BAB II
TEORI PENUNJANG

1.1 Penyakit dan Gejalanya

Keadaan sakit merupakan akibat dari kesalahan adaptasi terhadap lingkungan dan reaksi antara manusia dengan sumber-sumber penyakit. Ketika seseorang terserang bakteri atau virus penyakit, tubuh akan mengalami infeksi. Dari infeksi ini akan muncul gejala-gejala yang menimbulkan suatu penyakit. "Hampir semua penyakit akan mulai menunjukan eksistensinya di dalam tubuh setelah melalui beberapa tahapan atau gejala" [1]. Artinya sakit merupakan suatu keadaan yang memperlihatkan adanya keluhan dan gejala sakit secara subjektif dan objektif, sehingga pederita tersebut memerlukan pengobatan untuk mengembalikan keadaan sehat.

Gejala sakit ini merupakan indikasi adanya infeksi virus atau bakteri pada tubuh. Melalui gejala-gejala inilah kita bisa mengetahui jenis penyakit apa yang sedang diderita seseorang. Walaupun kenyataannya setiap keluhan bersifat relatif terhadap lingkungan dan usia. Namun mengenali penyakit melalui gejala-gejalanya merupakan langkah paling mudah dilakukan. Pada penelitian ini akan digunakan informasi berupa jenis-jenis penyakit yang biasa dialami orang dewasa beserta gejala-gejala yang timbul pada penyakit tersebut (daftar terlampir, lampiran A).

1.2 Klasifikasi Bayes

"Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Dalam mencapai tujuan tersebut, proses klasifikasi membentuk suatu model yang mampu membedakan data ke dalam kelas-kelas yang berbeda berdasarkan aturan atau fungsi tertentu" [14].

Atribut merupakan gambaran karakteristik dari suatu *variable class*. Dalam hal ini *variable class* yang dimaksud adalah jenis-jenis penyakit dan atributnya adalah gejala-gejala sakit. Pada beberapa penerapan, hubungan antara *attribute set* dengan *variable class* tidak pasti. Dengan kata lain, *class lable* dari sebuah pelatihan data tidak bisa ditentukan dengan pasti sebelum *attribute set* tersebut teridentifikasi melalui pelatihan. Keadaan ini mungkin timbul akibat adanya data nois atau adanya faktor luar yang mempengaruhi proses penklasifikasian.

Klasifikasi Bayes menjelaskan suatu model pendekatan probabilitas mengenai hubungan antara *attribute set* dengan *variable class*. Teorema Bayes merupakan prinsip statistik yang mengkombinasikan kelas-kelas yang sudah diketahui sebelumnya dengan data-data yang baru dikumpulkan [14].

1.2.1 Teorema Bayes

Misalkan X dan Y merupakan variabel acak. Hubungan probabilitas keduanya adalah P(X= x, Y= y), yaitu kemungkinan variabel X bernilai x dan variabel Y bernilai y. Probabilitas bersyaratnya adalah kemungkinan untuk variabel acak bernilai tertentu terhadap variabel acak yang lainnya yang sudah diketahui.

Misalkan probabilitas bersyarat P(Y=y|X=x) adalah probabilitas variabel Y akan bernilai y, untuk variabel X yang sudah diketahui bernilai x. Hubungan probabilitas bersyarat X dan Y ditulis dalam persamaan berikut:

$$P(X,Y) = P(Y|X)\times P(X) = P(X|Y)\times P(Y) \quad (2.1)$$

1.2.2 Klasifikasi Menggunakan Teorema Bayes

Dari persamaan (2.1), X menunjukan *attribute set* dan Y menunjukan *variable class*. Jika *variable class* memiliki hubungan yang tidak bisa dipastikan dengan atributnya, maka kita dapat memperlakukan X dan Y sebagai variabel acak dan menuliskan hubungan probabilitasnya dengan P(Y|X). Probabilitas

bersyarat ini diketahui sebagai probabilitas *posterior* untuk Y, dan sebagai lawannya probabilitas *prior* P(Y).

Selama tahap pelatihan, kita harus melatih probabilitas *posterior* P(Y|X) untuk semua kombinasi X dan Y berdasarkan informasi yang diperoleh dari pelatihan data. Dengan mengetahui probabilitas ini, *test record* X' dapat diklasifikasikan dengan menemukan kelas Y' adalah probabilitas *posterior* maksimum, P(Y'|X').

Menentukan probabilitas *posterior* secara akurat untuk setiap kombinasi yang mungkin dari *class lable* dan nilai atribut merupakan hal yang sulit karena membutuhkan set pelatihan yang sangat besar.

Teorema Bayes berguna karena memperbolehkan kita untuk menyatakan probabilitas *posterior* dengan menggunakan probabilitas *prior* P(Y), probabilitas kelas bersyarat P(X|Y), dan *evidence* P(X):

$$P(Y|X) = \frac{P(X|Y).P(Y)}{P(X)} \tag{2.2}$$

Ketika membandingkan probabilitas *posterior* untuk nilai Y yang berbeda, penyebut P(X) bernilai konstan sehingga bisa diabaikan. Probabilitas *prior* P(Y) bisa diperkirakan dengan mudah melalui set pelatihan dengan menghitung bagian *training record* untuk setiap kelas. Untuk memperkirakan probabilitas kelas bersyarat P(X|Y), kita gunakan klasifikasi Naïve Bayes.

1.2.3 Klasifikasi Naïve Bayes

Klasifikasi Naïve Bayes memperkirakan probabilitas kelas bersyarat dengan asumsi bahwa atribut bersyarat bebas, diberi *class lable* y. Asumsi bersyarat bebas bisa dinyatakan dalam persamaan berikut [14]:

$$P(X|Y = y) = \prod_{i=1}^{d} P(X_i|Y = y) \tag{2.3}$$

Di mana *attribute set* X={X_1, X_2, .. ,X_d} berjumlah d atribut.

Bersyarat Bebas

Sebelum memahami lebih dalam mengenai klasifikasi Naïve Bayes, kita pelajari dulu pengertian dari bersyarat bebas. Misalkan X, Y, dan Z merupakan variabel acak. Variabel X dinyatakan bebas dari Y untuk Z, atau dituliskan dalam:

$$P(X|Y,Z) = P(X|Z) \tag{2.4}$$

Bersyarat bebas antara X dan Y untuk persamaan (2.3) dapat dituliskan sebagai:

$$\begin{aligned} P(X,Y|Z) &= \frac{P(X,Y,Z)}{P(Z)} \\ &= \frac{P(X,Y,Z)}{P(Y,Z)} \times \frac{P(Y,Z)}{P(Z)} \\ &= P(X|Y,Z) \times (Y|Z) \\ &= P(X|Z) \times (Y|Z) \end{aligned} \tag{2.5}$$

Bagaimana klasifikasi Naïve Bayes bekerja

Dengan asumsi bersyarat bebas, untuk menghitung probabilitas kelas bersyarat untuk setiap kombinasi X, kita hanya perlu memperkirakan probabilitas dari setiap nilai x_i untuk Y. Pendekatan ini lebih praktis karena tidak memerlukan set pelatihan yang sangat besar untuk memperoleh perkiraan probabilitas yang baik.

Untuk mengklasifikasikan *test record*, klasifikasi Naïve Bayes menghitung probabilitas *posterior* untuk setiap kelas Y:

$$P(Y|X) = \frac{P(Y)\prod_{I=1}^{d} P(X_I|Y)}{P(X)} \tag{2.6}$$

Setelah P(X) diperoleh, untuk semua Y, kita bisa memilih kelas dengan nilai $P(Y)\prod_{I=1}^{d} P(X_I|Y)$ paling tinggi.

Memperkirakan Probabilitas untuk Atribut Mutlak

Untuk atribut mutlak X_i, probabilitas bersyarat $P(X_i=x_i|Y=y)$ adalah perkiraan pelatihan sampel pada kelas y untuk atribut bernilai x_i.

Memperkirakan Probabilitas untuk Atribut Kontinyu

Ada dua cara untuk memperkirakan probabilitas kelas bersyarat dari atribut yang kontinyu dengan klasifikasi Naïve Bayes [14]:

1. Dengan mendiskritkan setiap atribut kontinyu kemudian mengganti nilai atribut kontinyu tersebut sesuai dengan interval diskritnya. Dengan pendekatan ini atribut kontinu diubah menjadi atribut yang berurutan. Probabilitas bersyarat $P(X_i|Y=y)$ dihitung dengan menghitung catatan pelatihan milik kelas y yang termasuk dalam interval yang sesuai untuk X_i. Kesalahan estimasi tergantung pada jumlah interval diskrit. Jika intervalnya terlalu besar, perkiraan $P(X_i|Y)$ menjadi kurang akurat. Tapi, jika jumlah interval terlalu kecil, bisa jadi data pelatihannya termasuk kedalam kelas yang berbeda dan kita bisa kehilangan batas interval yang benar.
2. Kita dapat mengasumsikan bentuk tertentu dari distribusi probabilitas untuk variabel kontinyu dan mengestimasi parameter distribusi menggunakan *training records*. Sebuah distribusi Gauss biasanya dipilih untuk mewakili probabilitas kelas bersyarat untuk atribut kontinyu. Distribusi ini ditandai oleh dua parameter, rata-rata μ dan variannya σ. Untuk setiap y_j kelas, probabilitas bersyarat kelas untuk atribut X_i adalah

$$\boldsymbol{P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} exp^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}} \quad (2.7)$$

Parameter μ_{ij} bisa diperoleh dari rata-rata sampel $X_i(\bar{x})$ untuk setiap *training records* untuk kelas y_j. Begitu pun dengan $\boldsymbol{\sigma_{ij}^2}$ bisa diperoleh melalui varian sampel (s^2) dari *training records*.

Perhatikan bahwa sebelumnya interpretasi probabilitas kelas bersyarat tidak akurat. Sisi kanan dari persamaan (2.7) sesuai dengan fungsi kepadatan probabilitas, f (X_i: μ_{ij}, σ_{ij}). Karena fungsi kontinu, probabilitas dari variabel acak X_i untuk nilai tertentu adalah nol. Sebaliknya, kita dapat menghitung probabilitas bersyarat X_i ada pada interval, x_i dan $x_i+\epsilon$, di mana ϵ adalah konstanta kecil

$$P(x_i \leq X_i \leq x_i + \epsilon | Y = y_j) = \int_{x_i}^{x_i+\epsilon} f(X_i; \mu_{ij}, \sigma_{ij}) dX_i$$
$$\approx f(X_i; \mu_{ij}, \sigma_{ij}) \times \epsilon \quad (2.8)$$

Faktor perkalian konstan $\in$ untuk masing-masing kelas, bisa digunakan untuk memperoleh nilai probabilitas posterior untuk P (Y | X). Oleh karena itu, kita masih dapat menerapkan persamaan (2.7) untuk mendekati probabilitas kelas bersyarat P (X_i|Y).

M-estimate untuk Probabilitas Bersyarat

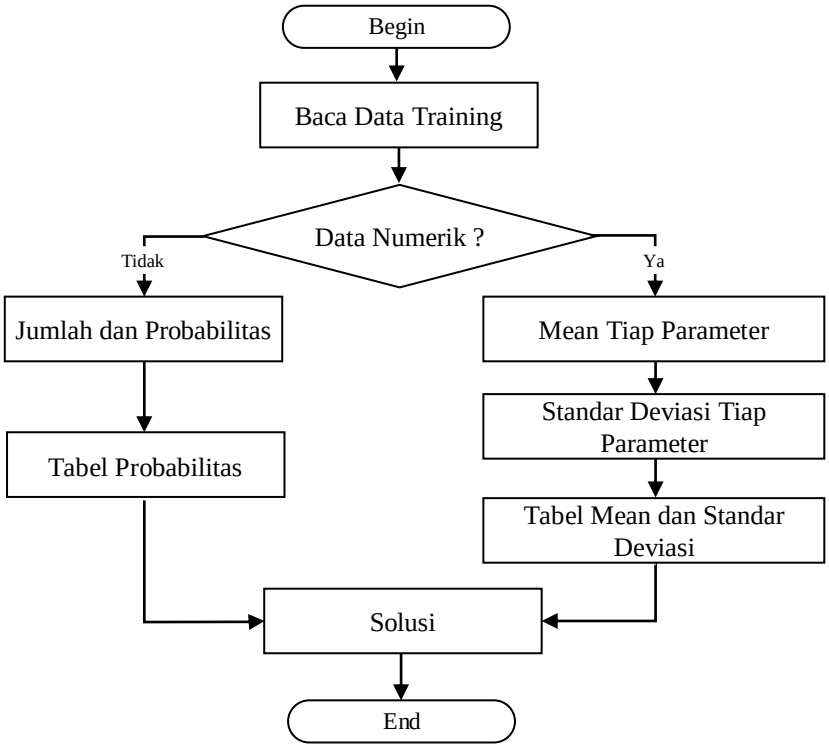
Jika probabilitas kelas bersyarat untuk salah satu atribut adalah nol, maka probabilitas *posterior* keseluruhan untuk kelas hilang. Pendekatan ini memperkirakan probabilitas kelas bersyarat menggunakan pecahan sederhana mungkin tampak terlalu rapuh, terutama ketika ada beberapa contoh pelatihan yang tersedia dan jumlah atribut besar [14].

Dalam kasus yang lebih sulit, jika contoh pelatihan tidak mencakup banyak nilai atribut, kita tidak akan bisa mengklasifikasikan *test record*. Masalah ini bisa diatasi dengan menggunakan pendekatan m-estimate untuk memperkirakan probabilitas bersyarat

$$P(x_i | y_j) = \frac{n_c + mp}{n+m} \quad (2.9)$$

Di mana n adalah jumlah *training set*, n_c adalah jumlah contoh pelatihan dari kelas y_j untuk nilai x_i, m adalah jumlah parameter, dan p adalah kemungkinan variabel dipilih. Jika tidak ada *training set* (n=0), maka $P(x_i | y_j) = p$. Oleh karena itu, p

dapat dianggap sebagai probabilitas *prior* nilai atribut x_i dengan kelas y_j. Ukuran sampel menentukan kesetaraan antara probabilitas *prior* p dan probabilitas pengamatan n_c/n



Gambar 2.1 Diagram Metode Naïve Bayes

1.3 UML

Pemodelan digunakan untuk penyederhanaan permasalahan. Sasaran model adalah abstraksi dari segala sesuatu yang ada menjadi gambaran-gambaran yang mudah dipahami [8]. UML (*Unified Modeling Language*) adalah salah satu model untuk merancang *software* yang berorientasikan pada objek (*object oriented*), di mana segala sesuatu dipandang sebagai objek. Pada pemodelan ini dilakukan pendekatan pada objek yang mengkombinasikan data dan fungsionalitasnya.

Perangkat lunak dikembangkan berdasarkan analisis kebutuhan. Analisis kebutuhan ini merupakan tahap konseptualisasi, yaitu tahap yang mengharuskan analis dan perancang sistem berusaha untuk tahu secara pasti mengenai hal yang

menjadi kebutuhan dan harapan pengguna sehingga aplikasi dapat digunakan oleh pengguna serta memuaskan kebutuhan dan harapannya [8].

Dalam pemodelan UML ada sembilan macam diagram sistem yang digunakan untuk menggambarkan permasalahan maupun solusinya. Setiap diagram digambarkan menurut sudut pandang tertentu. Pada penelitian ini hanya akan digunakan 4 diagram sebagai berikut:

1. *Use Case Diagram*

Use case diagram mendeskripsikan bagaimana perangkat lunak akan digunakan oleh penggunanya. *Use case diagram* berperan penting dalam perancangan, mencari kelas-kelas yang terlibat dalam aplikasi dan untuk melakukan pengujian. Dengan *use case diagram*, setiap objek yang berinteraksi dengan sistem disebut aktor sementara *use case* merupakan deskripsi lengkap tentang sistem berperilaku pada aktornya [8].

2. *Class Diagram*

Kelas (*class*) didefinisikan sebagai kumpulan objek yang memiliki kesamaan atribut, perilaku, serta cara berhubungan dengan objek lain [8]. Pada *class diagram* digambarkan struktur dan deskripsi *class, package,* dan objek serta hubungan satu sama lain. Ada tiga macam hubungan dalam *class diagram*, yaitu: *Association* sebagai hubungan antara dua kelas, *Aggregation* merupakan hubungan satu kelas sebagai bagian dari kumpulan kelas, dan G*eneralization* adalah hubungan suatu kelas sebagai turunan dari kelas lain [8].

3. *Sequence Diagram*

Sequence diagram menggambarkan skenario interaksi antar objek suatu sistem. Pada diagram ini dijelaskan bagaimana operasi dilakukan, pesan yang akan dikirim dan waktu pelaksanaanya. Digambarkan dalam dimensi waktu (vertikal) dan objek yang terkait (horizontal) [8].

4. *Collaboration Diagram*

Collaboration Diagram merupakan kelas diagram yang menjelaskan peran suatu klasifikasi serta peran asosiasi. Dalam diagram ini digambarkan konfigurasi objek dan link yang mungkin terjadi ketika kolaborasi dijalankan. Di mana objek merupakan peran klasifikasinya dan *link* merupakan peran asosiasinya. Peran asosiasi dapat digunakan pada berbagai jenis *link*, seperti argumen prosedur atau variabel prosedur lokal. Simbol link dapat digunakan untuk menunjukan *link* semetara (<<parameters>> atau <<local>>) atau dipanggil ke objek yang sama (<<self>>). Diagram ini hanya menjelaskan satu objek yang terlibat dalam kolaborasi, walaupun mungkin ada objek lain dalam sistem [11].

1.4 Java

"Bahasa pemrograman java digunakan untuk aplikasi-aplikasi desktop, aplikasi yang berjalan di lingkup enterprise dan internet, serta aplikasi yang berjalan di sarana-sarana komputasi yang terbatas sumber daya pemrosesan dan terbatas sumber daya penggunaan memorinya. Java merupakan bahasa pemrograman berorientasi objek yang dapat digunakan untuk aplikasi lintas *platform*" [8].

Bahasa pemrograman Java merupakan bahasa pemrograman berorientasi objek sehingga dapat digunakan untuk membangun aplikasi berbasis android. Bahasa pemrograman java dapat digunakan pada perangkat mobile.

1.5 Android

Android adalah sistem operasi *mobile* yang kuat, dibangun menggunakan kombinasi bahasa pemrograman Java dan berbasis *layout* XML dan konfigurasi file. Lingkungan pengembangan android berjalan melalui aplikasi dasar Hello World, dan mencakup alat Android dengan penekanan pada *User Interface* (UI) yang tersedia di *Developer Tools Android* (ADT) *Plugin* [9].

1.6 Pengujian Aplikasi

Alpha/ Beta testing biasa dilakukan untuk menentukan kualitas suatu aplikasi sebelum aplikasi tersebut digunakan secara luas. Pengujian *alpha* dilakukan oleh pengembang. Tujuannya untuk mengetahui apakah aplikasi dapat berjalan dengan baik sesuai dengan perancangan. Ada dua metode dalam pengujian *alpha* yaitu *white box* dan *black box*.

Metode *white box* merupakan pengujian dengan melihat kinerja sistem dari dalam. Pengujian dilakukan oleh pengembang dengan menguji setiap prosedur yang ada pada aplikasi satu persatu. Pengujian dikhususkan pada kode-kode program dalam aplikasi [10].

Sedangkan metode *black box* merupakan pengujian perangkat lunak, di mana fungi-fungsi perangkat lunak diuji tanpa perlu melihat bagian dalam dari aplikasi tersebut. Metode ini menganggap aplikasi sebagai kotak hitam yang tidak dapat dilihat bagian dalamnya [10].

Pengujian *beta*, merupakan tahapan pengujian aplikasi yang melibatkan pengguna. Pada pengujian *beta*, aplikasi diuji dan dinilai kualitasnya langsung oleh pengguna.

1.6.1 Metode Black Box Testing

Untuk menguji aplikasi yang telah dibuat, penulis akan menggunakan metode *black box testing*. *Black box testing* dilakukan tanpa mengetahui detil struktur dari sistem atau komponen yang dites dan hanya berfokus pada spesifikasi kebutuhan dari software. Dengan adanya *black box testing* perekayasa *software* dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan fugsioal pada suatu program [5].

Kategori *error* yang akan diketahui melalui *black box testing* [5]:

1. Fungsi yang hilang atau tidak benar
2. *Error* dari antarmuka
3. *Error* dari struktur data atau akses eksternal *database*
4. *Error* dari kinerja atau tingkah laku

5. *Error* dari inisialisasi dan terminasi

Black box testing digunakan pada tahap akhir dan berfokus pada domain informasi. Tes didisain untuk menjawab pertanyaan sebagai berikut [5]:

1. Bagaimana tepatasi fungsi yang akan dites
2. Bagaimana tingkah laku dan kinerja sistem dites
3. Kategori masukan apa yang bagus digunakan untuk *test cases*
4. Apakah sebagian sistem sensitif terhadap satu nilai masukan tertentu
5. Bagaimana batasan suatu kategori masukan ditetapkan
6. Sistem mempunyai toleransi jenjang dan volume data apa saja
7. Apa saja akibat dari kombinasi data tertentu yang terjadi pada operasi sistem

Dekomposisi Kebutuhan untuk Dites Secara Sistematis

Untuk dapat membuat *test cases* yang efektif, harus dilakukan dekomposisi dari tugas-tugas testing suatu sistem ke aktivitas-aktivitas yang lebih kecil dan dapat dikelola, hingga tercapai *test case* individual [5].

Spesifikasi atau model sistem adalah titik awal memulai desain tes. Model sistem dapat berupa spesifikasi fungsional, kinerja, skenario pengguna, atau spesifikasi berdasarkan resiko sistem. Spesifikasi menggambarkan kriteria yang digunakan untuk menentukan operasi yang benar atau dapat diterima, sebagai acuan pelaksanaan tes [5].

Selanjutnya dibuat desain tes pada spesifikasi komponen yang dites. Objektifitas tes tingkat atas disusun berdasarkan pada spesifikasi komponen. Tiap objektifitas tes ini kemudian didekomposisikan ke dalam objektifitas tes lainnya.

Equivalence Partitioning

Metode *black box testing* membagi domain masukan dari suatu program ke dalam kelas-kelas data, di mana *test cases* dapat diturunkan. Petunjuk pelaksanaan dalam melakukan *equivalence partitioning* adalah sebagai berikut [5]:

- Jika masukan mempunyai jenjang tertentu, maka definisikan kategori tepat dan tak tepat terhadap jenjang masukan tersebut.
- Jika masukan membutuhkan nilai tertentu, definisikan kategori tepat dan tak tepat.

- Jika masukan membutuhkan himpunan masukan tertentu, definisikan kategori tepat dan tak tepat.
- Jika masukan adalah boolean, definisikan kategori tepat dan tak tepat.

Sedangkan beberapa kombinasi yang mungkin dalam partisi ekuivalensi, adalah [5]:

- Nilai masukan yang tepat atau tak tepat.
- Nilai numerik yang negatif, positif, atau nol.
- String yang kosong atau tidak kosong.
- Daftar (list) yang kosong atau tidak kosong.
- File data yang ada dan tidak, yang dapat dibaca/ ditulis atau tidak.
- Tipe *file* data.
- Sumber atau tujuan *file.*

1.6.2 Skala Likert

Skala likert digunakan untuk memberi peluang kepada responden untuk mengekspresikan perasaan mereka dalam bentuk *persetujuan* terhadap suatu pernyataan. Pernyataan tersebut berjenjang mulai dari yang rendah hingga tertinggi. Jumlah pilihan jawabannya harus ganjil. Semakin banyak pilihannya, semakin mewakili jawaban responden. Namun, akan semakin sulit mencari kata-kata yang dapat dipahami secara umum. Sepanjang dapat disepakati dalam tata bahasa, pilihan jawaban tersebut dapat digunakan [13].

Pada skala likert, pilihan jawaban dibuat berjenjang seperti: *sangat tidak setuju, tidak setuju, ragu, setuju, dan sangat setuju*. Dengan begitu, setiap jawaban dapat diberi skor. Pemberian skor untuk pilihan jawaban tersebut adalah 1-5 atau bisa juga sebaliknya, asalakan konsisten [13].

Berikut adalah langkah-langkah yang dapat ditempuh untuk mengetahui bagaimana sikap tiap responden terhadap kualitas produk [4]:

1. Menentukan skor maksimal, yaitu skor jawaban terbesar dikali banyak item.
2. Menentukan skor minimal, yaitu skor jawaban terkecil dikali banyak item.

3. Menentukan nilai median, yaitu hasil penjumlahan skor maksimal dengan skor minimal dibagi dua.
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan skor minimal dengan median dibagi dua.
5. Menentukan nilai kuartil 3, yaitu hasil penjumlahan skor maksimal dengan median dibagi dua.
6. Buatlah skala yang menggambarkan skor minimal, nilai kuartil 1, median, kuartil 3, dan skor maksimal.
7. Carilah batas-batas skor untuk masing-masing kategori sikap.
8. Buatlah tabel distribusi frekuensi sikap tiap responden terhadap kualitas produk.

Keterangan pencarian rentang nilai pada frekuensi :

- Kategori sikap sangat positif, yaitu daerah yang dibatasi oleh kuartil 3 dan skor maksimal (kuartil 3 $\leq$ x $\leq$ skor maksimal).
- Kategori sikap positif, yaitu daerah yang dibatasi oleh median dan kuartil 3 (median $\leq$ x< skor maksimal).
- Kategori sikap negative, yaitu daerah yag dibatasi oleh kuartil 1 dan median (kuartil 1 $\leq$ x < median).
- Kategori sikap sangat negative, yaitu daerah yang dibatasi oleh skor minimal dan kuartil 1 median (skor minimal $\leq$ x < kuartil 1).

Skala Likert tidak mengijinkan adanya pernyataan *item* netral. Jadi pernyataan yang ada dalam skala Likert hanya dua, pernyataan item posistif dan *item* negatif.