yang memiliki ukuran

tersendiri dalam penentuan rute tercepat. Karena itu pada penelitian ini dilakukan sebuah percobaan untuk mencari nilai akurasi sistem dengan acuan nilai *error* yang dihasilkan dari percobaan. Hasil dari penelitian ini menunjukkan tingkat akurasi dari aplikasi yang telah dibuat, sebagai pembelajaran bahwa untuk memperoleh rute tercepat dengan menggunakan aplikasi ini memiliki akurasinya.

# BAB V

# PENUTUP

Bab ini merupakan penjelasan dari kesimpulan yang didapatkan dari hasil uji coba dan pembahasan dari penelitian ini. Serta dijelaskan mengenai saran yang dibutuhkan dalam pengembangan penelitian ini.

## 5.1 Kesimpulan

Berdasarkan analisa hasil dari uji coba yang telah dilakukan dan pembahasan mengenai pencarian rute tercepat menggunakan algoritma dijkstra dan *multiple regression*, dapat disimpulkan bahwa pembobotan yang dilakukan dengan metode *multiple regression* dan penentuan rute dengan algoritma *dijkstra* menghasilkan nilai rata-rata persentase *error* adalah sebesar 11,21% dengan persentase *error* yang terkecil sebesar 3,67% dan persentase *error* terbesar adalah sebesar 20,98%. Dan Nilai persentase keakuratan sistem sebesar 88,79% terhadap 10 percobaan yang diujikan. Sedangkan nilai rata-rata persentase *error* yang dihasilkan oleh Google *maps* adalah 38,30% sehingga nilai akurasi yang didapatkan adalah sebesar 61,70%. Penyebab besarnya persentase *error* yang dihasilkan oleh Google *maps* dikarenakan bergantungnya sistem Google *maps* pada fitur lokasi yang aktif dari setiap *device*. Sedangkan aplikasi ini melakukan pengambilan data jumlah kendaraan dan data kecepatan rata-rata secara langsung sehingga mempengaruhi estimasi waktu yang dihasilkan. Oleh karena itu, sistem ini memiliki persentase *error* yang dihasilkan lebih kecil dan persentase nilai akurasi yang lebih besar dibandingkan dengan sistem yang dimiliki oleh Google.

## 5.2 Saran

Berbagai kegiatan telah dilakukan untuk menyelesaikan penelitian ini. Namun terdapat beberapa saran yang perlu dilakukan yang mungkin berguna sebagai pengembangan untuk penelitian selanjutnya. Beberapa saran yang perlu dilakukan adalah sebagai berikut :

1. Memperbanyak variasi data pada data *training* untuk mengurangi persentase *error* yang dihasilkan oleh sistem.
2. Menggunakan metode penentuan rute yang lain untuk perbandingan.
3. Menggunakan data *real-time* pada parameter jumlah kendaraan.
4. Menambah arus bolak-balik pada rute yang ditentukan.
5. Menambah data pada lampu merah, simpangan jalan, dan penyebrangan agar tingkat persentase *error* yang dihasilkan oleh sistem semakin kecil.
6. Menambah rute dan tujuan.

## DAFTAR PUSTAKA


Algifari. (2000). *Analisis Regresi : Teori, Kasus, dan Solusi edisi 2.* Yogyakarta: BPFE UGM.

Capra, S. C. (2012). *Applied Numerical Method with Matlab for Engineers and Scientists : Errors Page 91.* New York: McGraw-Hill Companies.

Dzulfikri, A. (2019). *Perbandingan Metode Dice Similarity Dengan Cosine Similarity Menggunakan Query Expansion Pada Pencarian Ayatul Ahkam Dalam Terjemah Alquran Berbahasa Indonesia.* Malang: UIN Maulana Malik Ibrahim Malang.

Geovani, H. A. (2016). *Implementasi Algoritma Dijkstra Untuk Mengetahui Lokasi Tempat Ibadah Umat Muslim Di Kota Malang Pada Aplikasi Mobile Phone (Studi Kasus Tempat Ibadah Di Wilayah Kecamatan Lowokwaru).* Malang: UIN Maulana Malik Ibrahim.

Irawan. (2012). *Membuat Aplikasi Android Untuk Orang Awam.* Palembang: Maxikom.

Luhur, A. B. (2017, September 16). *Keistimewaan Gemar Menolong Orang Lain*. Diambil kembali dari NU Online | Suara Nahdlatul Ulama: http://www.nu.or.id/post/read/76171/keistimewaan-gemar-menolong-orang-lain

Mangkuatmodjo, S. (2004). *Statistik Lanjutan.* Jakarta: Rineka Cipta.

Muhammad, J., & Abdurrahman, J. (1983). *Tafsir Jalalain.* Terjemahan oleh K.H. Misbah bin Zain Al-Musthofa Bangilan. Surabaya: Al-Hidayah.

Mustafar, I. b. (2011). A Study on Prediction of Output in Oilfield Using Multiple Linear Regression. *International Journal of Applied Science and Technology*, Vol. 1, No. 4.

Pugas, D. O., Somantri, M., & Satoto, K. I. (2011). Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra dan Astar (A*) pada SIG Berbasis Web untuk Pemetaan Pariwisata Kota Sawahlunto. *Transmisi ISSN 1411 - 0814*, Vol. 13(1), 27-32.

Purwananto, Y., Purwitasari, D., & Wibowo, A. W. (2005). Implementasi dan Analisis Algoritma Pencarian Rute Terpendek di Kota Surabaya. *Jurnal Penelitian dan Pengembangan Telekomunikasi*, Vol. 10, No. 2.

Putri, A. R. (2018, Juli 27). *Bagaimana Cara Google Maps Mendeteksi Kemacetan Jalan Raya?* Diambil kembali dari kumparan.com: https://kumparan.com/@kumparantech/bagaimana-cara-google-maps-mendeteksi-kemacetan-jalan-raya-27431110790554544

Ratri, N. (2017, Maret 8). *Malang Raya Termacet Keempat di Asia ?* Diambil kembali dari Radar Malang: http://www.radarmalang.id/malang-raya-termacet-keempat-di-asia/

Rozikin, C., & Solichin, A. (2017). Implementasi Algoritma Genetika dan Regresi Linier Berganda Untuk Prediksi Persediaan Bahan Makanan Pada Restoran Cepat Saji. *Prosiding Seminar Nasional Multidisiplin Ilmu ISSN : 2087 - 0930*, 10-17.

Safaat, N. (2011). *Pengembangan Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android.* Bandung: Informatika.

Saputro, S. S. (2013). *Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata Di Kota Manado Berbasis Mobile Web Dengan Algoritma Dijkstra.* Semarang: Universitas Dian Nuswantoro.

Shihab, M. Q. (2000). *Tafsir Al-Mishbah : Pesan, Kesan dan Keserasian Alquran Vol.3.* Jakarta: Lentera Hati.

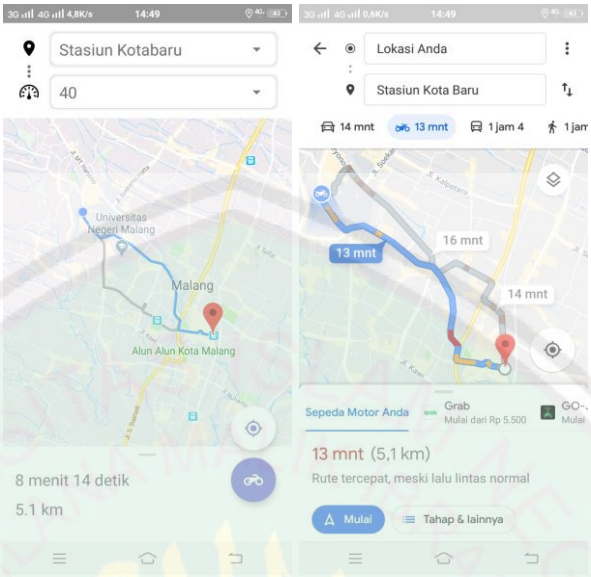
Sulaiman, W. (2004). *Analisis Regresi Menggunakan SPSS Contoh Kasus & Pemecahannya.* Yogyakarta: Andi.

Thabrani, H. (t.thn.). *Mu'jam Kabir li Ath-Thabrani juz 11 hlm.84.*

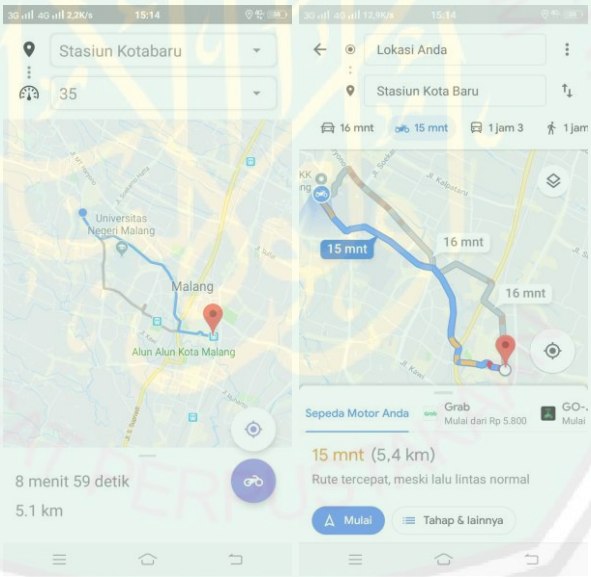
Wicaksono, S. A. (2016, Maret 26). *Sistem Penentuan Harga Buah Berdasarkan Tingkat Kematangan dan Berat Buah Berbasis Image Menggunakan Metode Euclidean Distance dan Multiple Linier Regression.* Malang: UIN Maulana Malik Ibrahim.

Yulia R, W. E., Istiadi, D., & Roqib, A. (2015). Pencarian SPBU Terdekat dan Penentuan Jarak Terpendek Menggunakan Algoritma Dijkstra (Studi Kasus Di Kabupaten Jember). *Jurnal Nasional Teknik Elektro ISSN: 2302 - 2949*, Vol: 4, No. 1.

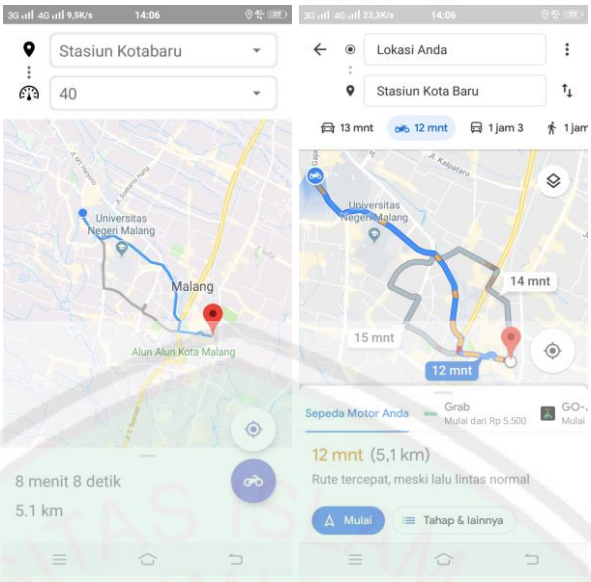
# LAMPIRAN



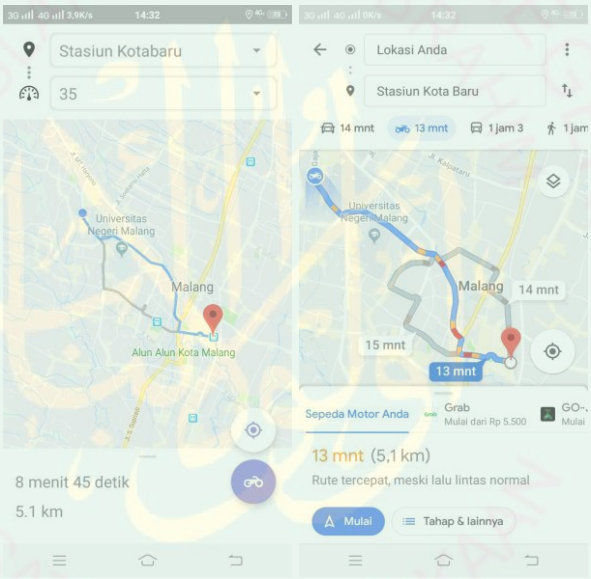

1. *Screenshoot* Aplikasi dan Google *maps* percobaan pertama.



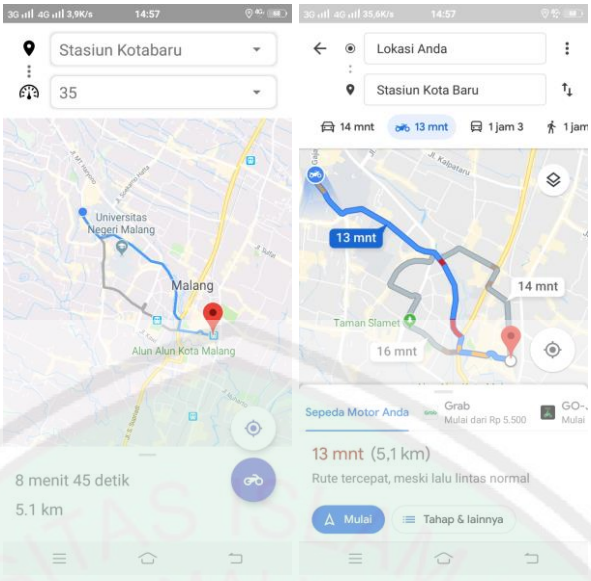

2. *Screenshoot* Aplikasi dan Google *maps* percobaan kedua.



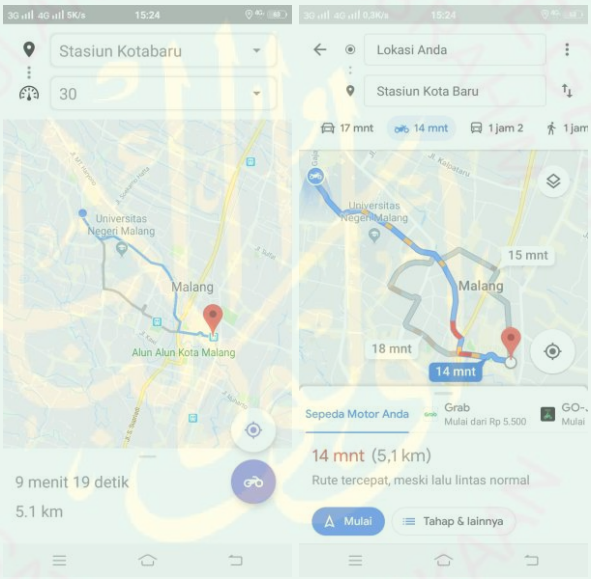

3. *Screenshoot* Aplikasi dan Google *maps* percobaan ketiga.



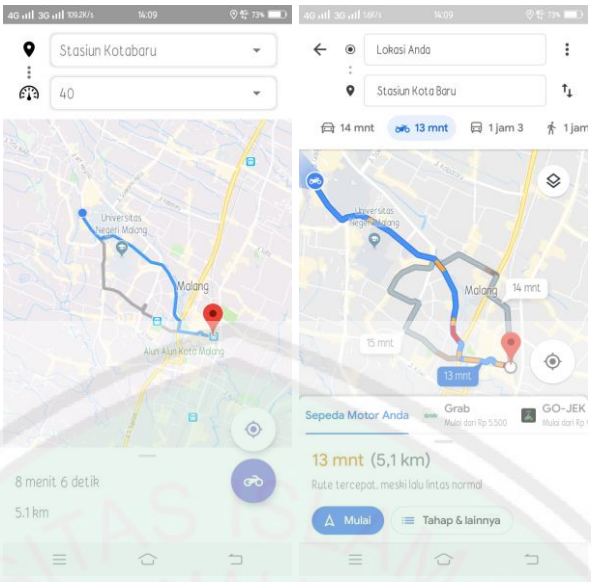

4. *Screenshoot* Aplikasi dan Google *maps* percobaan keempat.



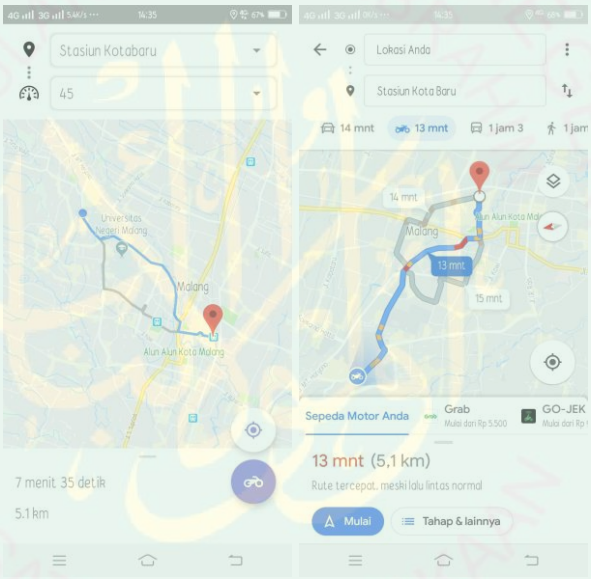

5. *Screenshoot* Aplikasi dan Google *maps* percobaan kelima.



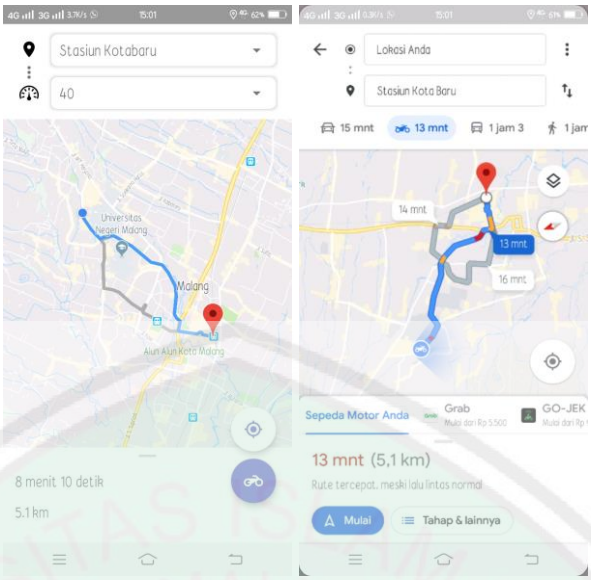

6. *Screenshoot* Aplikasi dan Google *maps* percobaan keenam.



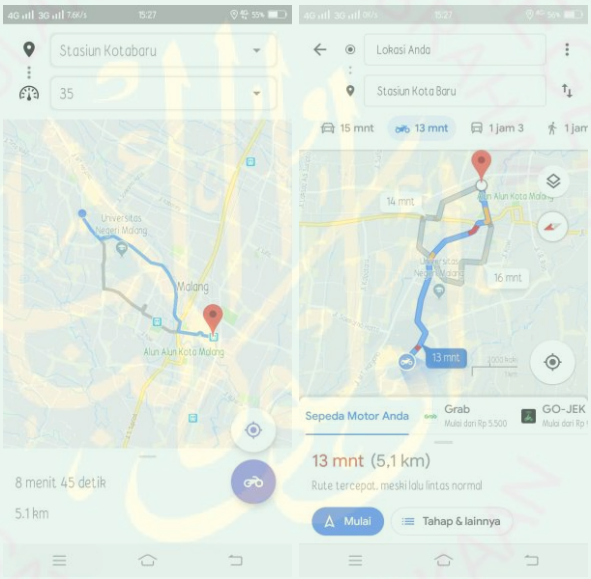

7. *Screenshoot* Aplikasi dan Google *maps* percobaan ketujuh.



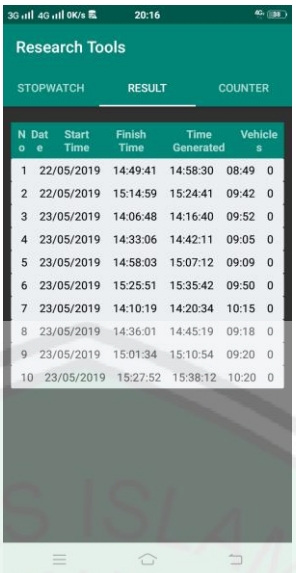

8. *Screenshoot* Aplikasi dan Google *maps* percobaan kedelapan.




9. *Screenshoot* Aplikasi dan Google *maps* percobaan kesembilan.




10. *Screenshoot* Aplikasi dan Google *maps* percobaan kesepuluh.




11. *Screenshoot* hasil percobaan.

12. Tabel estimasi parameter percobaan pertama.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 40 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 40 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 40 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 40 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 40 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 40 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 40 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 40 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 40 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 40 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 40 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 40 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 40 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 40 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 40 |

13. Tabel estimasi parameter percobaan kedua.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 35 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 35 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 35 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 35 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 35 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 35 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 35 |

13. Tabel estimasi parameter percobaan kedua (lanjutan).

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 35 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 35 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 35 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 35 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 35 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 35 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 35 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 35 |

14. Tabel estimasi parameter percobaan ketiga.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 40 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 40 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 40 |
| 4 | 4 - 5 | 0,77 | 0,01 | -0,01 | 31 | 40 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 40 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 40 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 40 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 40 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 40 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 40 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 40 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 40 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 40 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 40 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 40 |

15. Tabel estimasi parameter percobaan keempat.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 35 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 35 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 35 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 35 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 35 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 35 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 35 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 35 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 35 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 35 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 35 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 35 |

15. Tabel estimasi parameter percobaan keempat (lanjutan).

| | | | | | | |
|---|---|---|---|---|---|---|
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 35 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 35 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 35 |

16. Tabel estimasi parameter percobaan kelima.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 35 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 35 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 35 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 35 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 35 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 35 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 35 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 35 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 35 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 35 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 35 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 35 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 35 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 35 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 35 |

17. Tabel estimasi parameter percobaan keenam.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 30 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 30 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 30 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 30 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 30 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 30 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 30 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 30 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 30 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 30 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 30 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 30 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 30 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 30 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 30 |

18. Tabel estimasi parameter percobaan ketujuh.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 40 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 40 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 40 |
| 4 | 4 - 5 | 1,15 | 0,00 | -0,01 | 40 | 40 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 40 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 40 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 40 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 40 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 40 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 40 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 40 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 40 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 40 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 40 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 40 |

19. Tabel estimasi parameter percobaan kedelapan.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 45 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 45 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 45 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 45 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 45 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 45 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 45 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 45 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 45 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 45 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 45 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 45 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 45 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 45 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 45 |

20. Tabel estimasi parameter percobaan kesembilan.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 40 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 40 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 40 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 40 |

20. Tabel estimasi parameter percobaan kesembilan (lanjutan).

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 40 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 40 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 40 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 40 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 40 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 40 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 40 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 40 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 40 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 40 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 40 |

21. Tabel estimasi parameter percobaan kesepuluh.

| No | Node | a | b1 | b2 | x1 | x2 |
|---|---|---|---|---|---|---|
| 1 | 1 - 2 | 2,14 | 0,01 | -0,03 | 55 | 35 |
| 2 | 2 - 3 | 0,30 | 0,01 | 0,00 | 73 | 35 |
| 3 | 3 - 4 | 2,12 | 0,01 | -0,04 | 130 | 35 |
| 4 | 4 - 5 | 0,11 | 0,01 | 0,01 | 16 | 35 |
| 5 | 5 - 6 | 0,97 | 0,02 | -0,01 | 22 | 35 |
| 6 | 6 - 7 | 0,82 | 0,02 | -0,01 | 12 | 35 |
| 7 | 7 - 8 | -0,26 | 0,01 | 0,01 | 110 | 35 |
| 8 | 8 - 9 | 2,18 | 0,01 | -0,03 | 45 | 35 |
| 9 | 9 - 10 | 1,45 | 0,00 | 0,01 | 83 | 35 |
| 10 | 10 - 11 | 0,43 | 0,00 | 0,00 | 26 | 35 |
| 11 | 11 - 12 | 0,96 | 0,00 | -0,01 | 21 | 35 |
| 12 | 2 - 13 | 1,41 | 0,01 | -0,01 | 75 | 35 |
| 13 | 13 - 14 | 1,10 | 0,01 | -0,01 | 45 | 35 |
| 14 | 14 - 15 | 2,48 | 0,01 | -0,02 | 126 | 35 |
| 15 | 15 - 9 | 0,52 | 0,00 | -0,01 | 22 | 35 |

22. Tabel perhitungan persentase galat *error*.

| No | Hari | Tanggal | Input Kecepatan | Start | Finish | Nilai riil (Exact value) | | Aplikasi | | Google Maps | | Aplikasi | | Google Maps | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Nilai dugaan (Approximate value) | | Nilai dugaan (Approximate value) | | | | | |
| | | | | | | Jam | Desimal | Jam | Desimal | Jam | Desimal | %error | | %error | |
| 1 | Rabu | 22/05/2019 | 40 | 14:49:41 | 14:58:30 | 00:08:49 | 8,82 | 00:08:14 | 8,23 | 00:13:00 | 13 | 6,62 | % | 47,45 | % |
| 2 | Rabu | 22/05/2019 | 35 | 15:14:59 | 15:24:41 | 00:09:42 | 9,70 | 00:08:59 | 8,98 | 00:15:00 | 15 | 7,39 | % | 54,64 | % |
| 3 | Kamis | 23/05/2019 | 40 | 14:06:48 | 14:16:40 | 00:09:52 | 9,87 | 00:08:08 | 8,13 | 00:12:00 | 12 | 17,57 | % | 21,62 | % |
| 4 | Kamis | 23/05/2019 | 35 | 14:33:06 | 14:42:11 | 00:09:05 | 9,08 | 00:08:45 | 8,75 | 00:13:00 | 13 | 3,67 | % | 43,12 | % |
| 5 | Kamis | 23/05/2019 | 35 | 14:58:03 | 15:07:12 | 00:09:09 | 9,15 | 00:08:45 | 8,75 | 00:13:00 | 13 | 4,37 | % | 42,08 | % |
| 6 | Kamis | 23/05/2019 | 30 | 15:25:51 | 15:35:42 | 00:09:50 | 9,83 | 00:09:19 | 9,32 | 00:14:00 | 14 | 5,25 | % | 42,37 | % |
| 7 | Kamis | 23/05/2019 | 40 | 14:10:19 | 14:20:34 | 00:10:15 | 10,25 | 00:08:06 | 8,10 | 00:13:00 | 13 | 20,98 | % | 26,83 | % |
| 8 | Kamis | 23/05/2019 | 45 | 14:36:01 | 14:45:19 | 00:09:18 | 9,30 | 00:07:35 | 7,58 | 00:13:00 | 13 | 18,46 | % | 39,78 | % |
| 9 | Kamis | 23/05/2019 | 40 | 15:01:34 | 15:10:54 | 00:09:20 | 9,33 | 00:08:10 | 8,17 | 00:13:00 | 13 | 12,50 | % | 39,29 | % |
| 10 | Kamis | 23/05/2019 | 35 | 15:27:52 | 15:38:12 | 00:10:20 | 10,33 | 00:08:45 | 8,75 | 00:13:00 | 13 | 15,32 | % | 25,81 | % |
| **Error Minimum** | | | | | | | | | | | | **3,67** | **%** | **21,62** | % |
| **Error Maksimum** | | | | | | | | | | | | **20,98** | **%** | **54,64** | % |
| **Error Rata-rata** | | | | | | | | | | | | **11,21** | **%** | **38,30** | % |
| **Akurasi** | | | | | | | | | | | | **88,79** | **%** | **61,70** | % |