

**TUGAS AKHIR – KI003498**

**APLIKASI ANAMNESIS BERDASARKAN GEJALA MENGGUNAKAN *FREQUENT PATTERN TREE GROWTH***

**KEVIN ALIF FACHREZA**

**NRP 05111440000128**

Dosen Pembimbing I

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom

Dosen Pembimbing II

Anny Yuniarti, S.Kom, M.Comp.Sc

Departemen Informatika

Fakultas Teknologi Informasi dan Komunikasi

Institut Teknologi Sepuluh Nopember

Surabaya 2018

***(Halaman ini sengaja dikosongkan)***



**TUGAS AKHIR – KI003498**

**APLIKASI ANAMNESIS BERDASARKAN GEJALA MENGGUNAKAN *FREQUENT PATTERN TREE GROWTH***

**KEVIN ALIF FACHREZA**

**NRP 05111440000128**

**Dosen Pembimbing I**

**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom**

**Dosen Pembimbing II**

**Anny Yuniarti, S.Kom, M.Comp.Sc**

**Departemen Informatika**

**Fakultas Teknologi Informasi dan Komunikasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2018**

***(Halaman ini sengaja dikosongkan)***



**UNDERGRADUATE THESES – KI003498**

***ANAMNESA APPLICATION BASED ON APPLICATION USING FREQUET PATTERN TREE GROWTH***

**KEVIN ALIF FACHREZA**

**NRP 05111440000128**

**First Advisor**

**Dr. Eng. Chastine Fatichah, S.Kom., M.Kom**

**Second Advisor**

**Anny Yuniarti, S.Kom, M.Comp.Sc**

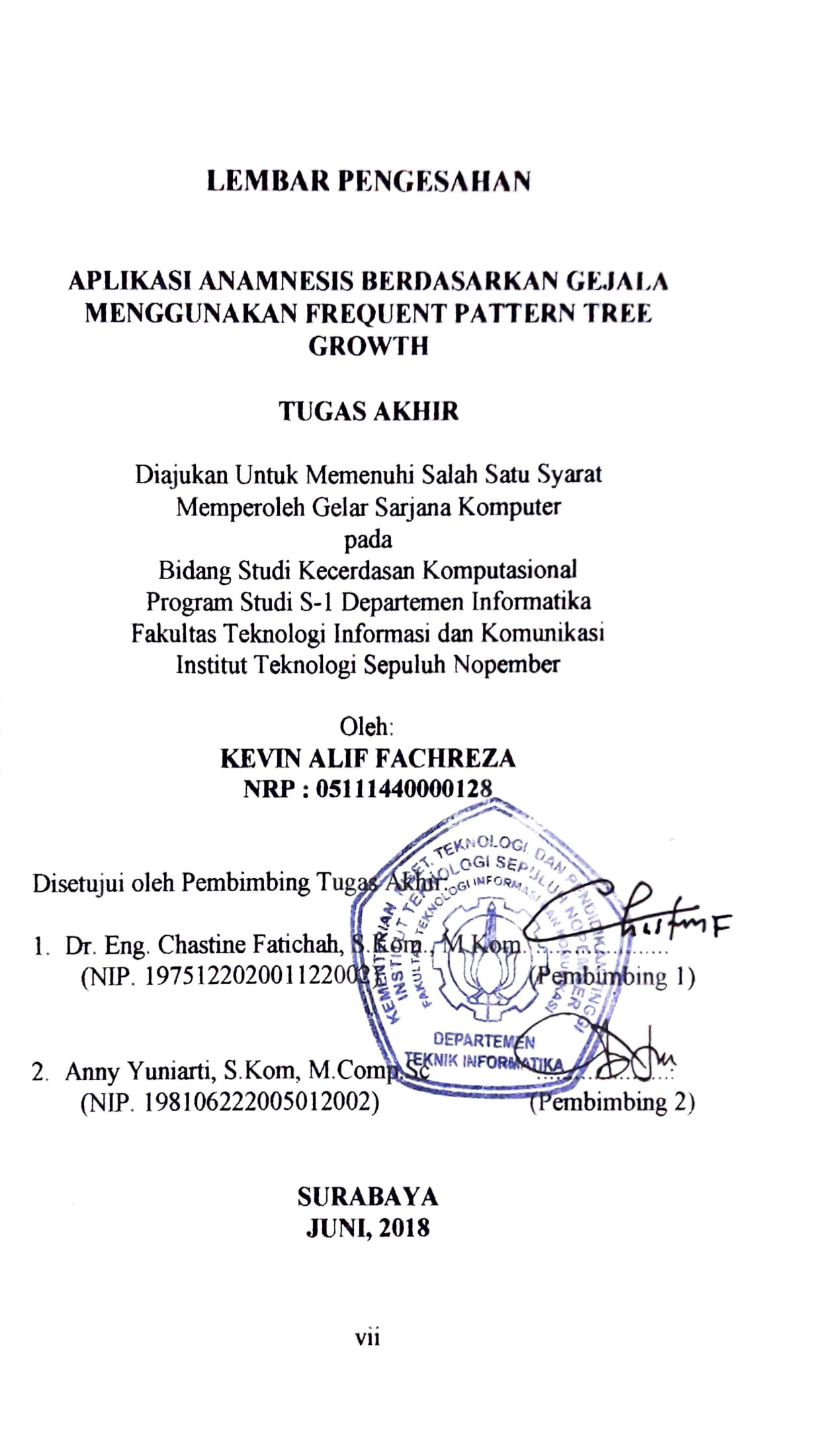
**Department of Informatics**

**Faculty of Information and Communication Technology**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2018**

***(Halaman ini sengaja dikosongkan)***

**LEMBAR PENGESAHAN**

**APLIKASI ANAMNESIS BERDASARKAN GEJALA MENGGUNAKAN FREQUENT PATTERN TREE GROWTH**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Kecerdasan Komputasional

Program Studi S-1 Departemen Informatika

Fakultas Teknologi Informasi dan Komunikasi

Institut Teknologi Sepuluh Nopember

Oleh:

**KEVIN ALIF FACHREZA**

**NRP : 05111440000128**

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. .....................

(NIP. 197512202001122002) (Pembimbing 1)

1. Anny Yuniarti, S.Kom, M.Comp.Sc ......................

(NIP. 198106222005012002) (Pembimbing 2)

**SURABAYA**

**JUNI, 2018**

***(Halaman ini sengaja dikosongkan)***

**APLIKASI ANAMNESIS BERDASARKAN GEJALA MENGGUNAKAN *FREQUENT PATTERN TREE GROWTH***

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa** | **:** | **Kevin Alif Fachreza** |
| **NRP** | **:** | **05111440000128** |
| **Departemen** | **:** | **Informatika FTIK-ITS** |
| **Dosen Pembimbing 1** | **:** | **Dr. Eng. Chastine Fatichah, S.Kom., M.Kom** |
| **Dosen Pembimbing 2** | **:** | **Anny Yuniarti, S.Kom, M.Comp.Sc** |

# Abstrak

*Machine learning* merupakan salah satu bidang di teknologi informasi yang sedang naik daun. Teknologi ini dimanfaatkan diberbagai bidang kehidupan manusia, mulai dari perbankan, transportasi, sosial media, termasuk kesehatan. Potensi pemanfaatan *machine learning* pada bidang kesehatan sangatlah besar. Salah satunya adalah diagnosis penyakit. Diagnosis penyakit atau dikenal dengan tahap anamnesis pada kedokteran adalah suatu proses dimana dokter akan menanyakan kepada pasien gejala gejala yang dialami pasien. Sehingga dokter dapat memperkecil kemungkinan penyakit yang mungkin dialami pasien dan melakukan tes penunjang seperti lab atau radiologi untuk mendapatkan keputusan diagnosis final.

Banyak aplikasi yang dapat mendiagnosis penyakit, akan tetapi *user experience* yang buruk menyebabkan diagnosis meleset. Aplikasi tersebut biasanya akan meminta pengguna untuk memasukkan gejala mereka baik teks maupun berupa *checkboxes*. Padahal pasien tidak terlalu mengetahui gejala gejala spesifik atau yang berkorelasi yang dapat membantu dokter dalam mendiagnosis penyakit secara signifikan.

Aplikasi yang dibuat pada tugas akhir ini akan berfokus pada proses anamnesis, proses dimana pasien memberikan informasi gejala yang dialami dengan spesifik dan sejelas mungkin. Sehingga dokter dapat memberikan diagnosis yang lebih baik. Untuk menyelesaikan masalah tersebut, akan digunakan algoritma *Frequent Pattern Tree Growth (FP Tree)*, yaitu salah satu metode *association rules* dimana algoritma ini dapat memetakan gejala gejala yang berkaitan, sehingga aplikasi dapat menanyakan gejala yang mungkin dialami oleh pasien berdasarkan gejala yang telah dimasukkan sebelumnya oleh pasien.

Dengan jumlah diagnosis penyakit sebanyak 66 penyakit, aplikasi ini memberikan hasil yang cukup baik. Dibuktikan dengan skenario pengujian dengan 198 data menghasilkan rata rata akurasi sebesar 69% dengan *classifier Naïve Bayes.* Dan rata rata akurasi sebesar 67% dengan *classifier Support Vector Machine.* Aplikasi juga diujikan kepada dokter spesialis dan mengatakan bahwa aplikasi memiliki kinerja baik pada proses anamnesis dan memberikan diagnosis yang sebagian besar sudah sesuai.

**Kata kunci: *Frequent Pattern Tree Growth*, anamnesis, diagnosis, *Naïve Bayes, Support Vector Machine***

**ANAMNESA BASED ON SYMPTOMS APPLICATION USING FREQUENT PATTERN TREE GROWTH**

|  |  |  |
| --- | --- | --- |
| **Student’s Name** | **:** | **Kevin Alif Fachreza** |
| **Student’s ID** | **:** | **05111440000128** |
| **Departemen** | **:** | **Informatika FTIK-ITS** |
| **First Advisor** | **:** | **Dr. Eng. Chastine Fatichah, S.Kom., M.Kom** |
| **Second Advisor** | **:** | **Anny Yuniarti, S.Kom, M.Comp.Sc** |

# Abstract

*Machine learning is currently one of the promising tech. This technology can be used in any fields, banking, transportation, social media, include health. Machine learning has a huge potency in health industry. One of them is diseases diagnosis. Diseases diagnosis or known as anamnesis in medical is a process where doctor will ask the patient about the symptoms or experience that patient felt. So doctor can narrow the possibility of diseases that the patient may suffer and do some lab and radiology tests to get final diagnosis decision*

*There is a lot app that able to diagnose diseases, but with bad user experience made the diagnosis went fumble. Those apps will asks user to input their symptoms using text or checkboxes. Whereas patients do not really know the specific symptoms or the symptoms that correlated to their main symptoms which can help doctor to diagnose disease significantly.*

*The app that made for this final project focused on anamnesis, where patients give information about their symptoms specifically and clearly. So doctor can give better diagnosis decision. To solve that problem, Frequent Pattern Tree Growth algorithm was used, Frequent Pattern Tree Growth is one of the association rules methods where this algorithm can map which symptoms is related to other symptoms, so the app can ask the next symptoms based on the previous symptoms that inputted by user.*

*With 66 diagnoses, this app gave a pretty good result. Proven by scenarios of testing with 198 datas, yielding 69% average accuracy with classifier Naïve Bayes. And 67% average accuracy with classifier Support Vector Machine. This app also tested to a doctor specialist and said that the app gave good result in anamnesis and diagnoses that given by the app were mostly correlated based on anamnesis input.*

***Keywords : Frequent Pattern Tree Growth, anamnesis, diagnosis, Naïve Bayes, Support Vector Machine***

# KATA PENGANTAR

Alhamdulillahirabbil’alamin, segala puji dan syukur bagi Allah SWT, yang telah melimpahkan rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **“Aplikasi Anamnesis Berdasarkan Gejala Menggunakan Frequent Pattern Tree Growth”**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Keluarga penulis (Mama, Papa, Yangti, Yangkong, Rehun, dan keluarga penulis yang lain) yang selalu memberikan dukungan moral dan material yang tidak ternilai.
3. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
4. Ibu Anny Yuniarti, S.Kom, M.Comp.Sc selaku pembimbing II Tugas Akhir yang telah memberikan banyak yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.
5. Dr Adi Suriyanto, Sp.OT selaku dokter yang meluangkan waktu dan pikiran dalam memberikan masukan dan melakukan pengujian pada Tugas Akhir ini.
6. Annisa Rahmawati, S.KG yang memberikan dukungan moral, motivasi, masukan dan hiburan pada pengerjaan Tugas Akhir ini.
7. Teman teman Jojoran yang menyediakan informasi dan tempat serta fasilitas dalam mengerjakan Tugas Akhir ini.
8. Yang terakhir untuk orang orang yang membaca dan akan mengembangkan Tugas Akhir ini. Semoga dapat mewujudkan aplikasi yang diimpikan.

Penulis memohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, 30 Juni 2018

Kevin A Fachreza

# DAFTAR ISI

[LEMBAR PENGESAHAN vii](#_Toc518512386)

[Abstrak ix](#_Toc518512387)

[Abstract xi](#_Toc518512388)

[KATA PENGANTAR xiii](#_Toc518512389)

[DAFTAR ISI xv](#_Toc518512390)

[DAFTAR GAMBAR xix](#_Toc518512391)

[DAFTAR TABEL xxi](#_Toc518512392)

[DAFTAR KODE SUMBER xxiii](#_Toc518512393)

[BAB I PENDAHULUAN 1](#_Toc518512394)

[1.1 Latar Belakang 1](#_Toc518512395)

[1.2 Rumusan Masalah 2](#_Toc518512396)

[1.3 Batasan Masalah 2](#_Toc518512397)

[1.4 Tujuan Pembuatan Tugas Akhir 2](#_Toc518512398)

[1.5 Manfaat Tugas Akhir 2](#_Toc518512399)

[1.6 Metodologi 2](#_Toc518512400)

[1.7 Sistematika Penulisan Laporan Tugas Akhir 4](#_Toc518512401)

[BAB II DASAR TEORI 7](#_Toc518512402)

[2.1 Anamnesis 7](#_Toc518512403)

[2.1.1 Pengertian Anamnesis 7](#_Toc518512404)

[2.1.2 Langkah Langkah Anamnesis 7](#_Toc518512405)

[2.2 Frequent Pattern Tree Growth 9](#_Toc518512406)

[2.2.1 Pengertian Frequent Pattern Tree Growth 9](#_Toc518512407)

[2.2.2 Support 9](#_Toc518512408)

[2.2.3 Algoritma FP Tree Growth 10](#_Toc518512409)

[2.3 Naïve Bayes 11](#_Toc518512410)

[2.3.1 Jenis *Naïve Bayes* 12](#_Toc518512411)

[*2.4 Support Vector Machine* 13](#_Toc518512412)

[*2.4.1 Hyperplane* 13](#_Toc518512413)

[2.4.2 *Kernel* SVM 13](#_Toc518512414)

[2.4.3 SVM Multiclass 14](#_Toc518512415)

[2.5 *Scikit Learn* 14](#_Toc518512416)

[2.6 *Python – Flask* 15](#_Toc518512417)

[2.7 *PHP - Laravel* 15](#_Toc518512418)

[BAB III PERANCANGAN 17](#_Toc518512419)

[3.1 Perancangan Data 17](#_Toc518512420)

[3.2 Desain Sistem Secara Umum 19](#_Toc518512421)

[3.2.1 Pembangunan Dataset 19](#_Toc518512422)

[3.2.2 Perancangan Model *FP Tree* 21](#_Toc518512423)

[3.2.3 Perancangan Model *Classifier* 30](#_Toc518512424)

[3.2.4 Metode Evaluasi 31](#_Toc518512425)

[3.2.5 Perancangan Aplikasi 32](#_Toc518512426)

[BAB IV IMPLEMENTASI 36](#_Toc518512427)

[4.1 Lingkungan Implementasi 37](#_Toc518512428)

[4.2 Implementasi 37](#_Toc518512429)

[4.2.1 Implementasi FP Tree 37](#_Toc518512430)

[4.2.2 Implementasi *Classifier* 42](#_Toc518512431)

[4.2.3 Implementasi Evaluasi 46](#_Toc518512432)

[4.2.4 Implementasi Aplikasi 46](#_Toc518512433)

[BAB V PENGUJIAN DAN EVALUASI 65](#_Toc518512434)

[5.1 Lingkungan Pengujian 65](#_Toc518512435)

[5.2 Data Uji Coba 65](#_Toc518512436)

[5.3 Skenario Uji Coba 67](#_Toc518512437)

[5.4 Hasil Uji Coba 68](#_Toc518512438)

[5.4.1 Hasil Uji Sistem 68](#_Toc518512439)

[5.4.2 Hasil Uji Pengguna 73](#_Toc518512440)

[5.4.3 Hasil Uji Dokter 74](#_Toc518512441)

[5.5 Analisis Hasil Uji Coba 75](#_Toc518512442)

[BAB VI KESIMPULAN DAN SARAN 77](#_Toc518512443)

[6.1 Kesimpulan 77](#_Toc518512444)

[6.2 Saran 77](#_Toc518512445)

[DAFTAR PUSTAKA 79](#_Toc518512446)

[LAMPIRAN 82](#_Toc518512447)

[1. Hasil Lengkap Uji Sistem 83](#_Toc518512448)

[2. Hasil Lengkap Uji Dokter 91](#_Toc518512449)

[A. Uji Coba 1 91](#_Toc518512450)

[B. Uji Coba 2 92](#_Toc518512451)

[C. Uji Coba 3 93](#_Toc518512452)

[D. Uji Coba 4 94](#_Toc518512453)

[E. Uji Coba 5 95](#_Toc518512454)

[F. Uji Coba 6 96](#_Toc518512455)

[G. Uji Coba 7 97](#_Toc518512456)

[H. Uji Coba 8 98](#_Toc518512457)

[I. Uji Coba 9 99](#_Toc518512458)

[J. Uji Coba 10 100](#_Toc518512459)

# 

# DAFTAR GAMBAR

[Gambar 2.1 Algoritma SVM 13](#_Toc518889319)

[Gambar 2.2 Contoh *Hyperplane* 13](#_Toc518889320)

[Gambar 3.1 Diagram Sistem Secara Umum 20](#_Toc518889321)

[Gambar 3.2 Diagram Algoritma FP Tree 22](#_Toc518889322)

[Gambar 3.3 Hasil *Tree* Data Sampel 25](#_Toc518889323)

[Gambar 3.4 Diagram Alur Aplikasi 33](#_Toc518889324)

[Gambar 3.5 Diagram Algoritma Tanya Jawab 34](#_Toc518889325)

[Gambar 5.1 Tampilan Hasil Diagnosis 68](file:///C:\Users\Kevin\PycharmProjects\TugasAkhir\Buku%20TA\Buku%20TA\BukuTA1.6.docx#_Toc518889326)

[Gambar 5.2 Pengisian Data Pasien 68](file:///C:\Users\Kevin\PycharmProjects\TugasAkhir\Buku%20TA\Buku%20TA\BukuTA1.6.docx#_Toc518889327)

[Gambar 5.3 Tampilan Tanya Jawab 68](file:///C:\Users\Kevin\PycharmProjects\TugasAkhir\Buku%20TA\Buku%20TA\BukuTA1.6.docx#_Toc518889328)

[Gambar 5.4 Grafik Hasil Uji Akurasi 70](#_Toc518889329)

[Gambar 5.5 Grafik Hasil Uji Retrieval 71](#_Toc518889330)

[Gambar 5.6 Grafik Hasil Uji Peringkat 72](#_Toc518889331)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR TABEL

[Tabel 3.1 Contoh Hasil Pembangunan Dataset 21](#_Toc518907037)

[Tabel 3.2 Sampel Dataset dari Database 23](#_Toc518907038)

[Tabel 3.3 Hasil Pengolahan Sampel Data 23](#_Toc518907039)

[Tabel 3.4 *Support* Atribut Data Sampel 24](#_Toc518907040)

[Tabel 3.5 *Itemset* Yang Telah Diurutkan dan Filter 24](#_Toc518907041)

[Tabel 3.6 Hasil *Conditional Pattern* 25](#_Toc518907042)

[Tabel 3.7 Hasil *Frequent Pattern* 26](#_Toc518907043)

[Tabel 3.8 Contoh Ekstraksi *Rules* Data Sampel 27](#_Toc518907044)

[Tabel 3.9 Hasil *Rules* Data Sampel 27](#_Toc518907045)

[Tabel 3.10 Lanjutan Tabel 3.9 28](#_Toc518907046)

[Tabel 3.11 Lanjutan Tabel 3.10 29](#_Toc518907047)

[Tabel 3.12 Lanjutan Tabel 3.11 30](#_Toc518907048)

[Tabel 4.1 Lingkungan Implementasi Perangkat Lunak 37](#_Toc518907049)

[Tabel 5.1 Lingkungan Pengujian *Server* 65](#_Toc518907050)

[Tabel 5.2 Lingkungan Pengujian *Client* (Pengguna) 66](#_Toc518907051)

[Tabel 5.3 Jumlah Data Gejala dan Rules 66](#_Toc518907052)

[Tabel 5.4 Hasil Uji Akurasi 69](#_Toc518907053)

[Tabel 5.5 Hasil Uji Retrieval 70](#_Toc518907054)

[Tabel 5.6 Hasil Uji Peringkat 72](#_Toc518907055)

[Tabel 5.7 Hasil Uji Pengguna Dengan Pertanyaan Maksimal 30 73](#_Toc518907056)

[Tabel 5.8 Hasil Uji Pengguna Dengan Pertanyaan Maksimal 20 73](#_Toc518907057)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR KODE SUMBER

[Kode Sumber 4.1 Filter pada *FP Tree* 38](#_Toc518512496)

[Kode Sumber 4.2 Pembuatan *Tree* Pada *FP Tree* 39](#_Toc518512497)

[Kode Sumber 4.3 Ekstrak *Frequent Pattern* 40](#_Toc518512498)

[Kode Sumber 4.4 Lanjutan Kode Sumber 4.3 41](#_Toc518512499)

[Kode Sumber 4.5 Lanjutan Kode Sumber 4.4 42](#_Toc518512500)

[Kode Sumber 4.6 Ekstrak Rules pada *FP Tree* 43](#_Toc518512501)

[Kode Sumber 4.7 Pembuatan Model *Naive Bayes* 44](#_Toc518512502)

[Kode Sumber 4.8 Pembuatan Model SVM 45](#_Toc518512503)

[Kode Sumber 4.9 Evaluasi Akurasi 46](#_Toc518512504)

[Kode Sumber 4.10 Evaluasi *Information Retrieval* 47](#_Toc518512505)

[Kode Sumber 4.11 Lanjutan Kode Sumber 4.10 48](#_Toc518512506)

[Kode Sumber 4.12 Filter Gejala pada Tanya Jawab Aplikasi 49](#_Toc518512507)

[Kode Sumber 4.13 Cek Gejala Khusus pada Tanya Jawab Aplikasi 49](#_Toc518512508)

[Kode Sumber 4.14 Lanjutan Sumber Kode 4.13 50](#_Toc518512509)

[Kode Sumber 4.15 *Query Rules* pada Tanya Jawab Aplikai 51](#_Toc518512510)

[Kode Sumber 4.16 Cek Hasil Query 52](#_Toc518512511)

[Kode Sumber 4.17 Menghilangkan Gejala Terakhir pada Tanya Jawab Aplikai 53](#_Toc518512512)

[Kode Sumber 4.18 Menjawab Melalui Data Pasien pada Tanya Jawab Aplikai 53](#_Toc518512513)

[Kode Sumber 4.19 Implementasi Prediksi pada Server 54](#_Toc518512514)

[Kode Sumber 4.20 Lanjutan Kode Sumber 4.19 56](#_Toc518512515)

[Kode Sumber 4.21 Lanjutan Kode Sumber 4.20 56](#_Toc518512516)

[Kode Sumber 4.22 Implementasi HTML pada Aplikasi Pengguna 57](#_Toc518512517)

[Kode Sumber 4.23 Lanjutan Kode Sumber 4.22 58](#_Toc518512518)

[Kode Sumber 4.24 Lanjutan Kode Sumber 4.23 59](#_Toc518512519)

[Kode Sumber 4.25 Lanjutan Kode Sumber 4.24 60](#_Toc518512520)

[Kode Sumber 4.26 Javascript Tanya Jawab Pada Interface User 60](#_Toc518512521)

[Kode Sumber 4.27 Lanjutan Kode Sumber 4.26 61](#_Toc518512522)

[Kode Sumber 4.28 Lanjutan Kode Sumber 2.27 62](#_Toc518512523)

[Kode Sumber 4.29 Implementasi *Javascript* Prediksi Diagnosis pada *user* 63](#_Toc518512524)

[Kode Sumber 4.30 Lanjutan Kode Sumber 4.29 64](#_Toc518512525)

# BAB I PENDAHULUAN

Bab ini akan membahas latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

## Latar Belakang

Kemajuan teknologi membuat informasi semakin mudah diakses. Masyarakat kini lebih haus informasi dibanding dengan generasi sebelumnya. Tidak terkecuali dalam bidang kesehatan. Masyarakat kini dapat membaca artikel kesehatan dengan mudah. Mendapatkan obat dengan mudah. Obat obatan kini bukan lagi sebuah rahasia dokter dan apoteker. Masyarakat kini cenderung mengetahui jenis jenis obat dan apa guna obat tersebut.

Sakit dapat terjadi pada siapapun. Mulai dari penyakit ringan hingga penyakit serius. Menurut statistik dari *UK Digital Health Report*, 1 dari 5 orang lebih memilih untuk melakukan diagnosis sendiri dengan bantuan *search engine* [1]. Hal ini tentu saja meningkatkan resiko kesalahan diagnosis pada pasien, dan dapat menyebabkan penyakit pasien semakin memburuk bahkan meninggal dunia.

Untuk mengatasi hal tersebut, diperlukan suatu solusi yang dapat memperkecil kesalahan masyarakat dalam mendiagnosis penyakit. Sehingga pasien lebih waspada dan tidak menganggap remeh gejala yang mereka alami. Solusi tersebut dapat dikemas dalam bentuk aplikasi yang didukung oleh kecerdasan buatan yang dapat mendiagnosis berdasarkan gejala gejala yang diberikan oleh pasien.

Dalam tugas akhir ini, akan digunakan *FP Tree* untuk dapat memberikan gejala gejala berkaitan dengan gejala utama pasien dan dapat memberikan diagnosis yang sesuai.

## Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut :

1. Bagaimana cara menghasilkan pertanyaan gejala yang sesuai dengan jawaban pertanyaan gejala sebelumnya menggunakan *Frequent Pattern Tree Growth*?
2. Bagaimana menentukan diagnosis yang cocok berdasarkan gejala gejala pasien?
3. Bagaimana pengaruh *minimum support* terhadap performa *classifier*?

## Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Jumlah diagnosis sebanyak 66 diagnosis.
2. Implementasi dilakukan pada lingkungan kerja berbasis *web*.

## Tujuan Pembuatan Tugas Akhir

Tujuan pembuatan tugas akhir ini adalah membangun aplikasi yang dapat membantu dokter dalam melakukan anamnesis dan memberikan diagnosis berdasarkan gejala gejala yang diberikan oleh pengguna menggunakan *Frequent Pattern Tree Growth*.

## Manfaat Tugas Akhir

Tugas akhir ini akan berguna untuk memberikan perkiraan penyakit kepada masyarakat awam, serta memberikan masukan diagnosis kepada dokter. Input data gejala dan kondisi pasien dapat dibaca oleh dokter yang ditunjuk oleh pasien sebagai pemeriksa dirinya. Sehingga akan membantu proses pemeriksaan dokter terhadap pasien.

## Metodologi

Metodologi yang akan digunakan pada tugas akhir ini adalah:

* 1. Penyusunan Proposal Tugas Akhir

Proposal akan berisi tentang pendahuluan, latar belakang, tujuan, manfaat, dan rumusan masalah. Yang akan didukung dengan penjelasan berupa tinjauan pustaka dan juga metode serta langkah langkah yang akan dilakukan untuk membangun produk.

* 1. Studi Literatur dan Wawancara

Sebelum pembuatan aplikasi, akan dilakukan studi literatur terkait aplikasi, dan juga melakukan wawancara kepada ahli, yang dalam tugas akhir ini adalah dokter untuk menyesuaikan metode yang dilakukan ahli untuk memberikan output yang diinginkan (diagnosis).

* 1. Analisis dan Desain

Pada tahap ini akan dilakukan analisis terkait bagaimana membangun model *Frequent Pattern Tree Growth* yang baik dan juga arsitektur terkait. Serta akan di analisa juga *dataset* yang sesuai untuk digunakan pada model.

* 1. Implementasi

Model yang telah dihasilkan pada langkah sebelumnya akan di implementasikan pada *server* yang berisi model, dan juga aplikasi untuk klien berupa *web*.

* 1. Evaluasi

Aplikasi yang telah dibuat akan dievaluasi oleh dokter dan juga pengguna untuk menilai kesesuaian anamnesis yang dilakukan oleh aplikasi dan kepuasan pengguna terhadap aplikasi. Selain itu aplikasi juga akan di evaluasi secara sistem menggunakan akurasi dan *retrieval*.

* 1. Penyusunan Buku Tugas Akhir

Pada tahap ini akan dilakukan penyusunan laporan berupa buku tugas akhir yang menjelaskan dasar teori pada tugas akhir ini serta hasil implementasi pada tugas akhir.

## Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I. Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.

1. Bab II Tinjauan Pustaka

Bab ini berisi penjelasan tentang metode, algoritma,dan *library* yang digunakan pada pembuatan aplikasi ini. Pembahasan akan berfokus pada algoritma *associative rules* terutama pada *Frequent Pattern Tree Growth* dan juga metode klasifikasi yang digunakan yaitu *Naïve Bayes* dan *Support Vector Machine*.

1. Bab III Perancangan Perangkat Lunak

Bab ini aka membahas mengenai perancangan, desain, model, dan proses proses lain yang digunakan untuk pembuatan aplikasi juga model *Frequent Pattern Tree Growth* serta model *Classifier*.

1. Bab IV. Implementasi

Bab ini akan menjelaskan proses pembuatan aplikasi dengan bahasa *Python* yang dibantu oleh *library* *Flask* dan juga *PHP* dengan *framework laravel.* Juga akan membahas pembuatan model menggunakan bahasa *Python* yang didukung dengan *ScikitLearn*.

1. Bab V. Hasil Uji Coba dan Evaluasi

Bab ini akan menjelaskan hasil percobaan yang dilakukan secara sistem dengan *data testing* yang telah diambil dari sumber buku dan web, juga hasil percobaan yang dilakukan oleh dokter.

1. Bab VI. Kesimpulan dan Saran

Bab ini akan membahas kesimpulan yang didapatkan dari hasil ujicoba dan juga akan membahas saran saran yang didapatkan baik dari penulis juga dari dokter yang melakukan uji coba. Sehingga aplikasi dapat dikembangkan menjadi lebih baik

***[Halaman ini sengaja dikosongkan]***

# BAB II DASAR TEORI

## Anamnesis

Pada sub bab ini akan dijelaskan pengertian anamnesis dan bagaimana proses anamnesis yang dilakukan oleh dokter.

### Pengertian Anamnesis

Proses akumulasi data yang menyangkut data medis pasien, latar belakang pasien, termasuk keluarga, lingkungan, pengalaman, terutama ingatan untuk digunakan dalam menganalisa kondisi [2].

### Langkah Langkah Anamnesis

Umumnya anamnesis dilakukan sesuai dengan cara cara berikut [3]:

1. Pasien memberikan gejala

Pasien akan memberikan gejala yang dialami, misal : nyeri dada, badan panas.

1. Mendapatkan informasi lebih dalam mengenai gejala

Dokter akan menanyakan lebih lanjut terkait gejala yang dialami pasien. Misal pada nyeri dada, dokter akan menanyakan pertanyaan berikut kepada pasien.

* Dimana tepatnya letak nyeri pada dada?
* Sejak kapan nyeri dada dirasakan?
* Apakah nyeri dada sering sekali muncul, terkadang muncul, atau jarang muncul?
* Apakah rasa nyeri berpindah?
* Jika nyeri dada kambuh, berapa lama biasanya nyeri dada tersebut terasa?
* Apakah nyerinya semakin sakit atau semakin tidak sakit?
* Dalam skala 1-10, seberapa sakit yang anda rasakan?

1. Mencari gejala lain yang dialami pasien

Dokter akan menanyakan apakah pasien mengalami gejala lain yang mungkin berkaitan dengan gejala sebelumnya. Misal : Nyeri dada, maka mungkin pasien akan mengalami sulit bernapas.

1. Menanyakan tindakan/obat yang sudah dilakukan terhadap gejala tersebut

Dokter akan menanyakan tindakan atau obat apa yang telah dikonsumsi pasien. Lebih lanjut dokter akan menanyakan terkait dosis, nama obat, dan seberapa sering pasien mengonsumsi obat atau melakukan tindakan tersebut.

1. Menanyakan informasi kesehatan keluarga

Dokter akan menanyakan informasi keluarga yang mungkin memiliki penyakit yang berkaitan yang bersifat genetik. Misal : diabetes

1. Menanyakan informasi lingkungan keseharian

Dokter akan menanyakan bagaimana keseharian pasien, apakah pasien merokok, atau apakah pasien menggunakan obat obatan terlarang.

1. Menanyakan informasi lain terkait sistem tubuh lain yang tidak tercakup pada gejala

Dokter akan menanyakan apakah ada sistem tubuh lain yang terganggu. Umumnya sistem yang akan ditanyakan dokter adalah sebagai berikut :

* Kardiovaskular
* Pernapasan
* Pencernaan
* Saraf
* Genital
* Muskuloskeletal
* Kejiwaan

1. Mengulas ulang keluhan yang diberikan pasien

Dokter akan mengulas poin poin penting yang diberikan pasien sebelum memberikan diagnosis.

1. Dokter memberikan diagnosis

Pada sub bab ini akan dijelaskan apa pengertian *Frequent Pattern Tree Growth,* istilah istilah pada algoritma tersebut dan juga algoritmanya.

## *Frequent Pattern Tree Growth*

Pada sub bab ini akan dijelaskan mengenai pengertian hingga algoritma tentang *Frequent Pattern Tree Growth*.

### 2.2.1 Pengertian *Frequent Pattern Tree Growth*

*Frequent Pattern Tree Growth* atau *FP Tree* merupakan salah satu algoritma *associative rules* yang sering digunakan pada berbagai permasalahan *data mining*. Algoritma ini sendiri bertujuan membuat *rules* yang didasarkan pada *tree* yang dibuat berdasarkan dataset yang diberikan.

Seperti *tree* pada umumnya, *tree* pada *FP Tree* juga memiliki *root*, *node* dan juga *leaf*. Pada *FP Tree* penempatan node akan didasarkan pada *support* pada setiap attribut pada sebuah data. Sehingga jika dilihat semakin tinggi posisi dari suatu *node* maka dapat dipastikan *node* tersebut memiliki *support* yang lebih tinggi daripada *child*-nya.

### 2.2.2 *Support*

*Support* adalah batas minimal indikasi seberapa sering *item* akan muncul pada *dataset*. [4]

*support(A→B) = P (A∪ B)* (2.1)

**Keterangan** :

A dan B adalah atribut

### 2.2.3 Algoritma *FP Tree*

Untuk mendapatkan rules dari *dataset*. *FP Tree* memiliki algoritma sebagai berikut:

*Input* : *Dataset*, dan *minimum support*

*Output* : *Rules*

Tahap 1. Pembuatan *Tree* [5]

1. *Scan* *database*, dan mengumpulkan kumpulan *frequent items*, dan *minimum support* untuk setiap *frequent items*. Urutkan data tersebut sesuai dengan nilai *support* secara *descending*.
2. Buat *root* dari *tree*
3. Pilih salah satu *frequent item* dan buat *node* untuk setiap *item*. Lanjutkan hingga *item* dari *set* tersebut habis.
4. Jika *node* telah terbuat untuk *item* tertentu, maka atribut jumlah akan ditambahkan sesuai dengan frekuensi dia muncul pada *node* tersebut.
5. Ulangi langkah 3 dan 4 hingga *tree* terbuat.

Tahap 2. Ekstrak *Frequent Pattern* [6]

1. Dari *tree* ambil salah satu *node* yang terletak paling akhir/bawah dari *tree*.
2. Lalu untuk ekstrak pertama maka akan mengambil *node* paling bawah.
3. Dari *node* terpilih akan di *traverse* dari *node* terpilih hingga *root* dan akan mendapatkan sebuah *path*.
4. *Path* yang dihasilkan akan dicatat. Jika *node* terpilih memiliki *support* > 1 maka akan ditulis sebanyak jumlah *support*.
5. Lakukan perhitungan *support* atribut pada *path* yang dihasilkan.
6. Setelah itu filter setiap atribut.
7. Dari atribut yang telah di filter akan dilakukan *divide and conquer*. Sehingga akan memiliki kombinasi *itemset* untuk atribut tersebut. *Support* dihitung berdasarkan banyaknya *itemset* yang muncul pada daftar *path* yang terbentuk.
8. Hilangkan *node* terpilih. Lalu naik ke *node* diatasnya.
9. Ulangi langkah 3.

Tahap 3. Ekstrak *Rules* [7]

1. Dari setiap *frequent pattern item* kita lakukan *divide and conquer* sehingga akan memunculkan banyak *itemset*. *Itemset* hasil *divide and conquer* disebut dengan anteseden, dan item yang tidak muncul pada *itemset* disebut konsekuen.
2. Lalu untuk setiap anteseden maka konsekuen akan digunakan rumus perhitungan *support*. S*upport* dari *itemset* akan dibagi dengan *support* dari anteseden.
3. Ulangi langkah pertama.

## Naïve Bayes

Naïve Bayes adalah salah satu algoritma *supervised* pada data mining. Naïve Bayes adalah sebuah *classifier* berbasis probabilitas yang sederhana yang menghitung frekuensi dan kombinasi nilai pada *dataset* [8]. Algoritma *Naïve Bayes* berbasiskan pada *Bayes Theorem*. Dengan persamaan sebagai berikut [9] :

(2.2)

**Keterangan**

A dan B adalah atribut

P(A) = Peluang A

P(B) = Peluang B

P (A | B) = Peluang A terhadap B

P (B | A) = Peluang B terhadap A

*Naïve Bayes* memiliki asumsi yang mendasar yaitu setiap fitur bersifat independen dan sama. Bersifat independen berarti setiap fitur tidak ada hubungannya oleh fitur lainnya. Misalnya hujan turun tidak ada hubungannya dengan angin yang bertiup kencang. Kedua fitur memiliki sifat sama artinya memiliki kontribusi atau *weight* yang sama. Asumsi yang dimiliki oleh Naïve Bayes pada umumnya tidak benar pada situasi dunia nyata [10].

### Jenis *Naïve Bayes*

*Naive Bayes* memiliki jenis yang berbeda beda tergantung pada jenis data yang digunakan. Perbedaan jenis *Naïve Bayes* sangat berpengaruh terhadap performa yang dihasilkan *classifer.*

#### *Gaussian* *Naïve Bayes*

*Gaussian* *Naïve Bayes* digunakan saat attribut memiliki data yang bersifat kontinu, diasumsikan bahwa setiap nilai attribut memiliki hubungan dengan setiap kelas yang terdistribusi menurut *Gaussian Normal Distribution* [11]*.*

#### *Multinomial* *Naïve Bayes*

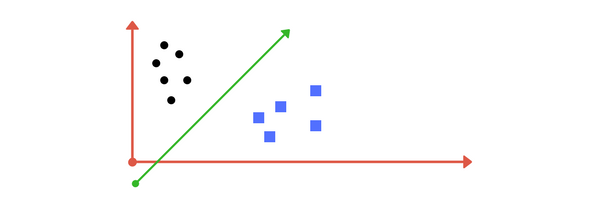
*Multinomial Naïve Bayes* lebih cocok jika digunakan pada data yang secara terdistirbusi secara multinomial. Biasanya digunakan untuk klasifikasi teks untuk menghitung frekuensi kata yang muncul pada suatu dokumen [11].

#### *Bernoulli Naïve Bayes*

*Bernoulli Naïve Bayes* digunakan saat data memiliki nilai *binary.* Seperti *Multinomial*, *Bernoulli* cocok digunakan untuk klasifikasi teks yang digunakan untuk menandakan apakah kata tersebut ada pada dokumen atau tidak [10].

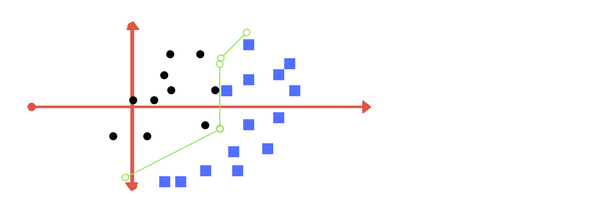
## *Support Vector Machine*

*Support Vector Machine* atau *SVM* adalah sebuah *classifier* yang menggolongkan dengan cara membagi data menjadi area yang terpisah dengan *hyperplane* [12]. Dengan adanya *hyperplane* tersebut data data yang berbeda dapat dikategorikan menjadi 2 data berbeda.



Gambar 2.1 Algoritma *SVM*

### *Hyperplane*



Gambar 2.2 Contoh *Hyperplane*

*Hyperplane* adalah pemisah bidang pada *SVM*. Umumnya *hyperplane* pada *SVM* adalah sebuah garis. Tetapi garis pemisah tidak selalu garis lurus. Garis pemisah dapat berupa persamaan kuadrat atau garis garis lainnya.

### *Kernel* *SVM*

Pemilihan *kernel* penting dalam memecahkan masalah *SVM*. Perbedaan antar kernel dapat memberikan perbedaan performa yang cukup siginifikan. Pada *SVM* Linier permasalahan dipecahkan menggunakan persamaan aljabar linier. Ada beberapa kernel *SVM* lain yang bisa digunakan, seperti kernel Polinomial, *RBF* dan *Gaussian*. *Kernel* Polinomial umumnya digunakan untuk pengolahan citra digital sedangkan *RBF* dan *Gaussian* bisa digunakan untuk data umum dan kita memiliki pengetahuan yang sedikit tentang data tersebut [13]. Performa *kernel* bergantung kepada data yang digunakan. Jika kita memiliki fitur data yang banyak sedangkan data sedikit, maka *kernel* linier lebih cocok digunakan [14].

### *SVM Multiclass*

*SVM* memiliki performa yang baik saat memisahkan antar 2 data atau data yang memiliki 2 kelas saja. Tetapi *SVM* juga dapat digunakan untuk memecahkan masalah yang datanya memiliki banyak kelas. Ada 2 pendekatan yang digunakan yaitu *One vs All* dan *One vs One.*

#### *One vs All*

Pada pendekatan ini data akan di training satu per satu. Setiap kelas yang di training akan dilabeli sebagai kelas yang bernilai positif sedangkan kelas lain bernilai negatif. Walaupun secara komputasi pendekatan ini lebih cepat tetapi jika data tidak seimbang, dapat menyebabkan performa yang kurang baik [15].

#### *One vs One*

Pada pendekatan ini setiap kelas akan saling di training satu sama lain sehingga akan terbuat sebanyak N(N-1)/2 *classifier*. Pendekatan ini menghabiskan lebih banyak proses komputasi tetapi baik dalam menangani data yang tidak seimbang [15].

## 2.5 *Scikit Learn*

*Scikit Learn* adalah *library* *machine learning* gratis berbasiskan bahasa pemrograman *Python*. *Library* ini memiliki banyak fitur mulai dari klasifikasi, regresi, kluster termasuk *SVM, Random Forest, Gradient Boosting, KMeans, DBScan* dan didesain untuk bekerja dengan *library* numerik dan ilmiah milik Python, *NumPy* dan *SciPy* [16].

## 2.6 *Python – Flask*

*Flask* adalah sebuah *microframework* untuk python [17]. *Flask* umumnya digunakan untuk membuat web. *Flask* tidak memiliki *database abstraction layer, form validation,* dan banyak fitur lainnya yang mana *library* lain sudah ada dan dapat mengatasi masalah tersebut. *Flask* mungkin *library* mikro namun mampu untuk digunakan dalam berbagai kebutuhan [18].

## 2.7 *PHP - Laravel*

*Laravel* adalah sebuah *framework* untuk *web* dengan *syntax* yang ekspresif dan elegan. *Laravel* mudah diakses, kuat, dan memiliki *tools* untuk aplikasi besar dan kuat. [19]. Berbeda dengan *flask*, *laravel* berbasiskan bahasa pemrograman PHP.

***[Halaman ini sengaja dikosongka***

# BAB III PERANCANGAN

## 3.1 Perancangan Data

Pada anamnesis, dokter melakukan pertanyaan kepada pasien mengenai gejala yang paling dirasakan. Lalu dokter akan menanyakan gejala berikutnya berdasarkan gejala tersebut. Dokter menanyakan gejala berikutnya berdasarkan kecenderungan antar gejala, bahwa jika seseorang memiliki gejala X maka orang tersebut juga cenderung untuk memiliki gejala Y, sehingga orang tesebut akan memiliki diagnosis Z. Sehingga berdasarkan hal tersebut dapat disimpulkan bahwa gejala memiliki hubungan dengan gejala lain jika gejala – gejala tersebut terdapat pada minimal satu diagnosis yang sama.

Dari analogi tersebut maka algoritma *Association Rules* cocok digunakan pada permasalahan ini. Dan algoritma *FP Tree* digunakan untuk memecahkan masalah tersebut. Untuk menjalankan *FP Tree* data yang akan digunakan pada aplikasi ini berasal dari berbagai sumber buku dan *website*. Penulis mengekstrak informasi data dari buku dan *website* dan memasukkannya pada *database*.

Data ini terdiri dari 858 data kasus, yang memiliki 66 diagnosis atau kelas. Serta 375 atribut. Data ini akan dibagi menjadi 2 kelompok yaitu data belajar dan data pengujian. Data belajar terdiri dari 10 kasus untuk setiap kelas. Dan data pengujian terdiri dari 3 kasus untuk setiap kelas.

Berikut adalah diagnosis yang terdapat pada dataset yang digunakan pada Tugas Akhir ini :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. | Rhinitis |  | 34. | Apendisitis |
| 2. | Influenza |  | 35. | Peritonitis |
| 3. | Faringitis |  | 36. | Askariasis |
| 4. | Tonsilitis |  | 37. | Ankilostomiasis |
| 5. | Laringitis |  | 38. | Skistosomiasis |
| 6. | Pneumonia |  | 39. | Taeniasis |
| 7. | Bronkopneumonia |  | 40. | Strongiloidiasis |
| 8. | Pneumotoraks |  | 41. | Mata Kering |
| 9. | PPOK |  | 42. | Buta Senja |
| 10. | Epistaksis |  | 43. | Hordeolum |
| 11. | Sinusitis |  | 44. | Konjungtivitis |
| 12. | Tuberkulosis |  | 45. | Blefaritis |
| 13. | Morbili |  | 46. | perdarahan subkonjungtiva |
| 14. | Varisela |  | 47. | Astigmatisme |
| 15. | Malaria |  | 48. | Hipermetropia |
| 16. | Leptospirosis |  | 49. | Miopia Ringan |
| 17. | Filariasis |  | 50. | Presbiopia |
| 18. | Lepra |  | 51. | Katarak |
| 19. | Keracunan Makanan |  | 52. | Glaukoma |
| 20. | Alergi Makanan |  | 53. | Trikiasis |
| 21. | DBD |  | 54. | Episkleritis |
| 22. | Anemia |  | 55. | Hifema |
| 23. | HIV/AIDS |  | 56. | Retinopati Diabetik |
| 24. | Lupus Eritematosus |  | 57. | Otitis Eksterna |
| 25. | Limfadenitis |  | 58. | Otitis Media |
| 26. | Asam Lambung |  | 59. | Benda Asing di Telinga |
| 27. | Gastritis |  | 60. | Serumen Prop |
| 28. | Gastroenteritis |  | 61. | Angina Pektoris Stabil |
| 29. | Disentri |  | 62. | Infark Miokard |
| 30. | Hemoroid |  | 63. | Takikardia |
| 31. | Hepatitis A |  | 64. | Gagal Jantung |
| 32. | Hepatitis B |  | 65. | Cardiorespiratory Arrest |
| 33. | Kolesistitis |  | 66. | Hipertensi |

## 3.2 Desain Sistem Secara Umum

Tugas Akhir akan dibuat dalam beberapa tahap sesuai dengan

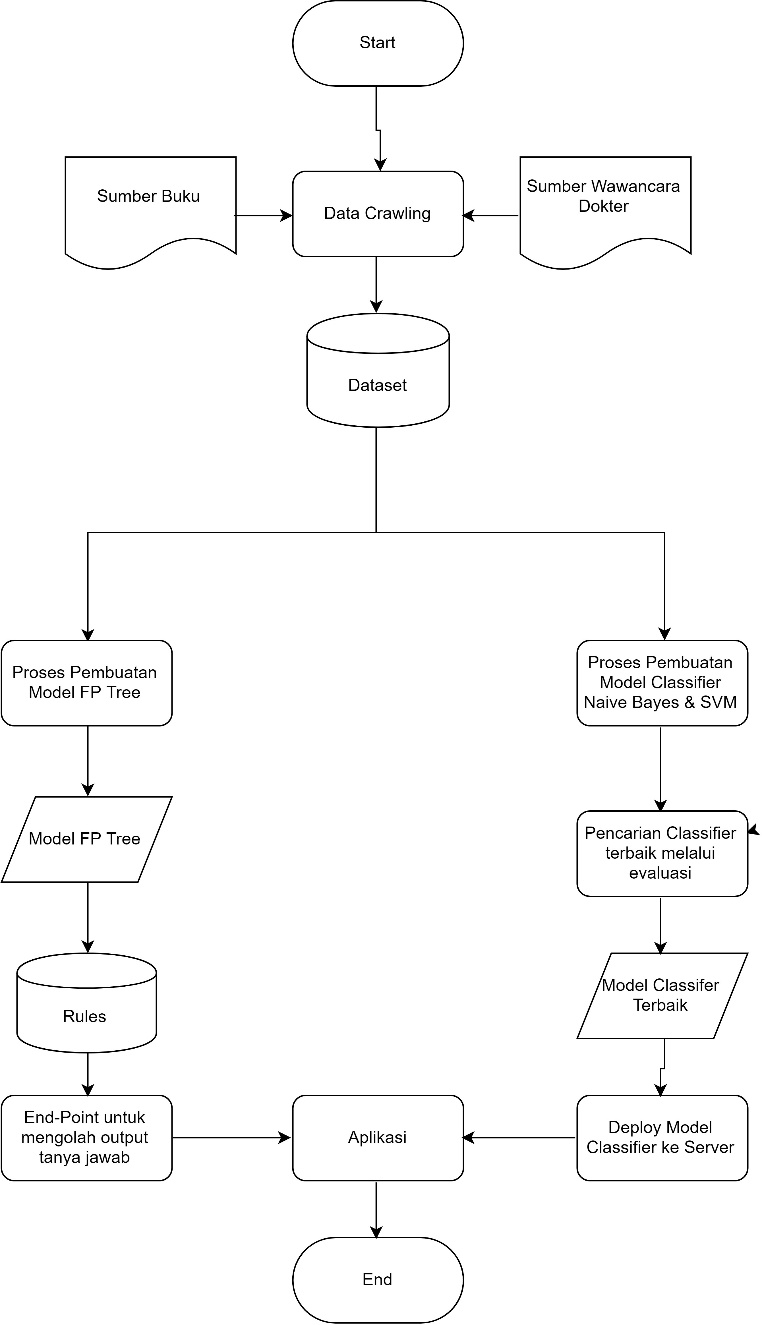
Gambar 3.1 Diagram Sistem Secara Umum. Tahap tahap tersebut adalah pembangunan dataset, pembuatan model FP Tree, tahap pembuatan model *classifier*, dan tahap pembuatan aplikasi untuk pengguna. Tahap pertama adalah pembangunan data, data data dari *database* akan diubah formatnya sehingga dapat dibaca dengan cepat oleh model *FP Tree* dan *classifier*. Pada tahap pembuatan model *FP Tree*, data akan difilter dengan *minimum support* sesuai dengan skenario. Lalu data yang telah di *filter* sesuai *minimum support* tersebut akan di training untuk membuat *rules*. *Rules* yang dihasilkan oleh *FP Tree* akan dimasukkan kedalam *database* yang nantinya akan digunakan oleh *server* untuk menentukan pertanyaan yang akan ditanyakan kepada pengguna.

Pada tahap pembuatan model *classifier,* dataset akan di filter berdasarkan *minimum support*, yang mana akan di *training* dengan *Naïve Bayes* dan juga *SVM.* Lalu masing masing hasil model akan di ujikan dengan data pengujian. Hasil dari pengujian ini adalah *classifer* dengan hasil yang terbaik, yang mana model ini yang akan digunakan untuk server *machine learning* yang akan menentukan diagnosis penyakit.

Tahap pembuatan aplikasi *user* akan menggabungkan antara hasil *rules* dengan model *classifier*. Aplikasi ini akan dibuat dengan *backend* *Laravel* dan *front-end Javascript Vanilla*.

### 3.2.1 Pembangunan Dataset

Data yang berasal dari buku (PPK Kedokteran, Diagnosis Banding Kedokteran, dan Kapita Selekta Kedokteran) dan wawancara dengan dokter akan dimasukkan ke dalam *database* yang memiliki tabel kasus, kasus\_gejala, dan gejala. Dimana kasus akan menyimpan diagnosis, dan kasus\_gejala akan menyimpan daftar gejala berdasarkan diagnosis pada kasus tersebut. Dari tabel tabel tersebut akan dibuat *dataset* yang terdapat pada 1 tabel yang sama agar bisa digunakan pada model.



Gambar 3.1 Diagram Sistem Secara Umum

Sehingga akan dibuat tabel dengan kolom berisi id dari gejala, dan baris yang berisi id kasus. Sehingga akan terbentu suatu data dengan dimensi jumlah kasus X jumlah gejala. Dan untuk setiap baris atau kasus, jika kasus memiliki suatu gejala maka akan dicari id gejala tersebut pada kolom dan akan diberi angka 1 untuk indeks baris dan id gejala.

Tabel 3.1 Contoh Hasil Pembangunan Dataset

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID Kasus | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 |
| 5 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

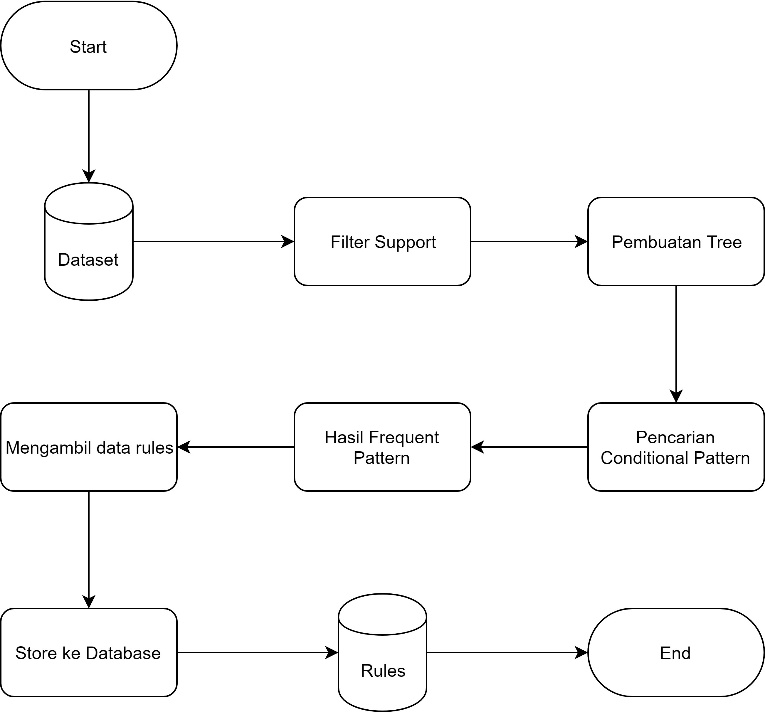
### 3.2.2 Perancangan Model *FP Tree*

*Dataset* akan diolah dengan algoritma *FP Tree* sesuai dengan Gambar 3.2 Diagram Algoritma FP Tree. Algoritma akan diimplementasikan dengan bahasa pemrograman *Python*.

Pada tahap *filter* atribut pada masing masing data akan di *filter* berdasarkan *minimum support*. Adapun *minimum support* yang akan diskenariokan akan dijelaskan pada bab berikutnya.

Setelah di *filter*, data akan diurutkan berdasarkan urutan *support* dari yang terbesar hingga terkecil. Dan urutan tersebut akan menunjuk urutan sebelumnya yang lebih besar sehingga atribut yang memiliki *support* terbesar akan terletak pada *layer* teratas dan *support* terkecil akan terletak pada *layer* terbawah. Jika ada data yang memiliki atribut sama pada sebuah *layer* yang sama dengan *parent* yang sama maka *tree* tidak perlu membuat *node* baru, melainkan hanya menambahkan atribut frekuensi pada *node* yang sama tersebut.

Setelah tree terbentuk, maka akan dicari *Conditional Pattern* dari setiap *node*. Pencarian ini akan mencari *path* dari *node* tersebut hingga ke *root*. Dari proses ini akan memunculkan *paths* yang dilalui oleh *node* yang memiliki atribut yang sama.



Gambar 3.2 Diagram Algoritma FP Tree

Dari *Conditional Pattern,* kita dapat mengekstrak *Frequent Pattern.* *Frequent Pattern* diambil dari kombinasi *Conditional Pattern* yang memiliki *support* lebih dari *minimum support*.

Dari hasil *Frequent Pattern*, akan diekstrak kombinasi yang akan memunculkan *rules*. *Rules* tersebut akan dihitung *confidence* dengan cara membagi hasil kombinasi dengan *pattern* utama. *Confidence* tidak akan di *filter* mengingat jumlah *rules* yang tidak begitu banyak.

*Rules* tersebut akan disimpan pada sebuah tabel pada *database* yang memiliki atribut *id, rules input, rules output* dan *probability*. Sehingga nantinya pada akhir proses akan memunculkan 9 tabel sesuai dengan jumlah skenario.

#### 3.2.2.1 Contoh Proses Model *FP Tree*

Berdasarkan penjelasan pada Bab 3.2.2 Perancangan Model *FP Tree* akan dijelaskan contoh penggunaan algoritma hingga menghasilkan *rules*. Pada Tabel 3.2 diberikan sampel *dataset* dari *database*, dan pada Tabel 3.3 adalah hasil pengolahan data sampel yang akan diproses pada model *FP Tree*. Pada contoh kali ini digunakan *minimum support* sebanyak 2, karena terbatasnya jumlah sampel data.

Tabel 3.2 Sampel Dataset dari Database

|  |
| --- |
| **Itemset** |
| 182, 163, 282, 281, 283 |
| 182, 283, 281, 282 |
| 279, 282, 281, 284 |
| 182, 279, 283, 282 |

Tabel 3.3 Hasil Pengolahan Sampel Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **163** | **182** | **279** | **281** | **282** | **283** | **284** |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |

Proses Pembangunan Model *FP Tree*

1. Filter *Support*

Hitung *support* untuk setiap *itemset* dengan hasil Tabel 3.4. Lalu urutkan dan filter itemset berdasarkan support Tabel 3.5.

1. Pembuatan *Tree*

Dari setiap *itemset* yang telah diurutkan akan dibuat *tree* yang memiliki atribut id gejala dan juga *support*, dengan hasil yang dapat dilihat pada Gambar 3.3.

Tabel 3.4 *Support* Atribut Data Sampel

|  |  |
| --- | --- |
| **Atribut** | ***Support*** |
| 163 | 1 |
| 182 | 3 |
| 279 | 2 |
| 281 | 3 |
| 282 | 4 |
| 283 | 3 |
| 284 | 1 |

Tabel 3.5 *Itemset* Yang Telah Diurutkan dan Filter

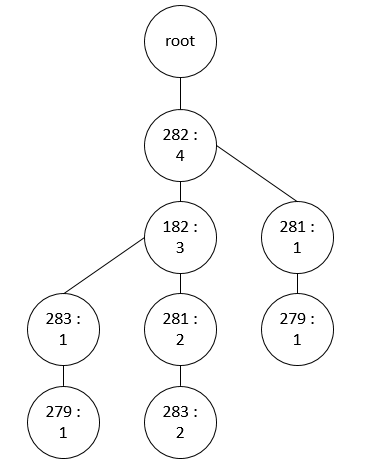
|  |
| --- |
| ***Itemset*** |
| 282, 182, 281, 283 |
| 282, 182, 281, 283 |
| 282, 281, 279 |
| 282, 182, 283, 279 |

1. *Conditional Pattern*

Dari *tree* pada Gambar 3.3 akan diambil *node* *node* yang terletak paling bawah dan akan di ekstrak *conditional* *pattern* dengan hasil pada Tabel 3.6.

1. *Frequent Pattern*

Dari *conditional pattern* akan diekstrak *frequent pattern* menggunakan *divide and conquer* sehingga menghasilkan *frequent pattern*. Pada contoh kali ini akan diambil *conditional pattern* yang memilliki *prefix* 283 sehingga menghasilkan *frequent pattern* yang dapat dilihat pada Tabel 3.7.



Gambar 3.3 Hasil *Tree* Data Sampel

Tabel 3.6 Hasil *Conditional Pattern*

|  |  |
| --- | --- |
| **Prefix** | ***Conditional Pattern Itemset*** |
| 279 | 282, 182, 283, 279 |
| 279 | 282, 281, 279 |
| 283 | 282, 182, 281, 283 |
| 283 | 282, 182, 281, 283 |
| 283 | 282, 182, 283 |
| 281 | 282, 182, 281 |
| 281 | 282, 182, 281 |
| 281 | 282, 281 |
| 182 | 282, 182 |
| 182 | 282, 182 |
| 182 | 282, 182 |
| 282 | 282 |
| 282 | 282 |
| 282 | 282 |
| 282 | 282 |

Tabel 3.7 Hasil *Frequent Pattern*

|  |  |  |
| --- | --- | --- |
| Prefix | *Pattern* | *Support* |
| 283 | 283 | 3 |
| 283, 281 | 2 |
| 283, 182 | 3 |
| 283, 282 | 3 |
| 283, 182, 281 | 2 |
| 283, 182, 282 | 3 |
| 283, 281, 282 | 2 |
| 283, 182, 281, 282 | 2 |
| 279 | 279 | 2 |
| 279, 283 | 1 |
| 279, 182 | 1 |
| 279, 282 | 2 |
| 279, 281 | 1 |
| 279, 283, 182 | 1 |
| 279, 283, 282 | 1 |
| 279, 182, 282 | 1 |
| 279, 281, 282 | 1 |
| 279, 283, 182, 282 | 1 |
| 281 | 281 | 3 |
|  | 281, 182 | 2 |
|  | 281, 282 | 3 |
|  | 281, 182, 282 | 2 |
| 182 | 182 | 3 |
|  | 182, 282 | 3 |
| 282 | 282 | 4 |

1. *Rules*

Dari hasil *frequent pattern* pada Tabel 3.7, untuk masing masing *itemset* akan di ekstrak *rules*-nya. Setiap itemset akan diproses menggunakan *divide and conquer* yang dapat dilihat pada Tabel 3.8. Setelah mendapatkan hasil anteseden dan konsekuen maka akan dihitung probabilitas sesuai dengan rumus pada sub bab 2.2.3 Algoritma *FP Tree*. Sehingga hasilnya dapat dilihat pada Tabel 3.9 hingga Tabel 3.12.

Tabel 3.8 Contoh Ekstraksi *Rules* Data Sampel

|  |  |  |
| --- | --- | --- |
| Itemset | Anteseden | Konsekuen |
| 283, 182, 281 | 283 | 182, 281 |
| 182 | 283, 281 |
| 281 | 283, 182 |
| 283, 182 | 281 |
| 283, 281 | 182 |
| 182, 281 | 283 |

Tabel 3.9 Hasil *Rules* Data Sampel

|  |  |  |  |
| --- | --- | --- | --- |
| Prefix | Anteseden | Konsekuen | Probability |
| 283 | 283 | 281 | 0.66 |
| 281 | 283 | 0.66 |
| 283 | 182 | 1 |
| 182 | 283 | 1 |
| 283 | 182 | 1 |
| 182 | 283 | 1 |
| 283 | 182,281 | 0.66 |
| 182 | 283,281 | 0.66 |
| 281 | 283,182 | 0.66 |
| 283, 182 | 281 | 0.66 |
| 283, 281 | 182 | 1 |
| 182, 281 | 283 | 0.66 |
| 283 | 182, 282 | 1 |
| 182 | 283, 282 | 1 |
| 282 | 283, 182 | 0.75 |
| 283, 182 | 282 | 1 |
| 283, 282 | 182 | 1 |
| 182, 282 | 283 | 1 |
| 283 | 281, 282 | 0.66 |
| 281 | 283, 282 | 0.66 |

**Tabel 3.10 Lanjutan Tabel 3.9**

|  |  |  |  |
| --- | --- | --- | --- |
| Prefix | Anteseden | Konsekuen | Probability |
| 283 | 282 | 283, 281 | 0.5 |
| 283, 281 | 282 | 1 |
| 283, 282 | 281 | 0.66 |
| 281, 282 | 283 | 0.66 |
| 283 | 182, 281, 282 | 0.66 |
| 182 | 283, 281, 282 | 0.66 |
| 281 | 283, 182, 282 | 0.66 |
| 282 | 283, 182, 281 | 0.5 |
| 283, 182 | 281, 282 | 0.66 |
| 283, 281 | 182, 282 | 1 |
| 283, 282 | 182, 282 | 0.66 |
| 182, 281 | 283, 282 | 1 |
| 182, 282 | 283,281 | 0.66 |
| 281, 282 | 283, 182 | 0.66 |
| 283, 182, 281 | 282 | 1 |
| 283, 182, 282 | 281 | 0.66 |
| 283, 281, 282 | 182 | 1 |
| 182, 281, 282 | 283 | 1 |
| 279 | 279 | 283 | 0.5 |
| 283 | 279 | 0.33 |
| 279 | 182 | 0.5 |
| 182 | 279 | 0.33 |
| 279 | 282 | 1 |
| 282 | 279 | 0.5 |
| 279 | 281 | 0.5 |
| 281 | 279 | 0.33 |
| 279 | 283, 182 | 0.5 |
| 283 | 279, 182 | 0.33 |
| 182 | 279, 283 | 0.33 |
| 279, 283 | 182 | 1 |
| 279, 182 | 283 | 1 |
| 283, 182 | 279 | 0.33 |
| 279 | 283, 282 | 0.5 |
| 283 | 279, 282 | 0.33 |

Tabel 3.11 Lanjutan Tabel 3.10

|  |  |  |  |
| --- | --- | --- | --- |
| Prefix | Anteseden | Konsekuen | Probability |
| 279 | 282 | 279, 283 | 0.25 |
| 279, 283 | 282 | 1 |
| 283, 282 | 279 | 0.33 |
| 279, 282 | 283 | 0.5 |
| 279 | 182, 282 | 0.5 |
| 182 | 279, 282 | 0.33 |
| 282 | 279, 182 | 0.25 |
| 279, 182 | 282 | 1 |
| 279, 282 | 182 | 0.5 |
| 282, 182 | 279 | 0.33 |
| 279 | 281, 282 | 0.5 |
| 281 | 279, 282 | 0.33 |
| 282 | 279, 281 | 0.25 |
| 279, 281 | 282 | 1 |
| 279, 282 | 281 | 0.5 |
| 281, 282 | 279 | 0.33 |
| 279 | 283, 182, 282 | 0.5 |
| 283 | 279, 182, 282 | 0.33 |
| 182 | 279, 283, 282 | 0.33 |
| 282 | 279, 182, 283 | 0.25 |
| 279, 283 | 182, 282 | 1 |
| 279, 182 | 283, 282 | 1 |
| 279, 282 | 283, 182 | 0.5 |
| 283, 182 | 279, 282 | 0.33 |
| 283, 282 | 279, 182 | 0.33 |
| 182, 282 | 279, 283 | 0.33 |
| 279, 283, 182 | 282 | 1 |
| 279, 182, 282 | 283 | 1 |
| 283, 182, 282 | 279 | 0.33 |
| 279, 283, 282 | 182 | 1 |
| 281 | 182 | 0.66 |
| 182 | 281 | 0.66 |
| 281 | 282 | 1 |
| 282 | 281 | 0.75 |
| 281 | 182, 282 | 0.66 |

Tabel 3.12 Lanjutan Tabel 3.11

|  |  |  |  |
| --- | --- | --- | --- |
| Prefix | Anteseden | Konsekuen | Probability |
| 281 | 182 | 281, 282 | 0.66 |
| 282 | 281, 182 | 0.5 |
| 281, 182 | 282 | 0.66 |
| 281, 282 | 182 | 0.66 |
| 182, 282 | 281 | 0.66 |
| 182 | 182 | 282 | 1 |
| 282 | 182 | 0.75 |

### 3.2.3 Perancangan Model *Classifier*

Pada sub bab ini akan dijelaskan bagaimana perancangan model *classifier* yaitu *Naïve Bayes dan* juga *SVM*.

#### 3.2.3.1 *Naïve Bayes*

Naïve Bayes yang digunakan pada model adalah *Gaussian Naïve Bayes* karena *dataset* memiliki format *binary*. Model *Naïve Bayes* akan dibuat berdasarkan dataset yang telah di filter berdasarkan *minimum support* yang sama dengan skenario. Setelah itu dataset yang telah di filter tersebut akan di masukkan sebagai data belajar dari model *Naïve Bayes* yang akan dibuat. Model *Naïve Bayes* akan dibuat menggunakan *library*. Nantinya model ini akan diujikan dengan data testing yang telah ditetapkan sebelumnya.

*Output* dari model ini adalah 5 kemungkinan diagnosis yang memiliki *probability* tertinggi. Dengan format diagnosis dan persentase kemungkinan untuk setiap diagnosis.

#### 3.2.3.2 *Support Vector Machine*

Pada tahap ini akan dibuat model *SVM* dengan *kernel linear*. Pada *Scikit Learn* lebih dikenal dengan *SVC*. Sama seperti *Naïve Bayes*, model *SVM* juga akan dibuat berdasarkan data belajar yang telah di filter berdasarkan *minimum support*. Model ini dibuat dengan library *Scikit Learn*.

*Output* dari model ini adalah 5 kemungkinan diagnosis yang memiliki *probability* tertinggi. Dengan format diagnosis dan persentase kemungkinan untuk setiap diagnosis.

### 3.2.4 Metode Evaluasi

Pada sub bab ini akan dijelaskan metode evaluasi dari tugas akhir ini. Evaluasi ini nantinya akan dibagi menjadi 2 yaitu berdasarkan akurasi dan juga *information retrieval*.

#### 3.2.4.1 Evaluasi Berdasarkan Akurasi

Metrik akurasi akan didasarkan pada hasil klasifikasi dengan probabilitas klasifikasi tertinggi dari model *classifier* dengan hasil yang seharusnya. Dengan rumus sebagai berikut :

(3.1)

**Keterangan** :

FP = *False Positive*

FN = *False Negative*

TP = *True Positive*

TN = *True Negative*

#### 3.2.4.2 Evaluasi Berdasarkan Information Retrieval

Metrik ini dihitung berdasarkan hasil pada data training yang akan dibandingkan dengan hasil model yang memiliki 5 hasil probabilitas tertinggi. Dan data benar tersebut akan dihitung berdasarkan peringkat, 100 untuk peringkat pertama dan turun 20 untuk setiap peringkat sehingga peringkat 5 akan mendapatkan nilai 20 dan jika data tidak muncul akan memberikan nilai 0. Atau disederhanakan menjadi rumus berikut :

(3.2)

**Keterangan** :

Peringkat merupakan urutan diagnosis berdasarkan probabilitas

Selain itu juga akan dihitung nilai kemunculan dari hasil diagnosis. Sehingga jika ada salah satu hasil klasifikasi dari 5 hasil yang sesuai dengan hasil pada data training maka akan dinilai benar.

### 3.2.5 Perancangan Aplikasi

Pada sub-bab ini akan dijelaskan perancangan desain aplikasi. Perancangan aplikasi akan dibagi menjadi 3 yaitu :

* 1. Perancangan Desain Aplikasi Secara Umm
  2. Perancangan Output Tanya Jawab
  3. Perancangan Output Diagnosis

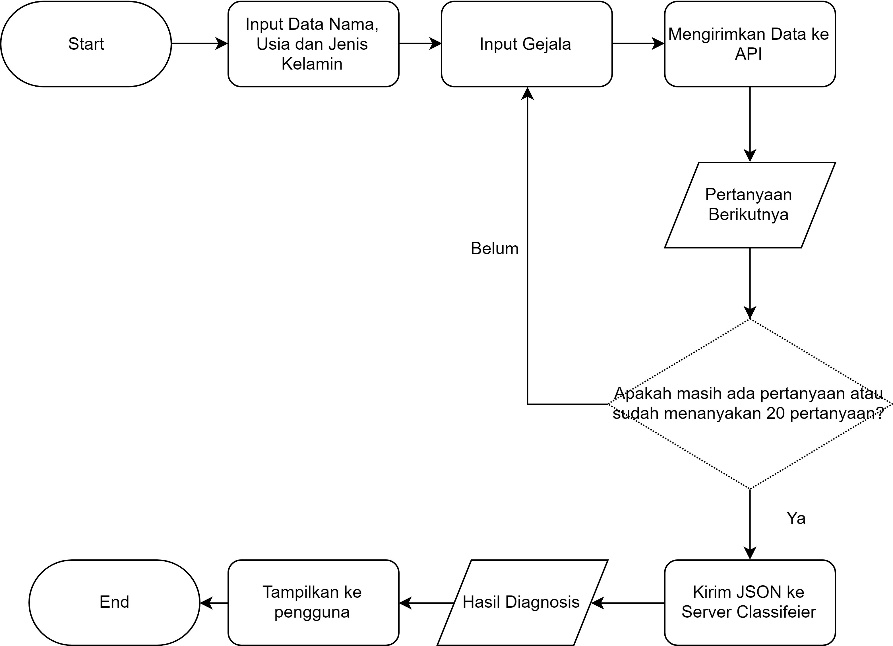
#### 3.2.5.1 Perancangan Desain Aplikasi Secara Umum

Sesuai dengan Gambar 3.4 Diagram Alur Aplikasi, aplikasi akan dimulai dengan menanyakan identitas pasien, nama, usia dan juga jenis kelamin. Setelah itu pasien akan menginputkan salah satu gejala yang dirasa paling mengganggu pasien. Setelah itu aplikasi akan menanyakan pertanyaan gejala yang mungkin berikutnya berdasarkan jawaban dari pertanyaan gejala sebelumnya. Pertanyaan akan memiliki 2 jawaban ya atau tidak.

Aplikasi akan menghentikan pertanyaan dan akan memberikan jawaban jika aplikasi telah menanyakan 20 aplikasi atau *server* tidak memiliki pertanyaan lain untuk ditanyakan.

#### 3.2.5.2 Perancangan *Output* Tanya Jawab

Agar aplikasi dapat menentukan pertanyaan mana yang harus ditanyakan selanjutnya, aplikasi akan mengakses *end-point* yang bertugas untuk menentukan pertanyaan selanjutnya. Adapun *end-point* ini akan menentukan pertanyaan selanjutnya berdasarkan jawaban pertanyaan sebelumnya. Untuk mendapatkan pertanyaan selanjutnya, *server* akan dirancangan sesuai dengan Gambar 3.5 Diagram Algoritma Tanya Jawab.

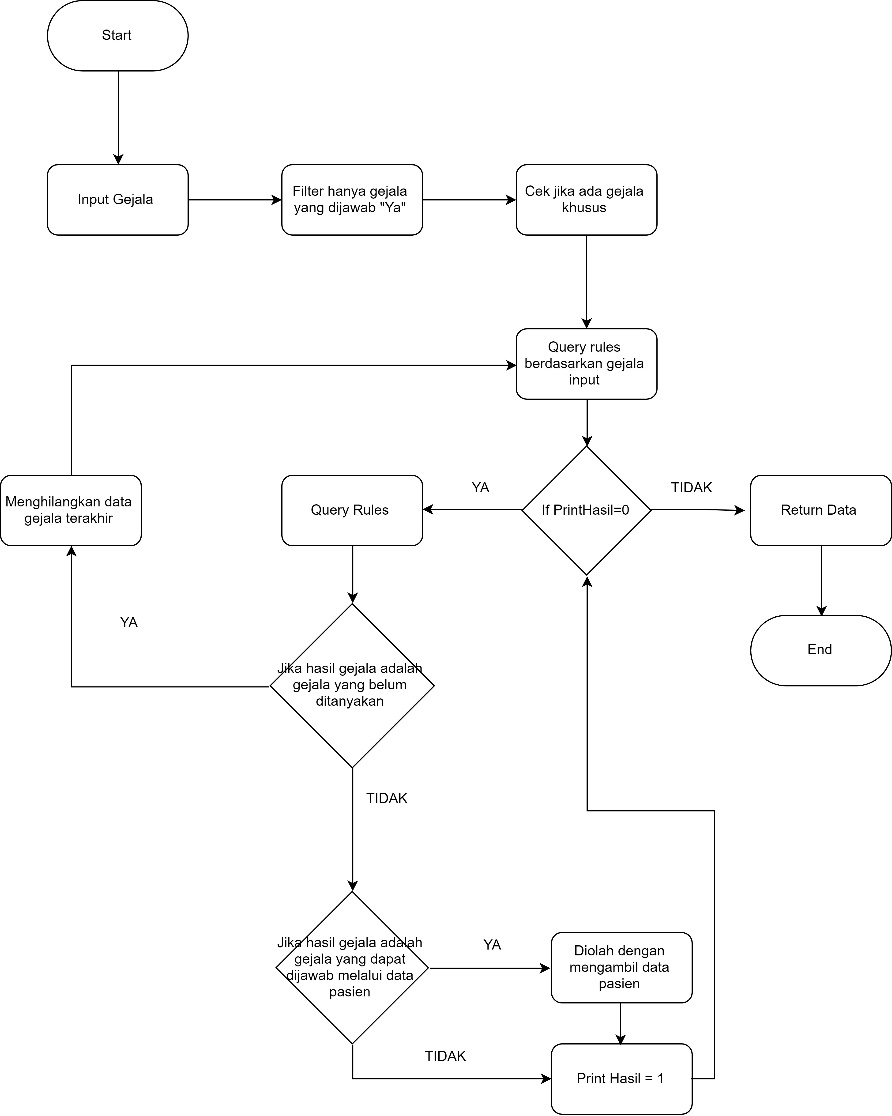


Gambar 3.4 Diagram Alur Aplikasi

Aplikasi tidak langsung mengekstrak *rules* berdasarkan gejala yang telah ditanyakan yang terdapat pada *database.* Melainkan gejala gejala sebelumnya akan diolah oleh *server* dengan cara mengambil data *rules* yang berasal dari output model *FP Tree*, dan akan menentukan kemungkinan pertanyaan selanjutnya berdasarkan input berupa gejala sebelumnya.

Adapun olahan khusus, seperti gejala yang memiliki sub gejala, seperti demam yang memiliki sub “demam tinggi” dan “demam lebih dari 3 hari”, maka mesin akan langsung menanyakan sub gejala sub gejala tersebut jika telah mendapai salah satu sub gejala ditanyakan. Serta pengolahan data khusus yang didasari dari input identitas pasien yaitu usia dan jenis kelamin. Adapun berikut adalah pembagian kategori menurut usia :

1. Bayi 0 – 1
2. Balita dan Remaja 3 - 18
3. Dewasa 18 – 65
4. Tua > 65



Gambar 3.5 Diagram Algoritma Tanya Jawab

#### 3.2.5.3 Perancangan *Output* Diagnosis

Jika aplikasi sudah selesai menanyakan pertanyaan, maka aplikasi akan mengirimkan data gejala yang dialami pasien ke server *machine learning* yang dijalankan menggunakan *flask* dan akan mengolahnya pada model *machine learning*. Setelah itu *server* ini akan mengirimkan kembali hasil data ke aplikasi. Dan aplikasi akan menampilkan 5 kemungkinan diagnosis tertinggi beserta persentase.

***[Halaman ini sengaja dikosongka***

# BAB IV IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijelaskan pada bab sebelumnya. Implementasi kode program bahasa yang berbeda beda yang akan dijelaskan setiap potongan kode.

## Lingkungan Implementasi

Spesifikasi komputer (*server*) yang digunakan untuk implementasi aplikasi ditampilkan pada Tabel 4.1

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| *CPU* | *Intel® Core i5™ 7200U CPU @ 2.50 GHz (4 CPUs), ~ 2.7GHz* |
| *GPU* | *NVidia GeForce 720 930MX 2GB* |
| Sistem Operasi | *Windows 10 Pro* |
| Memori | 8GB DDR4 |
| Penyimpanan | 1 TB |
| Perangkat Lunak Pendukung | * *Python* 3.6 * *Python* 2.7 * *PHP* 7.1 * *MySQL* 5.5 |

## Implementasi

Sub-bab ini akan menjelaskan implementasi dari setiap tahap tahap yang telah dijelaskan pada bab sebelumnya.

### Implementasi FP Tree

Pada sub bab ini akan dijelaskan tahap tahap pembuatan model *FP Tree* yang akan menghasilkan rules. Tahap pembuatan FP Tree ini sendiri ada 3 tahap yaitu :

* 1. *Filter Support*
  2. Pembuatan *Tree*
  3. Ekstrak *Frequent Pattern*
  4. Ekstrak *Rules*

Pada tahap ini semua implementasi kode sumber menggunakan bahasa *Python*.

1. **def** cleanData**(**data**,**min\_support**):**
2. items **=** data
3. item\_counter **=** Counter**(**x **for** sublist **in** data **for** x **in** sublist**)**
4. temp\_item\_counter **=** Counter**(**item\_counter**)**
5. **for** item **in** temp\_item\_counter**:**
6. **if(**temp\_item\_counter**[**item**]** **<** min\_support**):**
7. **del** item\_counter**[**item**]**
8. itemOrder **=** **{}**
9. lengthItems **=** len**(**item\_counter**)**
10. **for** i **in** range**(**1**,**lengthItems**+**1**):**
11. itemOrder**[**item\_counter**.**most\_common**()[-**i**][**0**]]** **=** i**;**
12. count **=** 0**;**
13. **for** item **in** items**:**
14. item **=** filter**(lambda** v**:** v **in** itemOrder**,** item**)**
15. item**.**sort**(**key**=lambda** v**:** itemOrder**[**v**],** reverse**=True)**
16. items**[**count**]** **=** item
17. count**+=**1
18. data **=** **[**items**,**item\_counter**]**
19. **return** data

Kode Sumber 4.1 Filter pada *FP Tree*

#### *Filtering Support*

Pada tahap ini data akan diolah oleh suatu fungsi yang akan memfilter data sesuai dengan *minimum support* yang dimiliki data tersebut. Pada baris ke 5 – 7 atribut pada gejala akan di *filter* sesuai *minimum support*. Dan di baris ke 11 – 12 data akan diurutkan. Dan pada baris ke 15 – 19 data akan di filter sesuai dengan urutan dan atribut mana yang melebihi *minimum support*. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.1.

1. **def** buildTree**(**dataset**):**
2. root **=** Node**(**"root"**,**support **=** 0**)**
3. **for** data **in** dataset**:**
4. parent **=** 0**;**
5. **for** item **in** data**:**
6. **if** parent**==**0**:**
7. parent **=** root
8. child **=** find**(**parent**,** filter\_**=lambda** node**:** node**.**name **==** item**,** stop**=None,** maxlevel **=** 2**)**
9. **if** child**:**
10. child**.**support **=** child**.**support **+** 1
11. parent **=** child
12. **else:**
13. new\_node **=** Node**(**item**,** parent**=**parent**,** support **=** 1**)**
14. parent **=** new\_node
15. **return** root

Kode Sumber 4.2 Pembuatan *Tree* Pada *FP Tree*

#### Pembuatan *Tree*

Pada tahap ini akan data yang telah di filter akan dimasukkan ke sebuah *tree*. Output dari fungsi ini adalah sebuah *tree*, yang mana setiap *node* nya memiliki atribut nama, *child*, *parent*, dan juga *support*. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.2.

#### Mengekstrak *Frequent Pattern*

Pada tahap ini dari *tree* fungsi akan mengekstrak *frequent patt*ern yang digunakan untuk membuat *rules* pada tahap berikutnya. Pada awal fungsi yaitu baris 3 – 6 *tree* akan di *reverse* dari bawah. Selanjutnya *node node* pada *tree* akan di jelajahi satu per satu untuk mencari *frequent pattern* terbanyak.

1. **def** mineFrequentPattern**(**dataset**,** tree**,** min\_support**,**itemCounter**):**
2. root **=** tree
3. itemReverse **=** **[]**
4. lengthItems **=** len**(**itemCounter**)**
5. **for** i **in** range**(**1**,**lengthItems**+**1**):**
6. itemReverse**.**append**(**itemCounter**.**most\_common**()[-**i**][**0**])**
7. condPattern **=** **{}**
8. frequentPattern **=** **[]**
9. **for** item **in** itemReverse**:**
10. nodes **=** findall\_by\_attr**(**root**,** item**)**
11. condPatternParents **=** **[]**
12. **for** node **in** nodes**:**
13. parent **=** str**(**node**)**
14. parent **=** parent**.**replace**(**"/root/"**,**""**)**
15. parent **=** parent**.**split**(**"'"**)[**1**]**
16. **for** i **in** range**(**0**,** node**.**support**):**
17. condPatternParents**.**append**(**parent**)**
18. condPattern**[**item**]** **=** condPatternParents

Kode Sumber 4.3 Ekstrak *Frequent Pattern*

1. **for** item **in** itemReverse**:**
2. itemsInPattern **=** **[]**
3. **for** pattern **in** condPattern**[**item**]:**
4. temp\_pattern **=** pattern**.**split**(**'/'**)**
5. **for** temp\_item **in** temp\_pattern**:**
6. itemsInPattern**.**append**(**temp\_item**)**
7. item\_counter **=** Counter**(**itemsInPattern**)**
8. temp\_item\_counter **=** Counter**(**item\_counter**)**
9. **for** temp\_item **in** temp\_item\_counter**:**
10. **if(**temp\_item\_counter**[**temp\_item**]** **<** min\_support**):**
11. **del** item\_counter**[**temp\_item**]**
12. willBeCombinatedItem **=** **[]**
13. willBeCombinatedItem **=** filter**(lambda** v**:** v **in** item\_counter**,** itemReverse**)**
14. **for** L **in** range**(**1**,** len**(**willBeCombinatedItem**)+**1**):**
15. **for** resultcombo **in** combinations**(**willBeCombinatedItem**,** L**):**
16. resultcombo **=** list**(**resultcombo**)**
17. **if** item **in** resultcombo**:**
18. support **=** 0
19. **for** pattern **in** condPattern**[**item**]:**
20. pattern\_array **=** pattern**.**split**(**'/'**)**
21. flag\_support\_increment **=** 1
22. **for** combo **in** resultcombo**:**
23. **if** combo **not** **in** pattern\_array**:**

Kode Sumber 4.4 Lanjutan Kode Sumber 4.3

1. flag\_support\_increment **=** 0
2. **if** flag\_support\_increment **==** 1**:**
3. support **+=** 1
4. **if** support **>=** min\_support**:**
5. frequentPattern**.**append**(**FrequentPattern**(**item**,**','**.**join**(**resultcombo**) ,**str**(**support**)))**
6. **return** frequentPattern

Kode Sumber 4.5 Lanjutan Kode Sumber 4.4

#### Ekstrak *Rules*

Setelah berhasil mengekstrak *frequent pattern*, program selanjutnya akan mengekstrak *rules* dari data *frequent pattern* dan akan dimasukkan kedalam database. Implementasi ini dapat dilihat pada Kode Sumber 4.6.

### Implementasi *Classifier*

Pada sub bab ini akan dijelaskan bagaimana tahapan pembuatan model *classifier*. Adapun pada tugas akhir ini *classifier* yang digunakan ada 2 yaitu :

* 1. *Naïve B*ayes
  2. *Support Vector Machine*

Semua implementasi pada kode sumber pada tahap ini menggunakan bahasa *Python*.

#### *Naïve Bayes*

Berikut merupakan implementasi kode dari pembuatan model *Naïve Bayes*, yang mana *output-*nya adalah berupa *file* model yang dapat di buka lagi tanpa perlu *training* oleh *server* *machine learning*. Implementasi pada tahap ini dapat dilihat pada Kode Sumber 4.7.

1. **def** generate\_rules**(**filteredList**,**threshold**):**
2. **for** i **in** range**(**len**(**filteredList**)):**
3. subset **=** filteredList**[**i**].**pattern**.**split**(**','**);**
4. **for** L **in** range**(**1**,** len**(**subset**)):**
5. **for** resultsubset **in** combinations**(**subset**,** L**):**
6. left **=** list**(**resultsubset**)**
7. right **=** list**(**subset**)**
8. **for** item **in** left**:**
9. right**.**remove**(**item**)**
10. left\_string **=** ','**.**join**(**left**)**
11. right\_string **=** ','**.**join**(**right**)**
12. subset\_string **=** ','**.**join**(**subset**)**
13. support\_subset **=** find\_by\_item**(**filteredList**,** subset\_string**)**
14. support\_left **=** find\_by\_item**(**filteredList**,** left\_string**)**
15. support **=** float**(**float**(**support\_subset**)/**float**(**support\_left**))**
16. right **=** sorted**(**right**,** key**=lambda** x**:** int**(**x**))**
17. left **=** sorted**(**left**,** key**=lambda** x**:** int**(**x**))**
18. left\_string **=** ','**.**join**(**left**)**
19. right\_string **=** ','**.**join**(**right**)**
20. left\_string **=** left\_string**.**replace**(**" "**,**""**)**
21. right\_string **=** right\_string**.**replace**(**" "**,**""**)**
22. query **=** 'INSERT INTO rules\_23(items,result,probability) VALUES("'**+**left\_string**+**'","'**+**right\_string**+**'","'**+**str**(**support**)+**'");'
23. cursor**.**execute**(**query**)**
24. cnx**.**commit**()**

Kode Sumber 4.6 Ekstrak Rules pada *FP Tree*

1. **def** generate\_model**():**
2. seed **=** 7
3. numpy**.**random**.**seed**(**seed**)**
4. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-training-t5.csv"**,** skipinitialspace**=True)**
5. dataset **=** dataframe**.**values
6. jumlah\_gejala **=** len**(**dataset**[**0**])** **-** 1
7. X\_train **=** dataset**[:,**0**:**jumlah\_gejala**]**
8. Y\_train **=** dataset**[:,**jumlah\_gejala**]**
9. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-testing-t5.csv"**,** skipinitialspace**=True)**
10. dataset **=** dataframe**.**values
11. X\_test **=** dataset**[:,**0**:**jumlah\_gejala**]**
12. Y\_test **=** dataset**[:,**jumlah\_gejala**]**
13. label **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/label.csv"**)**
14. label **=** label**.**values
15. model **=** BernoulliNB**()**
16. y\_pred **=** model**.**fit**(**X\_train**,** Y\_train**).**predict**(**X\_test**)**
17. class\_map **=** model**.**classes\_
18. score **=** model**.**score**(**X\_test**,** Y\_test**)**
19. filename **=** '../Users/Kevin/PycharmProjects/TugasAkhir/NaiveBayes/model\_architecture.sav'
20. pickle**.**dump**(**model**,** open**(**filename**,** 'wb'**))**

Kode Sumber 4.7 Pembuatan Model *Naive Bayes*

#### *Support Vector Machine*

Berikut merupakan implementasi kode dari pembuatan model *Support Vector Machine*, yang mana *output-*nya adalah berupa *file* model yang dapat di buka lagi tanpa perlu *training* oleh *server machine learning*. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.8.

1. **def** generate\_model**():**
2. seed **=** 7
3. numpy**.**random**.**seed**(**seed**)**
4. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-training-t5.csv"**,** skipinitialspace**=True)**
5. dataset **=** dataframe**.**values
6. jumlah\_gejala **=** len**(**dataset**[**0**])** **-** 1
7. X\_train **=** dataset**[:,**0**:**jumlah\_gejala**]**
8. Y\_train **=** dataset**[:,**jumlah\_gejala**]**
9. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-testing-t5.csv"**,** skipinitialspace**=True)**
10. dataset **=** dataframe**.**values
11. X\_test **=** dataset**[:,**0**:**jumlah\_gejala**]**
12. Y\_test **=** dataset**[:,**jumlah\_gejala**]**
13. label **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/label.csv"**)**
14. label **=** label**.**values
15. model = SVC(kernel = 'linear', probability=True).fit(X\_train, Y\_train)y\_pred **=** model**.**fit**(**X\_train**,** Y\_train**).**predict**(**X\_test**)**
16. class\_map **=** model**.**classes\_
17. score **=** model**.**score**(**X\_test**,** Y\_test**)**
18. filename **=** '../Users/Kevin/PycharmProjects/TugasAkhir/NaiveBayes/model\_architecture.sav'
19. pickle**.**dump**(**model**,** open**(**filename**,** 'wb'**))**

Kode Sumber 4.8 Pembuatan Model SVM

### Implementasi Evaluasi

Pada tahap ini akan dijelaskan bagaimana *classifier* melakukan evaluasi dengan masing masing model. Bahasa yang digunakan untuk implementasi pada sumber kode semua dalam bahasa *Python*.

#### Implementasi Evaluasi Akurasi

Berikut adalah implementasi evaluasi akurasi, yang mana akan mengukur akurasi dari model berdasarkan input dan output data *testing*. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.9.

1. **def** getScore**():**
2. filename = '../Users/Kevin/PycharmProjects/TugasAkhir/NaiveBayes/model\_architecture.sav'
3. loaded\_model = pickle.load(open(filename, 'rb'))
4. score = loaded\_model.score(X\_test, Y\_test)
5. return score

Kode Sumber 4.9 Evaluasi Akurasi

#### Implementasi Evaluasi *Retrieval*

Pada tahap ini ada 2 penilaian yang dilakukan. Yaitu penilaian berdasarkan kemunculan diagnosis benar dari 5 diagnosis yang di prediksi dan juga penilaian peringkat kemunculan diagnosis benar dari 5 diagnosis. Implementasi pada tahap ini dapat dilihat pada Kode Sumber 4.10.

### Implementasi Aplikasi

Implementasi terakhir adalah implementasi pada aplikasi. Semua output dari tahap tahap sebelumnya akan disatukan sehingga dapat digunakan oleh user. Adapun pada tahap pembuatan aplikasi ini dibagi menjadi 3 bagian yaitu :

* + 1. Tanya Jawab
    2. Prediksi
    3. *Interface*

1. **def** getIRScore**():**
2. predictions **=** model**.**predict\_proba**(**X\_test**)**
3. index **=** 0
4. filewrite **=** '../Users/Kevin/PycharmProjects/TugasAkhir/laporan/t18/svm\_prediction\_result.txt'
5. **with** open**(**filewrite**,** 'w'**)** **as** result\_file**:**
6. result\_file**.**write**(**'jumlah gejala ' **+** str**(**jumlah\_gejala**)** **+** '\n'**)**
7. result\_file**.**write**(**'jumlah diagnosis ' **+** str**(**len**(**class\_map**))** **+** '\n'**)**
8. result\_file**.**write**(**'akurasi ' **+** str**(**score**)** **+** '\n\n'**)**
9. item\_pred\_IR **=** 0**;**
10. pred\_class\_score\_array **=** **[]**
11. item\_count **=** 0**;**
12. **for** pred **in** predictions**:**
13. top5 **=** pred**.**argsort**()[-**5**:][::-**1**]**
14. item\_pred **=** top5**[**0**]**
15. item\_pred\_str **=** class\_map**[**item\_pred**]**
16. item\_true **=** Y\_test**[**index**]**
17. string **=** str**(**item\_pred\_str**)** **+** ' --- ' **+** str**(**item\_true**)**
18. result\_file**.**write**(**string**+**'\n'**)**
19. pred\_class\_score **=** 100**;**
20. pred\_class\_score\_fix **=** 0**;**

Kode Sumber 4.10 Evaluasi *Information Retrieval*

1. **for** item **in** top5**:**
2. result\_file**.**write**(**str**(**class\_map**[**item**])** **+** ' '**+** str**(**pred**[**item**])+**'\n'**)**
3. temp\_pred **=** str**(**class\_map**[**item**])**
4. temp\_true **=** str**(**item\_true**)**
6. **if(**pred\_class\_score\_fix **==** 0**):**
7. **if(**temp\_true **==** temp\_pred**):**
8. pred\_class\_score\_fix **=** pred\_class\_score
9. item\_pred\_IR **+=** 1
10. **else:**
11. pred\_class\_score **=** pred\_class\_score **-** 20**;**
12. pred\_class\_score\_array**.**append**(**pred\_class\_score\_fix**)**
13. result\_file**.**write**(**'skor IR' **+** str**(**pred\_class\_score\_fix**)** **+** '\n'**)**
14. result\_file**.**write**(**'\n'**)**

Kode Sumber 4.11 Lanjutan Kode Sumber 4.10

#### Tanya Jawab

Pada bagian ini, *server* akan menerima *input* hasil dari pertanyaan terbaru dan pertanyaan sebelumnya. Lalu *server* akan melakukan pengolahan data. Dan mengambil *rules* dari *database* dan akan memberikan pertanyaan berikutnya berdasarkan probabilitas terbesar. Implementasi dari bagian ini menggunakan bahasa PHP. Algoritma lebih jelas dapat dilihat pada Gambar 3.5.

##### Filter Gejala

Dari input data gejala gejala yang telah dijawab pasien, server akan memfilter gejala yang dijawab ya oleh pasien. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.12.

##### Cek Gejala Khusus

Dilakukan cek untuk gejala 10 yaitu batuk dan 23 yaitu demam. Implementasi pada tahap ini dapat dilihat pada   
Kode Sumber 4.13.

1. foreach($gejala as $key=>$value) {
2. if($value == 1)
3. array\_push($arrayGejala, $key);
4. array\_push($arrayGejalaAsked, $key);
5. }

Kode Sumber 4.12 Filter Gejala pada Tanya Jawab Aplikasi

1. if(in\_array(10, $arrayGejalaAsked) && $gejala['10'] == 1)
2. {
3. if(!in\_array(57, $arrayGejalaAsked) && !in\_array(66, $arrayGejalaAsked))
4. {
5. $id = 57;
6. }
7. elseif(in\_array(57, $arrayGejalaAsked) && $gejala['57'] == 0 && !in\_array(66, $arrayGejalaAsked))
8. {
9. $id = 66;
10. }
11. elseif(in\_array(66, $arrayGejalaAsked) && $gejala['66'] == 0 && !in\_array(57, $arrayGejalaAsked))
12. {
13. $id = 57;
14. }

Kode Sumber 4.13 Cek Gejala Khusus pada Tanya Jawab Aplikasi

1. if($id!=0){
2. $rules = Gejala::where('id',$id)->first();
3. return json\_encode([
4. 'command' => 'ask',
5. 'gejala' => $rules,
6. 'append' => 0,
7. 'append\_value' => 0
8. ]);
9. }
10. }
11. if(in\_array(23, $arrayGejalaAsked) && $gejala['23'] == 1){
12. $id = 0;
13. if(!in\_array(41, $arrayGejalaAsked))
14. {
15. $id = 41;
16. }
17. elseif(!in\_array(177, $arrayGejalaAsked))
18. {
19. $id = 177;
20. }
21. if($id!=0)
22. {
23. $rules = Gejala::where('id',$id)->first();
24. return json\_encode([
25. 'command' => 'ask',
26. 'gejala' => $rules,
27. 'append' => 0,
28. 'append\_value' => 0
29. ]);
30. }
31. }

Kode Sumber 4.14 Lanjutan Sumber Kode 4.13

##### *Query Rules*

Akan dilakukan *query* pada data *rules* berulang kali hingga mendapatkan pertanyaan yang akan ditanyakan kepada pasien, sesuai skema pada Gambar 3.5. Implementasi pada tahap ini dapat dilihat pada Kode Sumber 4.15.

1. while(!$printResult)
2. {
3. $rules = Rules::where('items',$stringGejalaSorted)->orderBy('probability','desc')->get();
4. }

Kode Sumber 4.15 *Query Rules* pada Tanya Jawab Aplikai

##### Cek Jika Hasil *Query* Adalah Gejala Yang Belum Ditanyakan

Setelah mendapatkan hasil *query.* Maka hasil terebut akan di cek apakah hasil *query* pada tahap sebelumnya sudah ditanyakan. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.16.

##### Menghilangkan Gejala Terakhir

Setiap hasil *query,* jika belum mendapatkan hasil maka akan berulang kali melakukan tahap ini. Pada input gejala akan dihilangkan atribut gejala yang paling terakhir dimasukkan. Implementasi pada tahap ini dapat dilihat pada Kode Sumber 4.17.

##### Menjawab Melalui Data Pasien

Untuk setiap hasil query. Setiap atribut konsekuen akan di cek apakah dapat dijawab melalui data pasien. Jika ya dan membutuhkan informasi usia, maka server bisa langsung mengekstrak dari usia yang diinput. Implementasi tahap ini dapat dilihat pada   
Kode Sumber 4.18.

1. if(!in\_array($rule->result, $arrayGejalaAsked))
2. {
3. if(in\_array($rule->result, $arrayGejalaSpecial))
4. {
5. $gejala\_result = explode(',', $rule->result);
6. $result = $this->getSpecialRule($gejala\_result[0],$pasien);
7. $command = $result['command'];
8. $gejala\_data = $result['gejala'];
9. $append = $result['append'];
10. $append\_value = $result['append\_value'];
11. $printResult = 1;
12. $found = 1;
13. break;
14. }
15. else
16. {
17. $gejala\_result = explode(',', $rule->result);
18. $gejala\_data = Gejala::where('id',$gejala\_result[0])->first();
19. $printResult = 1;
20. $command = 'ask';
21. $append = 0;
22. $found = 1;
23. break;
24. }
25. }

Kode Sumber 4.16 Cek Hasil Query

1. else
2. {
3. if(count($tempArrayGejala) > 1){
4. $out = 0;
5. array\_pop($tempArrayGejala);
6. $arrayGejalaSorted = $tempArrayGejala;
7. sort($arrayGejalaSorted);
8. $stringGejalaSorted = implode(",", $arrayGejalaSorted);
9. }
10. elseif(count($tempArrayGejala) == 1){
11. $out=0;
12. }
13. }
14. }

Kode Sumber 4.17 Menghilangkan Gejala Terakhir pada Tanya Jawab Aplikai

1. foreach($arrayGejalaSpecialUsia as $age){
2. if(in\_array($age, $arrayGejalaAsked))
3. {
4. array\_push($arrayGejalaAsked, 35);
5. array\_push($arrayGejalaAsked, 48);
6. array\_push($arrayGejalaAsked, 331);
7. array\_push($arrayGejalaAsked, 139);
8. break;
9. }
10. }

Kode Sumber 4.18 Menjawab Melalui Data Pasien pada Tanya Jawab Aplikai

#### Prediksi

Pada bagian ini, *server* akan membuka model hasil *training* yang memiliki performa terbaik. Lalu *server* akan menerima *json* dari *client* dan akan mengkonversi data menjadi array yang siap untuk di prediksi pada menggunakan model. Lalu hasil akan dikirimkan dengan format *json* ke *client*. Implementasi tahap ini dapat dilihat pada Kode Sumber 4.19.

1. **def** predict**():**
2. total\_attributes**=**376
3. label **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/label.csv"**)**
4. label **=** label**.**values
5. attributes **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/attributes-t5.csv"**)**
6. attributes **=** attributes**.**values
7. data **=** request**.**json
8. data\_gejala **=** data**.**get**(**"gejala"**)**
9. number **=** str**(**5**)**
10. gejala\_array **=** **[]**
11. **for** i **in** attributes**:**
12. number **=** str**(**i**[**0**])**
13. **if(**data\_gejala**.**get**(**str**(**number**))):**
14. value\_gejala **=** data\_gejala**.**get**(**number**)**
15. **else:**
16. value\_gejala **=** 0
17. gejala\_array**.**append**(**value\_gejala**)**
18. gejala\_array\_np\_dummy **=** **[]**

Kode Sumber 4.19 Implementasi Prediksi pada Server

1. gejala\_array\_np\_dummy**.**append**(**gejala\_array**)**
2. gejala\_array\_np **=** numpy**.**array**(**gejala\_array\_np\_dummy**)**
3. seed **=** 7
4. numpy**.**random**.**seed**(**seed
5. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-training-t5.csv"**,** skipinitialspace**=True)**
6. dataset **=** dataframe**.**values
7. jumlah\_gejala **=** len**(**dataset**[**0**])** **-** 1
8. X\_train **=** dataset**[:,**0**:**jumlah\_gejala**]**
9. Y\_train **=** dataset**[:,**jumlah\_gejala**]**
10. #assigning testing
11. dataframe **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/classifier-testing-t5.csv"**,** skipinitialspace**=True)**
12. dataset **=** dataframe**.**values
13. X\_test **=** dataset**[:,**0**:**jumlah\_gejala**]**
14. Y\_test **=** dataset**[:,**jumlah\_gejala**]**
15. #label
16. label **=** pandas**.**read\_csv**(**"../Users/Kevin/PycharmProjects/TugasAkhir/dataset/label.csv"**)**
17. label **=** label**.**values
18. #Create a Gaussian Classifier
19. model **=** BernoulliNB**()**

Kode Sumber 4.20 Lanjutan Kode Sumber 4.19

1. # Train the model using the training sets
2. y\_pred **=** model**.**fit**(**X\_train**,** Y\_train**).**predict**(**X\_test**)**
3. class\_map **=** model**.**classes\_
4. score **=** model**.**score**(**X\_test**,** Y\_test**)**
5. predictions **=** model**.**predict\_proba**(**gejala\_array\_np**)**
6. result **=** **[]**
7. jsondata **=** '{ "result":['
8. index **=** 0
9. **for** pred **in** predictions**:**
10. top5 **=** pred**.**argsort**()[-**5**:][::-**1**]**
11. **for** item **in** top5**:**
12. labels **=** label**[**item**][**0**]**
13. result**.**append**(**HasilDiagnosis**(**labels**,**pred**[**item**]))**
14. json\_item\_string **=** '{"diagnosis":"' **+** str**(**class\_map**[**item**])** **+**'","probability":"'**+** str**(**pred**[**item**])** **+**'"}'
15. **if** index **<** 4**:**
16. json\_item\_string **=** json\_item\_string **+** ','
17. index **=** index**+**1
18. jsondata **=** jsondata **+** json\_item\_string
19. jsondata **=** jsondata **+** "]}"
20. format\_jsondata **=** json**.**dumps**(**jsondata**)**
21. **return** jsondata

Kode Sumber 4.21 Lanjutan Kode Sumber 4.20

#### *Interface*

Pada bagian interface akan dibagi menjadi 3 potongan kode. Bagian pertama adalah *HTML*, bagian kedua adalah *Javascript* untuk tanya jawab, dan bagian yang ketiga adalah *javascript* untuk diagnosis penyakit. Implementasi pada bagian pertama dapat dilihat pada Kode Sumber 4.22. Implementasi pada bagian kedua dapat dilihat pada Kode Sumber 4.26. Dan Implementasi pada bagian ketiga dapat dilihat pada Kode Sumber 4.29.

1. @section('content')
2. <div class="container">
3. <div class="row">
4. <div class="col-xs-12 col-md-8 col-md-offset-2">
5. <div class="card">
6. <div class="card-content" style="padding-top: 50px; padding-bottom: 50px">
7. <div id="welcome" class="text-center">
8. <h4>Selamat datang di Aqeela dari Medify</h4>
9. <p>Aqeela akan membantu anda untuk mendiagnosis penyakit anda berdasarkan gejala yang anda alami. </p>
10. <button class="btn btn-primary">Selanjutnya<div class="ripple-container"></div></button>
11. </div>
12. <div id="pasien" class="text-center" style="display: none">
13. <h4>Isi data berikut</h4>
14. <div class="row">
15. <div class="col-md-8 col-md-offset-2 col-xs-12">

Kode Sumber 4.22 Implementasi HTML pada Aplikasi Pengguna

1. <input class="form-control" type="text" id="pasien-nama" placeholder="Nama">
3. <input class="form-control" type="text" id="pasien-usia" placeholder="Usia">
4. <select class="selectpicker" data-style="btn btn-primary btn-round" title="Single Select" data-width="100%" id="pasien-gender">
5. <option value="1" selected>Laki laki</option>
6. <option value="2">Perempuan</option>
7. </select>
8. </div></div>
9. <button class="btn btn-primary">Selanjutnya<div class="ripple-container"></div></button>
10. </div>
11. <div id="gejala-first" class="text-center" style="display: none">
12. <h4>Apa gejala yang paling anda rasakan?</h4>
13. <div class="row">
14. <div class="col-md-8 col-md-offset-2 col-xs-12">
15. <select class="selectpicker" data-style="btn btn-primary btn-round" title="Single Select" data-live-search="true" data-size="7">
16. @foreach($gejalas as $gejala)
17. <option value="{{$gejala->id}}">{{$gejala->name}}</option>
18. @endforeach
19. </select>

Kode Sumber 4.23 Lanjutan Kode Sumber 4.22

1. </div>
2. </div>
3. <button class="btn btn-primary">Selanjutnya<div class="ripple-container"></div></button>
4. </div>
5. <div id="gejala-question" class="text-center" style="display: none;min-height: 300px">
6. <h4>Apakah anda merasakan <span id="fill-gejala"></span></h4>
7. <button class="btn btn-primary yes">Ya<div class="ripple-container"></div></button>
8. <button class="btn btn-danger no">Tidak<div class="ripple-container"></div></button>
9. </div>
10. <div id="diagnosis-warning" class="text-center" style="display: none;">
11. <h4>Setelah ini, saya akan memberikan kamu 5 diagnosis penyakit yang mungkin kamu alami</h4>
12. <p>Ingat diagnosis ini tidak bersifat final, kami rekomendasikan anda menemui dokter untuk mendapatkan kepastian mengenai penyakit yang anda derita.</p>
13. <button class="btn btn-primary">Selanjutnya<div class="ripple-container"></div></button>
14. </div>
15. <div id="diagnosis-result" class="row" style="display: none">
16. <h4 class="text-center ">Penyakit yang mungkin kamu derita</h4>
17. <div id="result" class="col-md-4 col-md-offset-4">

Kode Sumber 4.24 Lanjutan Kode Sumber 4.23

1. </div>
2. </div>
3. </div>
4. </div>
5. </div>
6. </div>
7. </div>
8. @endsection

Kode Sumber 4.25 Lanjutan Kode Sumber 4.24

1 **function** setGejala**(**index**,**value**){**

2 **if(**value **==** 1**)**

3 **{**

4 count\_true**++**

5 **}**

6 count**++**

7 gejalaPasien**.**set**(**index**,**value**)**

8 **var** pasienNama **=** $**(**'#pasien-nama'**).**val**()**

9 **var** pasienUsia **=** $**(**'#pasien-usia'**).**val**()**

10 **var** pasienGender **=** $**(**'#pasien-gender'**).**val**()**

11

12 **var** gejalaPasienJSON **=** **new** Backbone**.**Model**({**

13 "gejala" **:** gejalaPasien**,**

14 "jenis\_kelamin" **:** pasienGender**,**

15 "usia" **:** pasienUsia

16 **});**

17

18 **var** gejalaPasienJSON **=**   
 JSON**.**stringify**(**gejalaPasienJSON**);**

19

Kode Sumber 4.26 Javascript Tanya Jawab Pada Interface User

20 $**.**ajax**({**

21 type**:** "POST"**,**

22 url**:** "{{url('api/app/question/next')}}"**,**

23 contentType**:** "application/json"**,**

24 headers**:** **{**

25 'Access-Control-Allow-Credentials' **:**   
 'true'

26 **},**

27 data **:** gejalaPasienJSON**,**

28 processData**:** **false,**

29 success**:** **function** **(**data**)** **{**

30 **var** obj **=** JSON**.**parse**(**data**)**

31 **var** command **=** obj**.**command

32 **var** gejala **=** obj**.**gejala

33 **var** append **=** obj**.**append

34 **var** append\_value **=** obj**.**append\_value

35 currentGejala **=** gejala**.**id

36 console**.**log**(**count**)**

37 **if(**count\_true **>** 10**)**

38 **{**

39 submitFinal**(**gejalaPasienJSON**);**

40 **}**

41 **else** **if(**count **>** 20**)**

42 **{**

43 submitFinal**(**gejalaPasienJSON**);**

44 **}**

45 **else**

46 **{**

47 **if(**command **===** 'ask'**)**

Kode Sumber 4.27 Lanjutan Kode Sumber 4.26

48 **{**

49 console**.**log**(**'ask'**)**

50 $**(**"#gejala-  
 question"**).**fadeOut**(**500**,** **function(){**

51 $**(**"#gejala-  
 question"**).**fadeIn**(**500**);**

52 $**(**'#gejala-question   
 #fill-gejala'**).**html**(**gejala**.**name' - '**+**   
 gejala**.**id**);**

53 **});**

54 **}**

55 **else** **if(**command **===** 'append'**)**

56 **{**

57 console**.**log**(**'append'**)**

58   
 setGejala**(**append**,**append\_value**);**

59 **}**

60 **else**

61 **{**

62 console**.**log**(**command**)**

63   
 submitFinal**(**gejalaPasienJSON**);**

64 **}**

65 **}**

66 **},**

67 error**:** **function** **(**e**)** **{**

68 **alert(**e**)**

69 **}**

70 **});**

71 **}**

Kode Sumber 4.28 Lanjutan Kode Sumber 2.27

1 **function** submitFinal**(**gejalaPasienJSON**)**

2 **{**

3 $**.**ajax**({**

4 type**:** "POST"**,**

5 url**:** "http://127.0.0.1:5000/predict"**,**

6 contentType**:** "application/json"**,**

7 headers**:** **{**

8 'Access-Control-Allow-Credentials' **:**   
 'true'

9 **},**

10 data **:** gejalaPasienJSON**,**

11 dataType **:** 'json'**,**

12 processData**:** **false,**

13 success**:** **function** **(**response**)** **{**

14 diagnosis **=** response**.**result**;**

15 $**(** "#diagnosis-result #result"   
 **).**html**(**''**)**

16

17 $**.**each**(**diagnosis**,** **function(**item**)** **{**

18 probability **=**   
 Number**(**diagnosis**[**item**].**probability **\***   
 100**).**toFixed**(**2**);**

19 result **=** '<h4   
 class="title">'**+**diagnosis**[**item**].**diagnosis**+** '<small>· '**+**probability**+**'%</small></h4>'

20

21 $**(** "#diagnosis-result #result" **).**append**(** result **);**

22 **})**

23

24 $**(**"#gejala-question"**).**fadeOut**(**500**,**   
 **function(){**

Kode Sumber 4.29 Implementasi *Javascript* Prediksi Diagnosis pada *user*

25 $**(**"#diagnosis-  
 warning"**).**fadeIn**(**500**);**

26 **});**

27 **},**

28 error**:** **function** **(**e**)** **{**

29 **alert(**e**);**

30 **}**

31 **});**

32

Kode Sumber 4.30 Lanjutan Kode Sumber 4.29

# BAB V PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan skenario dan hasil dari uji coba aplikasi yang telah diimplementasikan.

## Lingkungan Pengujian

Lingkungan pengujian yang digunakan untuk *server* dengan spesifikasi pada Tabel 5.1 Lingkungan Pengujian *Server* dan *client* (pengguna) dengan spesifikasi pada Tabel 5.2 Lingkungan Pengujian *Client* (Pengguna).

Tabel 5.1 Lingkungan Pengujian *Server*

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| *CPU* | *Intel® Core i5™ 7200U CPU @ 2.50 GHz (4 CPUs), ~ 2.7GHz* |
| *GPU* | *NVidia GeForce 720 930MX 2GB* |
| Sistem Operasi | *Windows 10 Pro* |
| Memori | 8GB DDR4 |
| Penyimpanan | 1 TB |
| Perangkat Lunak Pendukung | * *Python* 3.6 * *Python* 2.7 * *PHP* 7.1 * *MySQL* 5.5 * *Google Chrome* |

## Data Uji Coba

Data yang digunakan untuk uji coba aplikasi ini adalah kumpulan data kasus yang berasal dari dataset dan berjumlah 199 data. Sedangkan data training yang digunakan pada uji coba ini adalah 660 data kasus yang berasal dari dataset. Adapun data rules yang digunakan pada skenario data uji coba ini dijelaskan pada Tabel 5.3.

Tabel 5.2 Lingkungan Pengujian *Client* (Pengguna)

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| *CPU* | *Intel® Core i5™ 7200U CPU @ 2.50 GHz (4 CPUs), ~ 2.7GHz* |
| *GPU* | *NVidia GeForce 720 930MX 2GB* |
| Sistem Operasi | *Windows 10 Pro* |
| Memori | 8GB DDR4 |
| Penyimpanan | 1 TB |
| Perangkat Lunak Pendukung | * *Python* 3.6 * *Python* 2.7 * *PHP* 7.1 * *MySQL* 5.5 * *Google Chrome* |

Tabel 5.3 Jumlah Data Gejala dan Rules

|  |  |  |
| --- | --- | --- |
| **Minimum Support** | **Jumlah Gejala** | **Rules** |
|
| **3** | 250 | 289,940 |
| **5** | 204 | 30,940 |
| **8** | 159 | 6,006 |
| **10** | 133 | 3,290 |
| **12** | 110 | 1,932 |
| **15** | 88 | 1,070 |
| **18** | 74 | 706 |
| **20** | 64 | 528 |
| **23** | 57 | 346 |

## Skenario Uji Coba

Skenario pertama yang dijalankan pada evaluasi ini berupa perhitungan akurasi dan *retrieval*. Pengujian ini bertujuan untuk mencari *model classifer* terbaik yang akan digunakan pada *server classifier*. Pengujian akan dimulai dengan filter *support* untuk gejala gejala. Ada 9 *minimum support* yang akan di filter (3, 5, 8, 10, 12, 15, 18, 20, 23). Setelah filter, data testing gejala ini akan di ujikan kepada model *classifer*. Dan hasilnya akan dicatat pada file teks.

Skenario kedua adalah menguji coba aplikasi kepada pengguna. Dengan batasan pertanyaan hingga 30, pengguna akan menggunakan aplikasi hingga pengguna merasa jenuh. Jika pengguna telah merasa jenuh, pengguna dapat memberitahu dan diizinkan untuk berhenti melakukan pengujian aplikasi. Tujuan dari skenario ini adalah mencari jumlah pertanyaan yang tidak menyebabkan pengguna bosan hingga berakibat keluar dari aplikasi sebelum muncul diagnosis dari aplikasi.

Pada skenario kedua ini uji pengguna dilakukan oleh masyarakat awan yang berusia 20 – 25 tahun. Masing masing mencoba aplikasi 1 kali, setiap pengguna diberikan 1 diagnosis penyakit yang populer pada skenario ini yang digunakan adalah demam berdarah, dan ankilostomiasis (cacing tambang). Jika pengguna tidak mengetahui banyak mengenai diagnosis tersebut maka akan diberikan sedikit gambaran dan daftar gejala yang mungkin. Dan setelah diberikan penjelasan dan daftar gejala pengguna bebas untuk mengisi berdasarkan apa yang dibaca atau pengalaman yang dia miliki.

Selain itu aplikasi juga akan diujikan kepada dokter untuk menilai bagaimana pendapat dokter mengenai akurasi dari pemetaan gejala dan juga hasil diagnosis dokter. Pada uji coba ini dokter akan memasukkan gejala awal dan menjawab pertanyaan, lalu dokter akan menyimpulakan apakah diagnosis yang diberikan masuk akal dengan hasil yang diberikan oleh aplikasi. Pada pengujian ini dokter akan melakukan pengujian aplikasi sebanyak 10 kali.

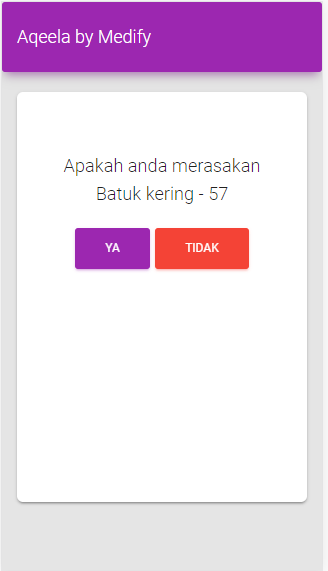
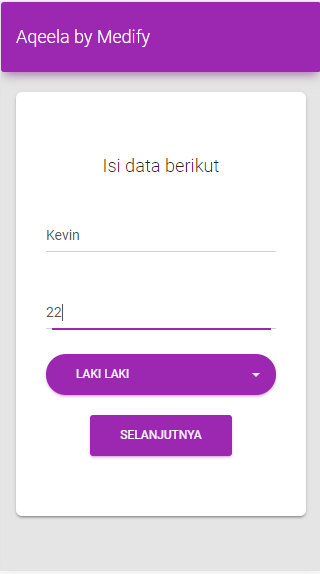
## Hasil Uji Coba

Pada sub bab ini akan dijelaskan bagaimana hasil dari hasil uji coba dari evaluasi secara sistem dan juga evaluasi oleh dokter. Adapun Gambar 5.1, Gambar 5.2, dan Gambar 5.3 adalah contoh tampilan dari aplikasi yang akan digunakan untuk pengujian.

Gambar 5.1 Tampilan Hasil Diagnosis

Gambar 5.2 Pengisian Data Pasien

Gambar 5.3 Tampilan Tanya Jawab



### Hasil Uji Sistem

Hasil uji skenario sistem akan dibagi menjadi 2 jenis evaluasi. Evaluasi secara akurasi dan evaluasi secara *information retrieval.* Pada uji coba berdasarkan akurasi ini dibagi menjadi 2 *classifer* yaitu *Naïve Bayes* dan *SVM*.

Pada Gambar 5.4 dan

Tabel 5.4, dapat dilihat bahwa cenderung menurun walaupun dengan selisih yang sangat kecil. Pada saat *minimum support* bernilai 5, *Naïve Bayes* memiliki akurasi tertinggi sedangkan SVM memiliki akurasi tertinggi pada saat *minimum support* bernilai 3.

Tabel 5.4 Hasil Uji Akurasi

|  |  |  |
| --- | --- | --- |
| ***Min Support*** | ***Naïve Bayes*** | ***SVM*** |
| **3** | 75.897 | 75.897 |
| **5** | 76.381 | 75.376 |
| **8** | 74.371 | 73.366 |
| **10** | 74.874 | 71.859 |
| **12** | 71.859 | 71.356 |
| **15** | 71.859 | 69.849 |
| **18** | 63.316 | 60.301 |
| **20** | 60.301 | 57.788 |
| **23** | 56.281 | 51.256 |
| **Rata rata** | 69.45 | 67.44 |

Selisih pada setiap hasil akurasi juga berbanding tipis antar *minimum support*. Rata rata selisih hasil akurasi classifier pada setiap *minimum support* adalah 2,69 untuk *Naïve Bayes* dan 3,08 untuk *SVM*. Selisih terbesar antar support terjadi saat *minimum support* 15 dibandingkan dengan *minimum support* 18, selisih antar kedua hasil ini adalah 8,5% untuk *Naïve Bayes* dan 9,5% untuk *SVM*. Adapun rata rata selisih hasil akurasi antar *minimum support* pada saat *minimum support* diantara 3 – 15 adalah 1.2% untuk *Naïve Bayes* dan *SVM*. Selisih dari hasil tertinggi dengan terendah adalah 20,1% untuk *Naïve bayes* dan 24,6% untuk *SVM*. Rata rata dari *Naïve Bayes* adalah 69,45% dan *SVM* adalah 67,44%.

Gambar 5.4 Grafik Hasil Uji Akurasi

Tabel 5.5 Hasil Uji Retrieval

|  |  |  |
| --- | --- | --- |
| ***Min Support*** | ***Naïve Bayes*** | ***SVM*** |
| **3** | 94,47% | 93,96% |
| **5** | 94,97% | 92,46% |
| **8** | 95,47% | 93,46% |
| **10** | 95,47% | 93.96% |
| **12** | 95,47% | 95,47% |
| **15** | 93,46% | 92,96% |
| **18** | 92,46% | 89,94% |
| **20** | 90,45% | 88,94% |
| **23** | 88,44% | 87,93% |

Berdasarkan hasil uji *retrieval* pada Tabel 5.5 dan Gambar 5.5 skor tertinggi adalah 190 yang dimiliki *Naïve Bayes* saat *minimum support* bernilai 8, 10, dan 12. Skor 190 juga dicatatkan oleh SVM pada saat *minimum support* bernilai 12. Selisih rata rata *retrieval* antar *minimum support* pada *Naïve Bayes* bernilai 2 sedangkan pada *SVM* bernilai 3. Sedangkan saat *minimum support* bernilai 3 hingga 15 *Naïve Bayes* memiliki skor rata rata selisih 1.2 sedangkan *SVM* 2.8. Pada *Naïve Bayes* penurunan skor tertinggi terjadi ketika saat *minimum support* ke 12 menuju 15, 18 menuju 20 dan 20 menuju 23. Sedangkan pada *SVM* penurunan skor tertinggi terjadi saat *minimum* *support* ke 15 menuju 18.

Gambar 5.5 Grafik Hasil Uji *Retrieval*

Berdasarkan hasil uji peringkat nilai pada   
Gambar 5.6 dan Tabel 5.6 nilai tertinggi adalah 88,04 yang dimiliki *Naïve Bayes* saat *minimum support* bernilai 10. *SVM* memiliki nilai tertinggi sebesar 85.829. Selisih rata rata nilai peringkat antar *minimum support* pada *Naïve Bayes* bernilai 1.53 sedangkan pada *SVM* bernilai 2.6. Sedangkan saat *minimum support* bernilai 3 hingga 15 *Naïve Bayes* memiliki skor rata rata selisih 0.46 sedangkan *SVM* 1.95. Pada *Naïve Bayes* penurunan skor tertinggi terjadi ketika saat *minimum support* ke 15 menuju 18. SVM mengalami penurunan skor tertinggi terjadi juga saat minimum support ke 15 menuju 18.

Tabel 5.6 Hasil Uji Peringkat

|  |  |  |
| --- | --- | --- |
| ***Min Support*** | ***Naïve Bayes*** | ***SVM*** |
| **3** | 87,135 | 85,829 |
| **5** | 87,738 | 85,829 |
| **8** | 87,738 | 86,030 |
| **10** | 88,040 | 85,326 |
| **12** | 87,738 | 85,628 |
| **15** | 86,633 | 82,512 |
| **18** | 82,512 | 78,894 |
| **20** | 80,201 | 76,884 |
| **23** | 76,683 | 73,969 |

Gambar 5.6 Grafik Hasil Uji Peringkat

### Hasil Uji Pengguna

Uji pengguna dilakukan oleh 10 orang dengan usia 20 – 25 tahun. Dengan hasil yang dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil Uji Pengguna Dengan Pertanyaan Maksimal 30

|  |  |
| --- | --- |
| **Pengguna** | **Pertanyaan ke** |
| 1 | 20 |
| 2 | 22 |
| 3 | 18 |
| 4 | 15 |
| 5 | 18 |
| 6 | 23 |
| 7 | 22 |
| 8 | 23 |
| 9 | 20 |
| 10 | 17 |

Tabel 5.8 Hasil Uji Pengguna Dengan Pertanyaan Maksimal 20

|  |  |
| --- | --- |
| **Pengguna** | **Pertanyaan ke** |
| 1 | 20 |
| 2 | 20 |
| 3 | 20 |
| 4 | 20 |
| 5 | 20 |
| 6 | 20 |
| 7 | 20 |
| 8 | 20 |
| 9 | 18 |
| 10 | 18 |

Berdasarkan hasil pada Tabel 5.7 karena tidak ada pengguna yang menggunakan aplikasi hingga selesai maka uji coba dilaksanakan ulang terhadap pengguna yang sama dengan jumlah pertanyaan sebanyak 20 dengan hasil yang dapat dilihat pada Tabel 5.8.

Dapat dilihat pada Tabel 5.8, 80% pengguna menyelesaikan aplikasi hingga memunculkan diagnosis. Kebanyakan pengguna berkomentar jika ada gejala yang tidak berhubungan, dan merasa bosan jika gejala tidak berhubungan. Tetapi jika gejala saling terkait dan mereka menjawab Ya maka keinginan mereka untuk menjawab pertanyaan berikutnya menjadi lebih tinggi.

### Hasil Uji Dokter

Uji pengguna dilakukan oleh dokter yang langsung mencoba aplikasi. Dari 10 percobaan yang dilakukan oleh dokter. Dokter menilai bahwa aplikasi cukup baik walau diagnosis yang diberikan terkadang sedikit meleset. Dari segi pertanyaan dokter menilai sudah relevan dan saling terkait. Berikut merupakan salah satu hasil uji coba yang dilakukan dokter:

Informasi pasien :

Nama pasien : Kevin

Jenis kelamin : Laki - laki

Usia : 22 Tahun

Hasil anamnesis dengan format deskripsi gejala (jawaban) :

* Sakit perut (Ya)
* Muntah (Ya)
* Mual (Ya)
* Demam (Tidak)
* Diare (Tidak)
* Nafsu makan berkurang (Tidak)
* Malaise (Tidak)
* Kepala sakit (Ya)
* Demam tinggi (Tidak)
* Otot nyeri (Tidak)
* Buang air besar berdarah (Tidak)
* Perut kram (Ya)
* Kulit bintik bintik merah (Tidak)
* Perdarah gusi (Tidak)
* Kulit ruam (Tidak)
* Batuk (Ya)
* Batuk kering (Tidak)
* Batuk berdahak (Tidak)
* Tenggorokan Sakit (Ya)
* Hidung mengeluarkan ingus (Ya)
* Bersin (Ya)

Hasil diagnosis :

* faringitis· 92.24%
* influenza· 3.00%
* rhinitis· 2.14%
* leptospirosis· 0.83%
* ankilostomiasis· 0.41%

## Analisis Hasil Uji Coba

Berdasarkan hasil uji coba pada sistem, *Naïve Bayes* memberikan hasil yang lebih baik daripada *SVM*. Peningkatan *minimum support* secara umum menyebabkan akurasi *classifier* turun. Hal ini dikarenakan gejala yang semakin sedikit menyebabkan kurangnya variasi fitur dari data.

Selain itu data dengan akurasi yang tertinggi tidak menyebabkan skor *retrieval* maupun peringkat juga menjadi tertinggi. Hal ini mungkin disebabkan karena sistem peringkat, karena tidak munculnya diagnosis pada 5 diagnosis tertinggi menyebabkan skor peringkat bernilai 0. Walaupun selisih nya sedikit, data dengan *classifier* Naïve Bayes dan *minimum support* sebesar 3 memberikan performa yang terbaik.

Jika diperhatikan hasil *retrieval* pada uji coba ini, artinya ada diagnosis yang tidak muncul pada uji coba data uji. Dari analisis *dataset* dapat diperkirakan penyebabnya adalah banyaknya diagnosis yang memiliki gejala sama dengan diagnosis lainnya, terutama pada diagnosis mata, kardiovaskular, dan pernapasan.

Pada hasil uji coba pengguna dapat dilihat bahwa terlalu banyak pertanyaan menyebabkan pengguna jenuh. Walaupun secara teori akan memberikan hasil retrieval yang lebih baik karena fitur semakin banyak dan spesifik, tetapi jika pengguna enggan menyelesaikan maka tujuan dari aplikasi ini juga tidak dipenuhi.

Pengguna mengeluhkan adanya ketidaksesuaian pertanyaan gejala dengan pertanyaan sebelumnya dapat disebabkan oleh beberapa hal berikut:

* Aplikasi memberikan gejala yang meluas untuk sebelum memberika gejala yang spesifik. Sedangkan kebanyakan pengguna langsung menginginkan pertanyaan yang spesifik berdasarkan pengetahuan dia mengenai penyakit tersebut.
* Banyak data pada sumber data yang memberikan daftar gejala berbeda dengan yang pernah dialami pengguna. Sehingga pengguna merasa gejala tersebut tidak relevan.
* Pengguna merasa pertanyaan diulang, karena ada banyak gejala mirip tapi tak sama. Seperti perut nyeri dengan mual, sakit kepala dan pusing.

Pada uji dokter, dari penilaian dokter sudah memberikan respon positif baik dari segi hasil diagnosis maupun relevansi antar pertanyaan. Walau ada beberapa diagnosis yang salah, umumnya diagnosis yang salah tersebut menempati peringkat 4 dan 5. Sehingga tidak terlalu mengganggu hasil diagnosis.

# BAB VI KESIMPULAN DAN SARAN

Pada bab ini akan diberikan hasil kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran tentang pengembangan dari tugas akhir ini yang dapat dilakukan di masa yang akan datang.

## Kesimpulan

Kesimpulan yang diperoleh berdasarkan hasil uji coba dan evaluasi tugas akhir ini adalah :

* + - 1. *FP Tree* memiliki hasil yang cukup baik dalam memetakan gejala gejala yang berkaitan sehingga dapat membuat tanya jawab yang saling terkait.
      2. *Naïve Bayes* dan *SVM* dapat mendiagnosis penyakit berdasarkan input gejala pasien dengan baik.
      3. Berdasarkan hasil pengujian *Naïve Bayes* dan SVM memberikan hasil yang hampir sama. Tetapi *Naïve Bayes* memberikan hasil yang lebih tinggi daripada SVM pada saat *minimum support* bernilai 5 dengan hasil akurasi sebesar 76,381%, hasil uji peringkat sebesar 87,738%, dan hasil uji *retrieval* sebesar 95,47%.
      4. Filter *minimum support* pada data gejala memberikan hasil yang cukup signifikan pada hasil diagnosis yang diberikan. Dengan selisih rata rata antar akurasi sebesar 2,69% untuk *Naïve Bayes* dan 3,08% untuk *SVM*.

## Saran

Saran yang diberikan untuk pengembangan aplikasi ini kedepannya adalah :

Memperbanyak variasi data kasus dan diagnosis, agar gejala yang ditanyakan lebih bervariatif dan hasil diagnosis yang lebih tepat. Data kasus juga didasarkan pada data lapangan yang mungkin di masa depan bisa didapatkan melalui lembaga pemerintah atau rumah sakit. Perkiraan agar data menjadi lebih baik adalah sekitar 100 kasus untuk setiap diagnosis.

Jika saran pertama terpenuhi, disarankan melakukan percobaan diagnosis menggunakan *classifier* *neural network*, karena *classifier* tersebut secara teori memiliki performa yang baik terhadap *dataset* yang berukuran besar.

# DAFTAR PUSTAKA

|  |  |  |
| --- | --- | --- |
| [1] |  | A. Kirk, "The Telegraph," Telegraph Media Group, 24 Juli 2015. [Online]. Available: https://www.telegraph.co.uk/news/health/news/11760658/One-in-four-self-diagnose-on-the-internet-instead-of-visiting-the-doctor.html. [Accessed 2018 Mei 27]. |
| [2] |  | ecisapare, "Quizlet," 2014. [Online]. Available: https://quizlet.com/34170376/medical-terminology-medical-terms-for-disease-diagnosis-treatment-flash-cards/. [Accessed 1 January 2018]. |
| [3] |  | Unknown, "Medical Bits," 5 Januari 2014. [Online]. Available: https://medicalbits.wordpress.com/2014/01/05/the-anamnesis-taking-a-medical-history/. [Accessed 8 Januari 2018]. |
| [4] |  | A. Maae, "Quora," Quora, 25 April 2017. [Online]. Available: https://www.quora.com/What-is-support-and-confidence-in-data-mining/answer/Azim-Maae?srid=3QjiQ. [Accessed 1 Januari 2018]. |
| [5] |  | Unknown, "WikiBooks," 6 Agustus 2017. [Online]. Available: https://en.wikibooks.org/wiki/Data\_Mining\_Algorithms\_In\_R/Frequent\_Pattern\_Mining/The\_FP-Growth\_Algorithm. [Accessed 8 Januari 2018]. |
| [6] |  | F. Verhein, "wimleers.com," 10 Januari 2008. [Online]. Available: https://wimleers.com/sites/wimleers.com/files/FP-Growth%20presentation%20handouts%20%E2%80%94%C2%A0Florian%20Verhein.pdf. [Accessed 30 Juni 2018]. |
| [7] |  | N. Sadawi, "Generating Association Rules from Frequent Itemsets," 28 Agustus 2014. [Online]. Available: https://www.youtube.com/watch?v=-5Uia2a5jxc. [Accessed 30 Juni 2018]. |
| [8] |  | M. S. S. S. Tina R. Patil, "Performance Analysis of Naive Bayes and J48," *International Journal Of Computer Science And Applications,* vol. 6, p. 2, 2013. |
| [9] |  | B. Stecanella, "MonkeyLearn," MonkeyLearn Inc., 25 Mei 2017. [Online]. Available: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/. [Accessed 30 Mei 2018]. |
| [10] |  | Geeks for Geeks, "Geeks for Geeks," Geeks for Geeks, [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-classifiers/. [Accessed 30 June 2018]. |
| [11] |  | R. Saxena, "Dataaspirant," Dataaspirant, 6 Februari 2017. [Online]. Available: http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/. [Accessed 2018 Juni 30]. |
| [12] |  | S. Patel, "Medium," Medium Inc, 3 Mei 2017. [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72. [Accessed 30 Mei 2018]. |
| [13] |  | DF Team, "Data Flair," Data Flair, 12 Agustus 2017. [Online]. Available: https://data-flair.training/blogs/svm-kernel-functions/. [Accessed 2018 Juni 30]. |
| [14] |  | N. Twomey, "Quora," Quora, 7 April 2015. [Online]. Available: https://www.quora.com/Why-am-I-obtaining-better-results-with-a-linear-kernel-for-a-test-classification-task-with-SVM. [Accessed 2018 Juni 30]. |
| [15] |  | S. Singh, "Sadanand Notes," Github, 8 Juli 2017. [Online]. Available: https://sadanand-singh.github.io/posts/svmpython/. [Accessed 30 Juni 2018]. |
| [16] |  | Unknown, "ScikitLearn - Wikipedia," Wikipedia, 16 April 2018. [Online]. Available: https://en.wikipedia.org/wiki/Scikit-learn. [Accessed 30 Mei 2018]. |
| [17] |  | A. Ronacher, "Flask," - - 2010-2018. [Online]. Available: http://flask.pocoo.org/. [Accessed 30 Mei 2018]. |
| [18] |  | A. Ronacher, "Flask," Flask, 2013. [Online]. Available: https://web.archive.org/web/20171117015927/http://flask.pocoo.org:80/docs/0.10/foreword. [Accessed 30 Mei 2018]. |
| [19] |  | Laravel, "Github," Github Inc, 19 Mei 2018. [Online]. Available: https://github.com/laravel/framework. [Accessed 30 Mei 2018]. |
| [20] |  | J. P. a. Y. Y. Jiawei Han, "Mining Frequent Patterns without Candidate Generation". |
| [21] |  | B. Stecanella, "MonkeyLearn," MonkeyLearn Inc, 22 Juni 2017. [Online]. Available: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/. [Accessed 30 Mei 2018]. |

***[Halaman ini sengaja dikosongkan]***

# LAMPIRAN

## Hasil Lengkap Uji Sistem

|  |  |  |
| --- | --- | --- |
| TRUE | Prediksi |  |
| rhinitis | rhinitis | ✓ |
| rhinitis | rhinitis | ✓ |
| rhinitis | rhinitis | ✓ |
| influenza | influenza | ✓ |
| influenza | influenza | ✓ |
| influenza | influenza | ✓ |
| faringitis | influenza |  |
| faringitis | influenza |  |
| faringitis | leptospirosis |  |
| tonsilitis | tonsilitis | ✓ |
| tonsilitis | tonsilitis | ✓ |
| tonsilitis | tonsilitis | ✓ |
| laringitis | laringitis | ✓ |
| laringitis | laringitis | ✓ |
| laringitis | laringitis | ✓ |
| pneumonia | tuberkulosis |  |
| pneumonia | pneumonia | ✓ |
| pneumonia | pneumonia | ✓ |
| bronkopneumonia | bronkopneumonia | ✓ |
| bronkopneumonia | pneumotoraks |  |
| bronkopneumonia | pneumonia |  |

|  |  |  |
| --- | --- | --- |
| TRUE | Prediksi |  |
| pneumotoraks | pneumotoraks | ✓ |
| pneumotoraks | pneumotoraks | ✓ |
| pneumotoraks | pneumotoraks | ✓ |
| ppok-penyakit-paru-obstruktif-kronis | ppok-penyakit-paru-obstruktif-kronis | ✓ |
| ppok-penyakit-paru-obstruktif-kronis | ppok-penyakit-paru-obstruktif-kronis | ✓ |
| ppok-penyakit-paru-obstruktif-kronis | pneumotoraks |  |
| epistaksis | epistaksis | ✓ |
| epistaksis | epistaksis | ✓ |
| epistaksis | epistaksis | ✓ |
| sinusitis | sinusitis | ✓ |
| sinusitis | sinusitis | ✓ |
| sinusitis | sinusitis | ✓ |
| tuberkulosis | tuberkulosis | ✓ |
| tuberkulosis | tuberkulosis | ✓ |
| tuberkulosis | tuberkulosis | ✓ |
| morbili | morbili | ✓ |
| morbili | morbili | ✓ |
| morbili | morbili | ✓ |
| varisela | varisela | ✓ |
| varisela | varisela | ✓ |
| varisela | varisela | ✓ |
| malaria | malaria | ✓ |
| malaria | malaria | ✓ |
| malaria | malaria | ✓ |
| TRUE | Prediksi |  |
| leptospirosis | leptospirosis | ✓ |
| leptospirosis | leptospirosis | ✓ |
| leptospirosis | leptospirosis | ✓ |
| filariasis | filariasis | ✓ |
| filariasis | filariasis | ✓ |
| filariasis | filariasis | ✓ |
| lepra | perdarahan-subkonjungtiva |  |
| lepra | lepra | ✓ |
| lepra | lepra | ✓ |
| lepra | lepra | ✓ |
| keracunan-makanan | keracunan-makanan | ✓ |
| keracunan-makanan | keracunan-makanan | ✓ |
| keracunan-makanan | kolesistitis |  |
| alergi-makanan | alergi-makanan | ✓ |
| alergi-makanan | ankilostomiasis |  |
| alergi-makanan | perdarahan-subkonjungtiva |  |
| demam-berdarah-dengue | leptospirosis |  |
| demam-berdarah-dengue | demam-berdarah-dengue | ✓ |
| demam-berdarah-dengue | kolesistitis |  |
| anemia | anemia | ✓ |
| anemia | anemia | ✓ |
| anemia | anemia | ✓ |
| hiv-aids | hiv-aids | ✓ |
| hiv-aids | lupus-eritematosus-sistemik |  |
| TRUE | Prediksi |  |
| hiv-aids | keracunan-makanan |  |
| lupus-eritematosus-sistemik | lupus-eritematosus-sistemik | ✓ |
| lupus-eritematosus-sistemik | lupus-eritematosus-sistemik | ✓ |
| lupus-eritematosus-sistemik | lupus-eritematosus-sistemik | ✓ |
| limfadenitis | limfadenitis | ✓ |
| limfadenitis | peritonitis |  |
| limfadenitis | limfadenitis | ✓ |
| asam-lambung | asam-lambung | ✓ |
| asam-lambung | asam-lambung | ✓ |
| asam-lambung | asam-lambung | ✓ |
| hipertensi | pneumotoraks |  |
| hipertensi | hipertensi | ✓ |
| hipertensi | takikardia |  |
| cardiorespiratory-arrest | cardiorespiratory-arrest | ✓ |
| cardiorespiratory-arrest | cardiorespiratory-arrest | ✓ |
| cardiorespiratory-arrest | cardiorespiratory-arrest | ✓ |
| gagal-jantung | gagal-jantung | ✓ |
| gagal-jantung | gagal-jantung | ✓ |
| gagal-jantung | ppok-penyakit-paru-obstruktif-kronis |  |
| takikardia | takikardia | ✓ |
| gastritis | kolesistitis |  |
| gastritis | gastritis | ✓ |
| takikardia | infark-miokard |  |
| TRUE | Prediksi |  |
| gastritis | gastritis | ✓ |
| takikardia | takikardia | ✓ |
| infark-miokard | infark-miokard | ✓ |
| gastroenteritis | malaria |  |
| gastroenteritis | keracunan-makanan |  |
| gastroenteritis | keracunan-makanan |  |
| disentri | disentri | ✓ |
| disentri | disentri | ✓ |
| disentri | disentri | ✓ |
| hemoroid | hemoroid | ✓ |
| infark-miokard | infark-miokard | ✓ |
| hemoroid | hemoroid | ✓ |
| hemoroid | hemoroid | ✓ |
| infark-miokard | angina-pektoris-stabil |  |
| angina-pektoris-stabil | infark-miokard |  |
| hepatitis-a | hepatitis-b |  |
| hepatitis-a | hepatitis-a | ✓ |
| hepatitis-a | hepatitis-b |  |
| angina-pektoris-stabil | angina-pektoris-stabil | ✓ |
| angina-pektoris-stabil | angina-pektoris-stabil | ✓ |
| hepatitis-b | hepatitis-b | ✓ |
| hepatitis-b | hepatitis-a |  |
| hepatitis-b | hepatitis-b | ✓ |
| kolesistitis | kolesistitis | ✓ |
| kolesistitis | kolesistitis | ✓ |
| kolesistitis | kolesistitis | ✓ |
| TRUE | Prediksi |  |
| apendisitis | peritonitis |  |
| apendisitis | peritonitis |  |
| apendisitis | peritonitis |  |
| peritonitis | peritonitis | ✓ |
| peritonitis | peritonitis | ✓ |
| peritonitis | peritonitis | ✓ |
| serumen-prop | serumen-prop | ✓ |
| serumen-prop | serumen-prop | ✓ |
| serumen-prop | serumen-prop | ✓ |
| benda-asing-di-telinga | benda-asing-di-telinga | ✓ |
| benda-asing-di-telinga | benda-asing-di-telinga | ✓ |
| benda-asing-di-telinga | benda-asing-di-telinga | ✓ |
| askariasis | askariasis | ✓ |
| askariasis | askariasis | ✓ |
| askariasis | askariasis | ✓ |
| ankilostomiasis | ankilostomiasis | ✓ |
| ankilostomiasis | ankilostomiasis | ✓ |
| ankilostomiasis | ankilostomiasis | ✓ |
| skistosomiasis | skistosomiasis | ✓ |
| skistosomiasis | skistosomiasis | ✓ |
| skistosomiasis | skistosomiasis | ✓ |
| taeniasis | taeniasis | ✓ |
| taeniasis | taeniasis | ✓ |
| taeniasis | taeniasis | ✓ |
| strongiloidiasis | strongiloidiasis | ✓ |
| strongiloidiasis | strongiloidiasis | ✓ |
| TRUE | Prediksi |  |
| strongiloidiasis | strongiloidiasis | ✓ |
| mata-kering | trikiasis |  |
| mata-kering | perdarahan-subkonjungtiva |  |
| mata-kering | konjungtivitis |  |
| buta-senja | buta-senja | ✓ |
| buta-senja | buta-senja | ✓ |
| buta-senja | buta-senja | ✓ |
| hordeolum-bintitan | hordeolum-bintitan | ✓ |
| hordeolum-bintitan | hordeolum-bintitan | ✓ |
| hordeolum-bintitan | hordeolum-bintitan | ✓ |
| konjungtivitis | konjungtivitis | ✓ |
| konjungtivitis | hordeolum-bintitan |  |
| konjungtivitis | konjungtivitis | ✓ |
| blefaritis | konjungtivitis |  |
| blefaritis | blefaritis | ✓ |
| blefaritis | blefaritis | ✓ |
| perdarahan-subkonjungtiva | trikiasis |  |
| perdarahan-subkonjungtiva | perdarahan-subkonjungtiva | ✓ |
| perdarahan-subkonjungtiva | perdarahan-subkonjungtiva | ✓ |
| astigmatisme | astigmatisme | ✓ |
| astigmatisme | astigmatisme | ✓ |
| astigmatisme | astigmatisme | ✓ |
| hipermetropia | astigmatisme |  |
| TRUE | Prediksi |  |
| hipermetropia | hipermetropia | ✓ |
| hipermetropia | hipermetropia | ✓ |
| miopia-ringan | miopia-ringan | ✓ |
| miopia-ringan | miopia-ringan | ✓ |
| miopia-ringan | miopia-ringan | ✓ |
| presbiopia | astigmatisme |  |
| presbiopia | astigmatisme |  |
| presbiopia | presbiopia | ✓ |
| katarak | presbiopia |  |
| katarak | retinopati-diabetik |  |
| katarak | katarak | ✓ |
| glaukoma | glaukoma | ✓ |
| glaukoma | glaukoma | ✓ |
| glaukoma | glaukoma | ✓ |
| trikiasis | trikiasis | ✓ |
| trikiasis | blefaritis |  |
| trikiasis | episkleritis |  |
| episkleritis | episkleritis | ✓ |
| episkleritis | episkleritis | ✓ |
| episkleritis | episkleritis | ✓ |
| hifema | hifema | ✓ |
| hifema | hifema | ✓ |
| hifema | hifema | ✓ |
| retinopati-diabetik | retinopati-diabetik | ✓ |
| retinopati-diabetik | retinopati-diabetik | ✓ |
| retinopati-diabetik | hifema |  |
| TRUE | Prediksi |  |
| otitis-eksterna | otitis-eksterna | ✓ |
| otitis-eksterna | otitis-eksterna | ✓ |
| otitis-eksterna | otitis-eksterna | ✓ |
| otitis-media | otitis-media | ✓ |
| otitis-media | otitis-media | ✓ |
| otitis-media | otitis-eksterna |  |

## 2. Hasil Lengkap Uji Dokter

### Uji Coba 1

Informasi Pasien

Jenis Kelamin : Laki - laki

Usia : 22 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Sakit perut | Ya (jawaban awal) |
| Muntah | Ya |
| Mual | Ya |
| Demam | Tidak |
| Diare | Tidak |
| Nafsu makan berkurang | Tidak |
| Malaise | Tidak |
| Kepala sakit | Ya |
| Demam tinggi | Tidak |
| Otot nyeri | Tidak |
| Buang air besar berdarah | Tidak |
| Perut kram | Ya |
| Kulit bintik bintik merah | Tidak |
| Perdarah gusi | Tidak |
| Kulit ruam | Tidak |
| **Gejala** | **Jawaban** |
| Batuk | Ya |
| Batuk kering | Tidak |
| Batuk berdahak | Tidak |
| Tenggorokan Sakit | Ya |
| Hidung mengeluarkan ingus | Ya |
| Bersin | Ya |

Hasil diagnosis :

* faringitis· 92.24%
* influenza· 3.00%
* rhinitis· 2.14%
* leptospirosis· 0.83%
* ankilostomiasis· 0.41%

### Uji Coba 2

Informasi Pasien

Jenis Kelamin : Laki - laki

Usia : 62 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Penglihatan Kabur | Ya (jawaban awal) |
| Kepala Sakit/Nyeri | Tidak |
| memincingkan mata untuk melihat lebih jelas | Ya |
| Penglihatan kabur jarak jauh | Ya |
| Mata lelah | Ya |
| Pusing | Tidak |
| Mata perih mata panas | Tidak |
| penglihatan kabur jarak dekat | Ya |
| Mata kering | Tidak |
| Mata gatal | Tidak |
| **Gejala** | **Jawaban** |
| Fotofobia | Tidak |
| kesulitan membaca | Ya |
| Mata merah | Tidak |

Hasil diagnosis :

* miopia-ringan· 52.41%
* hipermetropia· 45.48%
* astigmatisme· 1.91%
* presbiopia· 0.21%
* retinopati-diabetik· 0.00%

### Uji Coba 3

Informasi Pasien

Jenis Kelamin : Perempuan

Usia : 22 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Sakit Perut | Ya (jawaban awal) |
| Muntah | Tidak |
| Mual | Tidak |
| Diare | Tidak |
| Demam | Tidak |
| Nafsu Makan Berkurang | Tidak |
| Malaise | Ya |
| Penurunan berat badan | Tidak |
| Ikterus | Tidak |
| Menggigil | Tidak |
| Batuk | Tidak |
| Kepala Sakit/Nyeri | Tidak |
| Buang air besar berdarah | Ya |
| Urine berdarah | Tidak |
| kulit ruam | Tidak |
| **Gejala** | **Jawaban** |
| Kulit gatal | Tidak |
| Perut kram | Ya |
| Buang air besar berulang | Ya |
| Demam Tinggi | Tidak |
| Kembung | Ya |
| Nyeri tekan perut | Ya |

Hasil diagnosis :

* gastroenteritis· 71.45%
* disentri· 25.36%
* taeniasis· 0.67%
* peritonitis· 0.53%
* skistosomiasis· 0.51%

### Uji Coba 4

Informasi Pasien

Jenis Kelamin : Laki laki

Usia : 34 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Sesak Dada | Ya |
| Dada nyeri | Ya |
| Sesak nafas | Ya |
| Nadi cepat jantung cepat | Ya |
| takipnea | Ya |
| mual | Tidak |
| Dada terasa berat | Ya |
| Batuk | Tidak |
| Malaise | Tidak |
| Demam | Tidak |
| Batuk berdahak | Tidak |
| Pusing | Tidak |
| **Gejala** | **Jawaban** |
| Kepala sakit/nyeri | Tidak |
| Palpitasi | Ya |
| Kulit Ruam | Tidak |
| Tenggorokan Sakit Nyeri | Tidak |
| Merokok | Ya |
| Obesitas | Tidak |
| Mengi | Ya |
| Riwayat gangguan jantung | Ya |
| Batuk kronis | Tidak |

Hasil diagnosis :

* pneumotoraks· 83.33%
* infark-miokard· 12.60%
* gagal-jantung· 2.80%
* angina-pektoris-stabil· 0.64%
* takikardia· 0.60%

### Uji Coba 5

Informasi Pasien

Jenis Kelamin : Perempuan

Usia : 15 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Hidung tersumbat | Ya |
| Hidung mengeluarkan ingus (pilek) | Ya |
| Bersin | Ya |
| Hidung gatal | Tidak |
| Kepala sakit/nyeri | Tidak |
| Batuk | Ya |
| Batuk kering | Tidak |
| Batuk berdahak | Ya |
| Demam | Ya |
| **Gejala** | **Jawaban** |
| Demam Tinggi | Tidak |
| Demam > 3 hari | Tidak |
| Tenggorokan sakit/nyeri | Ya |
| Malaise | Ya |
| Wajah nyeri | Ya |
| Kulit bintik bintik merah | Tidak |
| Ingus berwarna hijau | Tidak |
| Otot nyeri | Tidak |
| Nafas Bau | Tidak |
| Sesak nafas | Tidak |
| Nafsu makan berkurang | Ya |
| Mengigil | Tidak |

Hasil diagnosis :

* influenza· 69.48%
* rhinitis· 20.35%
* faringitis· 4.81%
* sinusitis· 3.79%
* bronkopneumonia· 1.03%

### Uji Coba 6

Informasi Pasien

Jenis Kelamin : Laki laki

Usia : 34 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Suara serak | Ya |
| Tenggorokan Sakit/Nyeri | Ya |
| Batuk | Tidak |
| Demam | Tidak |
| Suara hilang | Ya |
| Batuk kering | Tidak |
| **Gejala** | **Jawaban** |
| Suara parau | Ya |
| Kepala sakit/nyeri | Tidak |
| Malaise | Tidak |
| Hidung mengeluarkan ingus | Tidak |
| Tenggorokan Sakit/Nyeri saat menelan | Ya |
| Otot nyeri | Tidak |
| Nafsu makan berkurang | Tidak |
| Mual | Tidak |
| Demam tinggi | Tidak |
| Sendi nyeri | Tidak |
| Bersin | TIdak |
| Diare | Tidak |
| Sesak nafas | Tidak |
| Muntah | Tidak |
| Kulit ruam | Tidak |

Hasil diagnosis :

* laringitis· 99.35%
* asam-lambung· 0.41%
* tonsilitis· 0.16%
* perdarahan-subkonjungtiva· 0.02%
* faringitis· 0.02%

### Uji Coba 7

Informasi Pasien

Jenis Kelamin : Laki laki

Usia : 35 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Demam tinggi | Ya |
| Demam | Ya |
| Demam > 3 hari | Ya |
| **Gejala** | **Jawaban** |
| Kepala sakit/nyeri | Ya |
| Muntah | Tidak |
| Mual | TIdak |
| Otot Nyeri | Tidak |
| Tenggorokan Sakit/Nyeri | Tidak |
| Diare | Tidak |
| Malaise | Ya |
| Tenggorokan Sakit/Nyeri saat menelan | Tidak |
| Batuk | Ya |
| Batuk Kering | Ya |
| Hidung mengeluarkan ingus (pilek) | Tidak |
| Sesak nafas | Ya |
| Batuk berdahak | Ya |
| Nafsu makan berkurang | Ya |
| Mengigil | Tidak |
| Dada nyeri | Tidak |
| Napas Pendek dan terengah engah (napas cepat) | Tidak |
| Penurunan berat badan | Tidak |

Hasil diagnosis :

* pneumonia· 59.65%
* bronkopneumonia· 35.01%
* influenza· 1.82%
* tuberkulosis· 1.60%
* ppok-penyakit-paru-obstruktif-kronis· 1.51%

### Uji Coba 8

Informasi Pasien

Jenis Kelamin : Perempuan

Usia : 22 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Pingsan mendadak | Ya |
| Tidak ada nafas | Ya |
| Pusing | Tidak |
| Henti Jantung | Ya |
| Tidak sadar | Ya |
| Dada nyeri | Ya |
| Palpitasi (Denyut Detak Jantung Tidak Beraturan) | Ya |
| Takipnea | Ya |
| Malaise | Tidak |
| Sesak nafas | Ya |
| Nadi cepat jantung cepat | Ya |
| Sesak dada | Ya |
| Mual | Tidak |
| Dada terasa berat | Tidak |
| Dada nyeri menjalar ke punggung, lengan, bahu | Ya |

Hasil diagnosis :

* cardiorespiratory-arrest· 98.17%
* pneumotoraks· 1.34%
* infark-miokard· 0.28%
* takikardia· 0.13%
* angina-pektoris-stabil· 0.08%

### Uji Coba 9

Perempuan, 57 tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Mata merah | Ya |
| Mata berair | Ya |
| Fotofobia | Ya |
| Mata gatal | Ya |
| **Gejala** | **Jawaban** |
| Mata kering | Tidak |
| Mata perih | Tidak |
| Mata bengkak | Tidak |
| Memiliki riwayat alergi | Tidak |
| Kelopak mata bengkak | Tidak |
| Kulit gatal | Tidak |
| Bersin | Tidak |
| Mata nyeri | Ya |
| Bulu mata lengket | Tidak |
| Mata sensasi berpasir | Tidak |
| Mata lelah | Tidak |
| Hidung mengeluarkan ingus (pilek) | Tidak |
| Kulit ruam | Tidak |
| Sesak nafas | Tidak |
| Mata muncul kotoran di sekeliling kelopak mata (belek) | Ya |
| Takipnea | Tidak |
| Hidung tersumbat | Tidak |

Hasil diagnosis :

* konjungtivitis· 86.16%
* episkleritis· 8.29%
* perdarahan-subkonjungtiva· 2.19%
* trikiasis· 2.01%
* hordeolum-bintitan· 0.85%

### Uji Coba 10

Informasi Pasien

Jenis Kelamin : Laki laki

Usia : 67 Tahun

|  |  |
| --- | --- |
| **Gejala** | **Jawaban** |
| Kulit ruam | Ya |
| Demam | Ya |
| Demam tinggi | Tidak |
| Demam > 3 hari | Tidak |
| Malaise | Tidak |
| Diare | Tidak |
| Otot nyeri | Tidak |
| Kepala sakit/nyeri | Tidak |
| Batuk | Tidak |
| Tenggorokan sakit | Tidak |
| Penurunan berat badan | Tidak |
| Sendi nyeri | Tidak |
| Kulit bintik merah | Ya |
| Hidung mengeluarkan ingus | Tidak |
| Kulit memerah | Ya |
| Mual | Tidak |
| Muntah | Tidak |
| nafsu makan berkurang | Tidak |
| Sakit perut | Tidak |
| Mengigil | Tidak |
| Sesak nafas | Tidak |

Hasil diagnosis :

* morbili· 69.75%
* lupus-eritematosus-sistemik· 27.99%
* filariasis· 1.01%
* lepra· 0.39%
* ankilostomiasis· 0.22%

***[Halaman ini sengaja dikosongkan]***

**BIODATA PENULIS**

Kevin Alif Fachreza lahir di Magetan pada 9 Maret 1997 menghabiskan masa kecil di Tangerang Selatan dan masa remaja di Surabaya. Penulis menempuh pendidikan sekolah dasar di SD Tirta Buaran Tangerang (2002 – 2007) dan SD Dewi Sartika Surabaya (2007 - 2008), lalu di SMPN 1 Surabaya (2008 – 2011) terakhir di SMAN 9 Surabaya (2011 – 2014). Setelah menempuh pendidikan SMA penulis melanjutkan kuliah di Teknik Informatika ITS pada 2014 – 2018. Selama menempuh kuliah penulis aktif pada organisasi himpunan jurusan.

Pada 2017 penulis membuat aplikasi Medify yang digunakan oleh banyak dokter dan pada tahun yang sama penulis mendirikan perusahaan PT Global Medika Digitama, dan memiliki jabatan sebagai *Chief Executive Officer*. Penulis dapat dihubungi melalui *email* kevin@medify.id