Aplikasi Anamnesis Berdasarkan Gejala Menggunakan *Frequent Pattern Tree Growth*

Kevin Alif Fachreza, Chastine Fathichah, Anny Yuniarti
Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: kevin.fachreza14@mhs.if.its.ac.id[1)], chastine@cs.its.ac.id[2)], anny@if.its.ac.id[3)]

***Abstrak*—Dewasa ini ketika pasien sedang sakit, pasien akan pergi ke dokter untuk mendapatkan diagnosis. Dalam melakukan diagnosis dokter akan melakukan proses anamnesis, proses tanya jawab gejala yang dialami pasien untuk mendapatkan diagnosis yangn dialami pasien. Kini ada banyak aplikasi yang dapat memberikan diagnosis, akan tetapi tidak ada proses anamnesis pada aplikasi aplikasi tersebut. Pada penelitian ini aplikasi akan difokuskan pada proses anamnesis, sehingga diharapkan akan memberikan diagnosis kepada pasien secara akurat. Proses anamnesis pada aplikasi akan menggunakan *Frequent Pattern Tree Growth*, yaitu salah satu metode dari *association rules*. Sedangkan pada proses diagnosis akan digunakan algoritma *Naïve Bayes* dan *Support Vector Machine*. Aplikasi diujikan secara sistem dengan metode akurasi, *retrieval* dan peringkat. Selain itu juga aplikasi diujikan kepada masyarakat awam dan juga dokter. Hasilnya aplikasi memberikan hasil yang baik, dengan akurasi rata rata 69% dengan *Naïve Bayes* dan 67% dengan *Support Vector Machine*. Dokter penguji juga memberikan umpan balik bahwa aplikasi memiliki proses anamnesis yang baik dan memberikan diagnosis yang sebagian besar sesuai.**

***Kata Kunci*—*Frequent Pattern Tree Growth*, *Support Vector Machine*, *Naïve Bayes*, anamnesis, diagnosis.**

I. PENDAHULUAN

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.

Solusi aplikasi tersebut akan meniru proses dokter untuk mendapatkan diagnosis. Pada ilmu kedokteran, untuk mendapatkan diagnosis, seorang dokter harus melakukan proses anamnesis. Anamnesis adalah proses akumulasi data yang menyangkut data medis pasien, latar belakang pasien, termasuk keluarga, lingkungan, pengalaman, terutama ingatan untuk digunakan dalam menganalisa kondisi [2]. Pasien yang melakukan diagnosis dengan bantuan *search engine* sebenarnya sudah melakukan tahap tahap anamnesis. Akan tetapi pasien tidak memiliki ilmu kedokteran sehingga kemampuan anamnesisnya diragukan.

Aplikasi yang dapat memberikan diagnosis kepada pasien sebenarnya sudah pernah dibuat pada penelitian penelitian sebelumnya. Akan tetapi kebanyakan aplikasi tersebut tidak memiliki proses anamnesis untuk melakukan diagnosis. Sehingga proses diagnosis tidak sesuai dengan ilmu kedokteran.

Pada penelitian kali ini, akan difokuskan pada proses anamnesis sehingga aplikasi dapat memberikan hasil diagnosis yang sesuai. Aplikasi akan menanyakan gejala yang mungkin dialami pasien sebagai proses anamnesis. Pertanyaan gejala akan berkaitan dengan gejala yang ditanyakan sebelumnya. Agar pertanyaan dapat saling berkaitan akan digunakan algoritma *Frequent Pattern Tree Growth*.

Frequent Pattern Tree Growth merupakan algoritma *association rules* yang akan melakukan kalkulasi mengenai keterkaitan antar atribut. Dengan adanya nilai keterkaitan atribut mesin dapat memberikan pertanyaan gejala yang sesuai dengan jawaban pertanyaan sebelumnya.

Sebagai tambahan, aplikasi ini akan memberikan hasil akhir berupa diagnosis. Pada tahap diagnosis digunakan algoritma *Naïve Bayes* dan *Support Vector Machine*.

II. URAIAN PENELITIAN

A. Survei Literatur

1) Anamnesis

Anamnesis adalah proses akumulasi data yang menyangkut data medis pasien, latar belakang pasien, termasuk keluarga, lingkungan, pengalaman, terutama ingatan untuk digunakan dalam menganalisa kondisi [2]. Anamnesis bertujuan untuk mengekstrak informasi

pasien agar dokter dapat memberikan diagnosis yang akurat. Adapun dalam melakukan anamnesis dokter umumnya melakukan langkah langkah berikut [3]:

1. Pasien memberikan gejala yang paling dirasakan
2. Dokter menanyakan gejala yang disebutkan pasien untuk mendapatkan informasi lebih dalam.
3. Menanyakan gejala lain yang mungkin dialami pasien.
4. Menanyakan obat atau tindakan yang telah dilakukan.
5. Menanyakan informasi kesehatan keluarga.
6. Menanyakan lingkungan keseharian.
7. Menanyakan informasi lain terkait sistem tubuh lain yang tidak tercakup pada gejala.
8. Mengulas ulang keluhan.
9. Dokter memberikan diagnosis.

2) 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.

Adapun berikut adalah algoritma dari *FP Tree:*

Input : *Dataset* dan *minimum support*
Output : *Rules*

Tahap 1 Pembuatan *Tree*

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.

3) Naive 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 *Bayes Theorem* yang sebagai berikut [9] :

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \quad (1)$$

dengan A dan B adalah atribut, P(A) adalah peluang A, P(B) adalah peluang B, P(A|B) adalah peluang A terhadap B, dan peluang P(B|A) adalah peluang B terhadap A.

Ada beberapa jenis *Naïve Bayes* yang dapat digunakan untuk jenis *dataset* yang berbeda beda. *Gaussian Naïve Bayes* digunakan untuk dataset yang 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 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 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].

4) Support Vector Machine

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

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 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*.

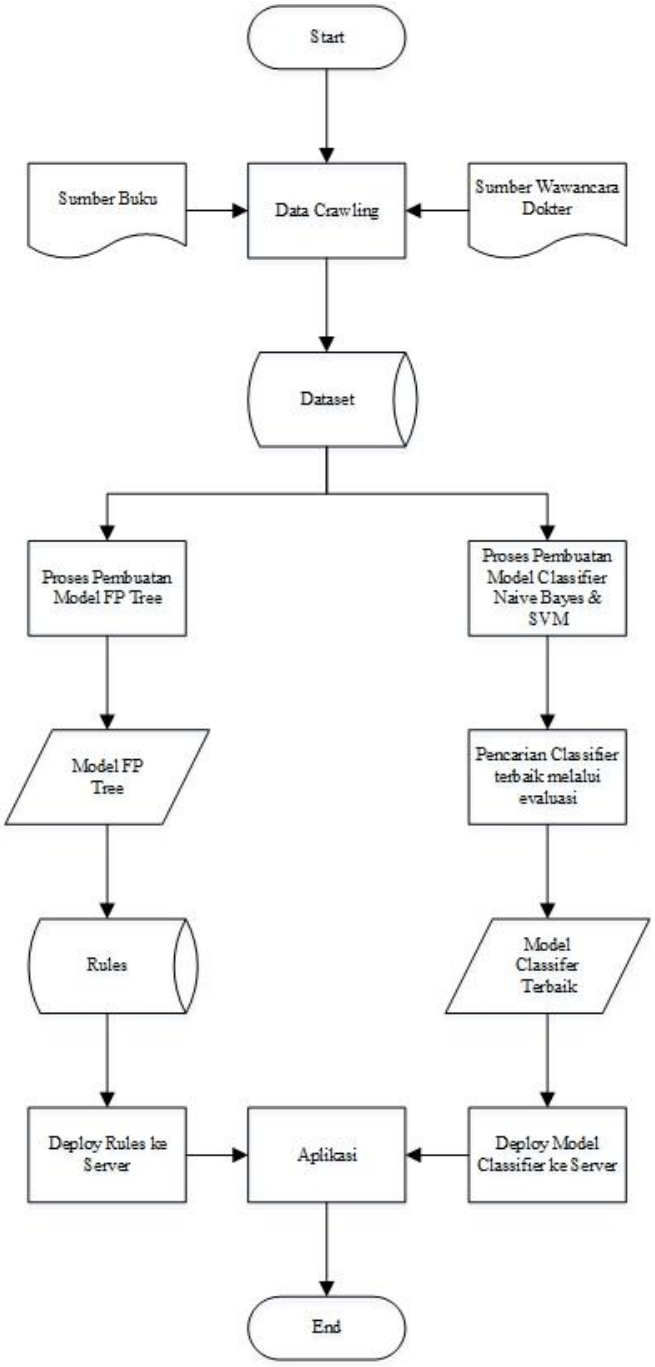
Pada *One vs All* data akan di training satu per satu. Setiap kelas yang di training akan dilabeli sebagai kelas yang bernilai positif sedangkan kelas lain bernilai negatif. Walaupun secara komputasi pendekatan ini lebih cepat tetapi jika data tidak seimbang, dapat menyebabkan performa yang kurang baik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

Pada *One vs One* 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].

B. Perancangan

1) Perancangan Data

Data berasal dari buku (PPK Kedokteran, Diagnosis Banding Kedokteran, dan Kapita Selekta Kedokteran) dan wawancara dengan dokter akan dimasukkan ke dalam *database*. Dataset 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.



Gambar 1. Alur Penelitian Secara Umum

2) Perancangan Penelitian Secara Umum

Aplikasi akan dibuat dalam beberapa tahap yang dapat dilihat pada Gambar 1. Tahap tersebut adalah 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) Perancangan Model FP Tree

Pada tahap model *FP Tree* akan dibuat sesuai dengan algoritma *FP Tree* yang ada pada bagian sebelumnya.

Hasil dari proses ini adalah *rules*. *Rules* akan disimpan pada sebuah tabel pada *database* yang memiliki atribut *id, rules input, rules output* dan *probability*.

4) Perancangan Model 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 ScikitLearn*. Nantinya model ini akan diujikan dengan data testing yang telah ditetapkan sebelumnya.

5) Perancangan Model SVM

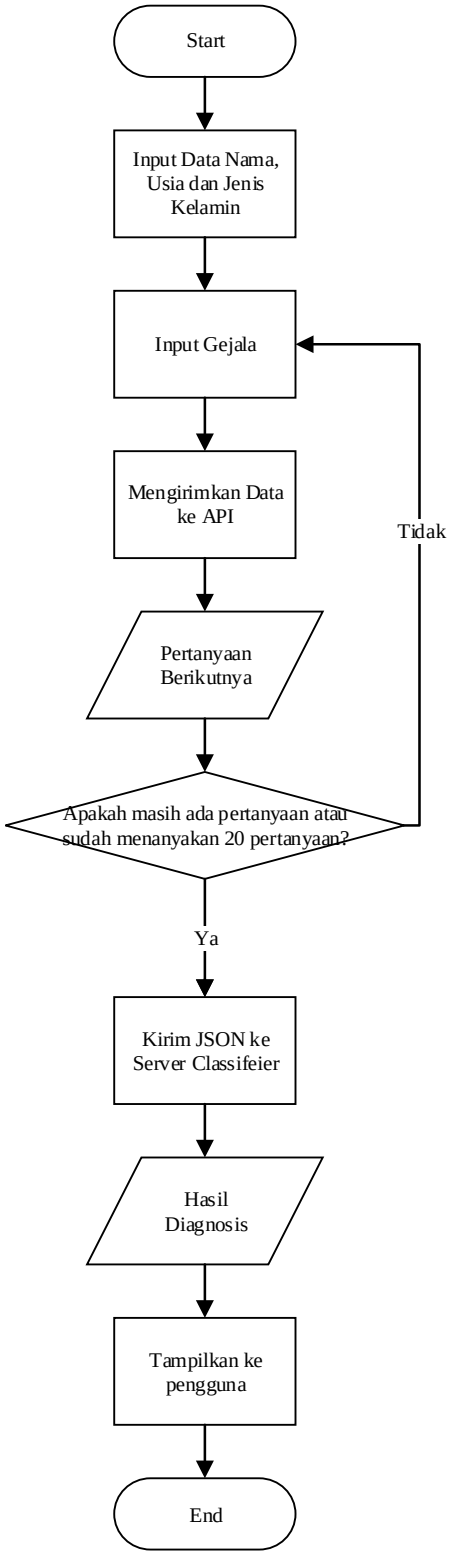
Pada tahap ini akan dibuat model *SVM* dengan *kernel linear*. Pada *Scikit Learn* lebih dikenal dengan *SVC*. Sama seperti *Naïve Bayes*, model *SVM* juga akan dibuat berdasarkan data belajar yang telah di filter berdasarkan *minimum support*. Model ini dibuat dengan library *Scikit Learn*.

6) Perancangan Alur Aplikasi

Alur aplikasi dapat dilihat pada **Error! Reference source not found.**. 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. Selanjutnya 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 sejumlah batas pertanyaan pada aplikasi atau *server* tidak memiliki pertanyaan lain untuk ditanyakan.

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.



Gambar 2. Perancangan 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.

Jika aplikasi sudah selesai menanyakan pertanyaan, maka aplikasi akan mengirimkan data gejala yang dialami pasien ke *server* diagnosis. Setelah itu *server* akan mengirimkan kembali hasil data ke aplikasi. Dan aplikasi akan menampilkan 5 kemungkinan diagnosis tertinggi beserta persentase.

C. Metode Evaluasi

Aplikasi dievaluasi secara sistem menggunakan akurasi dan juga *retrieval*. Metrik akurasi akan didasarkan pada hasil klasifikasi dengan probabilitas klasifikasi tertinggi dari model *classifier* dengan hasil yang seharusnya. Dengan rumus sebagai berikut

$$Akurasi = \frac{TP + TN}{TP+TN + FP + FN} \quad (2)$$

dimana *TP* adalah *True Positive*, *TN* adalah *True Negative*, *FP* adalah *False Positive*, dan *FN* adalah *False Negative*.

Sedangkan pada evaluasi *retrieval* aplikasi akan dinilai berdasarkan hasil pada data testing 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

$$Skor = (5 - \{Peringkat - 1\}) * 20 \quad (3)$$

dengan peringkat adalah 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.

Pada evaluasi pengguna aplikasi akan diuji oleh masyarakat untuk mengetahui berapa pertanyaan gejala yang harus ditanyakan aplikasi agar pengguna tidak bosan. Evaluasi juga akan dilakukan dokter, yang mana dokter akan menilai aplikasi setelah melakukan beberapa kali percobaan secara kualitatif.

III. HASIL UJICOBA

A. Uji Sistem

Pada uji coba sistem menggunakan akurasi, aplikasi memberikan hasil yang cukup baik. Pada Tabel 1 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. 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%.

Tabel 1.
Hasil Ujicoba Sistem dengan Metrik 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**

Berdasarkan hasil uji *retrieval* pada Tabel 2 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.

Tabel 2.
Hasil Ujicoba Sistem dengan Metrik *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 peringkat nilai pada Tabel 3 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* bernilai 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 3.
Hasil Ujicoba Sistem dengan Metrik 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

B. Uji Pengguna Masyarakat Awam

Pada uji pengguna, dilakukan oleh 10 orang dengan usia 20 – 25 tahun. Pengguna akan mencoba aplikasi dengan Batasan pertanyaan gejala sebanyak 30. Ketika proses ujicoba, jika pengguna merasa bosan dengan pertanyaan gejala, maka pengguna dapat berhenti menggunakan aplikasi dan akan dicatat pada pertanyaan keberapa pengguna merasa bosan.

Tabel 4.
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

Dengan hasil pengujian yang dapat dilihat pada Tabel 4.

Berdasarkan hasil pada Tabel 4 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.

Tabel 5.
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

Pada Tabel 5, 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.

C. Uji Pengguna 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

IV. DISKUSI

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.

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.

V. KESIMPULAN DAN SARAN

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

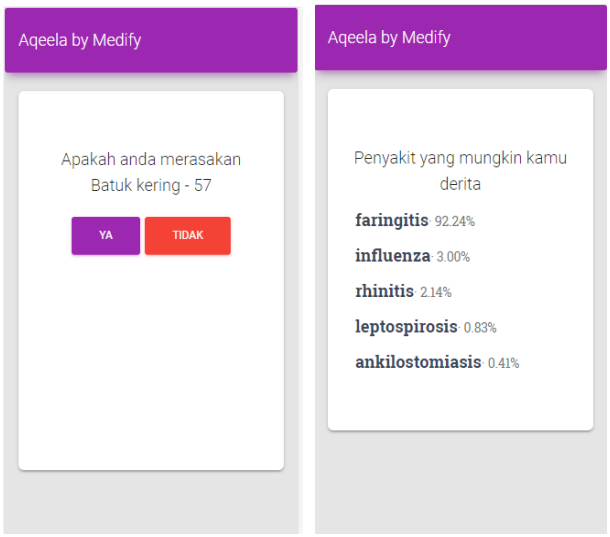
1. *FP Tree* memiliki hasil yang cukup baik dalam memetakan gejala gejala yang berkaitan sehingga dapat membuat tanya jawab yang saling terkait.
2. *Naïve Bayes* dan *SVM* dapat mendiagnosis penyakit berdasarkan input gejala pasien dengan baik.

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 yang diberikan untuk pengembangan aplikasi ini kedepannya adalah:

1. 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.
2. 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.

LAMPIRAN



Gambar 3. Contoh Tampilan Aplikasi

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] 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].

[3] N. Sadawi, "Generating Association Rules from Frequent Itemsets," 28 Agustus 2014. [Online]. Available: https://www.youtube.com/watch?v=-5Uia2a5jxc. [Accessed 30 Juni 2018].

[4] 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.

[5] B. Stecanella, "MonkeyLearn," MonkeyLearn Inc., 25 Mei 2017. [Online]. Available: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/. [Accessed 30 Mei 2018].

[6] R. Saxena, "Dataaspirant," Dataaspirant, 6 Februari 2017. [Online]. Available: http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/. [Accessed 2018 Juni 30].

[7] Geeks for Geeks, "Geeks for Geeks," Geeks for Geeks, [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-classifiers/. [Accessed 30 June 2018].

[8] 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].

[9] DF Team, "Data Flair," Data Flair, 12 Agustus 2017. [Online]. Available: https://data-flair.training/blogs/svm-kernel-functions/. [Accessed 2018 Juni 30].

[10] 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].

[11] S. Singh, "Sadanand Notes," Github, 8 Juli 2017. [Online]. Available: https://sadanand-singh.github.io/posts/svmpython/. [Accessed 30 Juni 2018].