Data Mining

Algoritma dan penerapan sederhana

# Data Mining

Data Mining adalah melakukan ekstraksi untuk mendapatkan informasi penting yang sifatnya implisit dan sebelumnya tidak diketahui, dari suatu data. Data mining adalah proses menemukan korelasi baru yang bermakna, pola dan tren dengan memilah-milah sejumlah besar data yang tersimpan dalam repositori, menggunakan teknologi penalaran pola serta teknik-teknik statistik dan matematika.

Data Mining melakukan analisa pada dataset (yang besar) untuk menemukan hubungan dan menyimpulkan data tersebut berguna dan bermanfaat bagi pemilik data. Data mining dapat dikatakan sebagai proses dalam memperoleh informasi baru dari data-data lama yang belum diolah dan diproses.

Data Mining dapat berperan dalam:

1. **Estimasi**

Model dibangun dari data dengan record yang lengkap, yang menyediakan nilai dari variabel sebagai prediktor, kemudian estimasi nilai dari variable target ditentukan berdasarkan nilai dari variabel prediktor. Penentuan kebijakan atau suatu nilai pada proses yang akan dilakukan. Estimasi dapat dilakukan dari data-data lama yang akan diolah.

Metode yang cocok untuk estimasi, yakni: Linear Regression, Neural Network, Support Vector Machine, dan yang lainnya.

1. **Prediksi**

Algoritma prediksi sama dengan algoritma estimasi di mana label/target/class bertipe numerik, bedanya adalah data yang digunakan merupakan data rentetan waktu (data time series). Sifat prediksi bisa menghasilkan class berdasarkan berbagai atribut yang kita sediakan. Penentuan hasil dari proses yang sedang berlangsung. Data-data yang digunakan untuk prediksi berasal dari data yang ada saat proses sedang berlangsung.

Metode yang cocok untuk estimasi, yakni: Linear Regression, Neural Network, Support Vector Machine, dan yang lainnya.

1. **Klasifikasi**

algoritma yang menggunakan data dengan target/class/label berupa nilai kategorikal (nominal). Pengelompokan data-data yang ada menjadi dalam kelompok yang sudah ditentukan nama kelompoknya.

Metode yang cocok untuk estimasi, yakni: Naive Bayes, K-Nearest Neighbor, C4.5, ID3, CART, Linear Discriminant Analysis, dan yang lainnya.

1. **Klastering**

Klastering adalah pengelompokkan data, hasil observasi dan kasus ke dalam class yang mirip, koleksi data yang mirip antara satu dengan yang lain, dan memiliki perbedaan bila dibandingkan dengan data dari klaster lain. Pengelompokan data-data yang ada dan memiliki beberapa kesamaan menjadi dalam kelompok yang tidak perlu ditentukan nama kelompoknya.

Metode yang cocok untuk estimasi, yakni: K-Means, K-Medoids, Self-Organizing Map (SOM), Fuzzy C-Means, dan yang lainnya.

1. **Asosiasi**

Algoritma *association rule* (aturan asosiasi) adalah algoritma yang menemukan atribut yang “muncul bersamaan”. Pengelompokan data-data yang saling terkait dan berhubungan. Algoritma asosiasi akan mencari aturan yang menghitung hubungan diantara dua atau lebih atribut.

Metode yang cocok untuk estimasi, yakni: FP-Growth, A Priori, dan yang lainnya.

# Algoritma Klasifikasi

Klasifikasi dalam *data mining*  merupakan metode pembelajaran data untuk memprediksi nilai dari sekelompok attribut. Algoritma klasifikasi akan menghasilkan sekumpulan aturan yang disebut *rule* yang akan digunakan sebagai indicator untuk dapat memprediksi kelas dari data yang ingin diprediksi.Klasifikasi digunakan dalam banyak sekali bidang, dan secara teori algoritma klasifikasi sama seperti otak manusia. Otak manusia mampu mengolah data yang sudah ada sebagai pengalaman dalam bertindak.

Dalam *data mining* ada beberapa algoritma klasifikasi seperti *naïve bayes*, *neural network, logistic regression, support vector machine,*dan *decision tree*. Tujuan dari algoritma klasifikasi adalah untuk menemukan relasi antara beberapa variable yang tergolong dalam kelas yang sama. Relasi tersebut akan digambarkan dengan aturan aturan agar dapat memprediksi kelas dari data yang attributnya sudah diketahui.

Proses pembuatan model klasifikasi dibedakan menjadi tiga tahap besar :

1. **Tahap Pembelajaran**

Tahap dimana algoritma klasifikasi diterapkan kedalam data contoh untuk mendapatkan relasi data dalam setiap kelasnya. Tahap ini akan membentuk model yang berisikan aturan aturan attribut dalam menentukan kelas data.

1. **Tahap Pengujian**

Tahap pengujian merupakan tahap penerapan aturan aturan yang sudah terbentuk pada tahap pembelajaran kedalam contoh data yang tidak termasuk dalam data pembelajaran. Dalam Tahap ini, aturan model yang dimiliki akan diterapkan pada setiap attribut dalam data pengujian dan dilihat kecocokan antara kelas yang diprediksi dengan kelas data sesungguhnya.

1. **Tahap Prediksi**

Tahap ini merupakan tahap dimana model yang dihasilkan benar benar diterapkan pada data yang belum diketahui kelasnya.

Penilaian algoritma klasifikasi biasanya dilihat dari akurasi model. Akurasi model merupakan ketepatan model dalam memprediksi kelas data. Selain akurasi kecepatan pembentukan model, kemampuan algoritma dalam mengatasi data yang tidak relevan atau bahkan data yang tidak lengkap, serta kemampuan algoritma ketika diterapkan pada data jumlah besar maupun kecil.

## Decision Tree C4.5 dengan Information Gain

C4.5 didesain oleh J. Ross Quinlan, dinamakan C4.5 karena merupakan keturunan dari pendekatan ID3 untuk membangun pohon keputusan. C4.5 merupakan algoritma yang cocok digunakan untuk masalah klasifikasi pada machine learning dan data mining. C4.5 memetakan atribut dari kelas sehingga dapat digunakan untuk menemukan prediksi terhadap data yang belum muncul.

Pohon keputusan sendiri merupakan pendekatan “*divide and conquer*” dalam mempelajari masalah dari sekumpulan data independen yang digambarkan dalam bagan pohon. Pohon keputusan juga merupakan sekumpulan pertanyaan yang tersusun secara sistematis, dimana setiap pertanyaan yang ada menentukan percabangan berdasarkan nilai atribut dan berhenti pada daun dari pohon yang merupakan prediksi dari kelas variable.

Semua metode yang menghasilkan pohon keputusan bekerja mulai dari akar yang paling atas, dengan menguji data yang ada dan dipecah kedalam setiap cabang dari pohon yang lebih kecil. Keputusan dipilih berdasarkan kesesuaian antara data yang diuji dengan aturan dari cabang pohon keputusan yang ada hingga mencapai daun sehingga nilai kelas yang terdapat pada daun diberikan pada tupel data yang diperiksa tersebut.



Gambar 1  
Contoh pohon keputusan

Pada Gambar diatas, digambarkan konsep dari *buys\_computer*, yang digunakan untuk prediksi pelanggan yang ingin membeli komputer. Data terdiri dari umur, pekerjaan dan *creditrating*. Node internal digambarkan dalam bentuk persegi dan node daun digambarkan dalam bentuk oval.

Didalam pohon keputusan node pusat merupakan attribute dari data yang diuji (*tuple*), cabang merupakan hasil dari pengujian atribut, dan daun merupakan kelas yang terbentuk. Jika atribut yang diuji merupakan nilai nominal, biasanya jumlah cabang yang terbentuk akan sejumlah dengan jumlah kemungkinan dari nilai attribute yang dipilih sesuai dengan metode pemilihan atribut. Metode pemilihan attribute akan membuat daftar attribut yang paling berpengaruh dalam penyusunan cabang pohon keputusan agar cabang yang terbentuk terdiri dari data yang sejenis. Oleh karena itu, semakin mirip data dalam cabang yang terbentuk akan membuat pohon keputusan yang semakin akurat.

Dalam pembuatan cabang, ada tiga kemungkinan yang akan terjadi sesuai dengan jenis data. Jika A merupakan salah satu atribut dari data yang diuji, maka kemungkinan yang akan terjadi akan bergantung pada nilai A:

1. Jika A bernilai diskrit, maka cabang akan terbentuk untuk setiap nilai A. Atribut A akan dikeluarkan dari daftar atribut yang perlu diperiksa karena setelah cabang A terbentuk nilai atribut A pada setiap cabang akan selalu sama (Gambar 2.3 a).
2. Jika A bernilai kontinyu, maka akan terbentuk 2 buah cabang, dimana A ≤ dari nilai perpecahan, dan A > nilai perpecahan. Nilai perpecahan ditentukan oleh metode pemilihan atribut saat membuat daftar attribute (Gambar 2.3 b).
3. Jika A bernilai Diskrit dan biner, maka akan terbentuk 2 cabang yaitu cabang untuk nilai benar, dan cabang untuk nilai salah(Gambar 2.3 c).



Gambar 2   
Pembuatan cabang pohon keputusan

Dalam pembuatan pohon keputusan, setiap algoritma menerapkan ukuran pemilihan atribut yang berbeda-beda. Ukuran pemilihan atribut merupakan ukuran yang digunakan dalam menentukan kriteria yang terbaik untuk mengelompokkan *tuple*. Ukuran pemilihan atribut ini juga disebut sebagai *splitting rules* karena menentukan bagaiman data akan dipisahkan kesetiap cabang. C4.5 yang merupakan pengembangan dari ID3 menggunakan *Information gain* untuk ukuran pemilihan atribut.

*Information gain* diciptakan oleh Claude Shannon dengan mempelajari nilai informasi dari data, dan menggunakan nilai tersebut sebagai acuan dalam menentukan atribut yang akan digunakan dalam menyusun pohon keputusan. Atribut yang dipilih akan menghasilkan partisi dengan data yang lebih seragam, dan dapat menghasilkan pohon keputusan yang sesederhana mungkin dengan perulangan yang sedikit.

Pohon keputusan menjadi sangat popular karena dalam pembuatan klasifikasi pohon keputusan tidak memerlukan pengetahuan yang banyak ataupun pengaturan yang rumit. Pohon keputusan mampu mengolah data dengan dimensi yang besar, serta menghasilkan pengetahuan dalam diagram pohon yang mudah dimengerti oleh manusia. Metode pembelajaran dalam membuat pohon keputusan tergolong mudah dan cepat, serta memiliki akurasi tinggi. Pohon keputusan banyak diaplikasikan dalam berbagai bidang seperti obat-obatan, produksi, analisa keuangan, astronomi, dan biologi.

Beberapa kelebihan dari pohon keputusan :

1. Hasil analisa berupa diagram pohon yang sangat mudah dimengerti.
2. Mudah untuk dibangun, serta membutuhkan data percobaan yang lebih sedikit dibandingkan algoritma klasifikasi lainnya.
3. Mampu mengolah data nominal dan kontinyu.
4. Model yang dihasilkan dapat dengan mudah dimengeri, beberapa teknik klasifikasi yang berbeda seperti neural network menyajikan model dengan informasi logis yang tersirat, sehingga perlu dipelajari.
5. Menggunakan teknik statistic sehingga dapat divalidasikan.
6. Cepat dan handal dalam mengolah dataset besar.
7. Akurasi yang dihasilkan mampu menandingi teknik klasifikasi yang lainnya.

#### Tahapan algoritma C4.5

Berikut tahapan algoritma dalam pembuatan pohon keputusan dengan algoritma C4.5:

**Tahap 1**

Perhatikan label pada data, jika sudah sama semua, maka akan dibentuk daun dengan nilai label data keseluruhan.

**Tahap 2**

Menghitung nilai informasi dengan menggunakan semua data yang ada dengan formula (2.1)

(2.1)

Dimana merupakan probabilitas tuple dalam *D* yang menjadi kelas dengan asumsi .*Info(D)* atau disebut juga *entropy* dari D merupakan rata rata informasi yang diperlukan untuk identifikasi tuple dalam *D*(Alpaydın, 2010: p. 189). Jika Nilai A adalah nilai diskrit maka data D akan dipisahkan sejumlah nilai data A sehingga nilai setiap cabang akan murni dan sejenis. Setelah percabangan pertama, jumlah percabangan yang mungkin terjadi diukur dengan persamaan.

**Tahap 3**

Menghitung nilai informasi dengan formula (2.2) untuk setiap attribut dengan memperhatikan isi data dari attribut.

(2.2)

Dimana merupakan bobot dari partisi j. merupakan informasi yang diperlukan untuk mengklasifikasikan tuple dari *D* pada partisi *A*. Semakin kecil hasil persamaan ini, semakin baik pula partisi yang dihasilkan. Nilai dari sebuah atribut menentukan penting tidaknya atribut tersebut dalam penyusunan pohon keputusan.

Jika atribut bernilai kontinyu, maka akan dicari *split\_point* dengan cara mengurutkan seluruh data menurut attribut tersebut dari kecil ke besar, lalu dirata-rata antar satu data dengan data setelahnya. Nilai informasi akan dihitung menurut satu persatu calon *split\_point*. Dan nilai *split\_point* yang akan dipilih yang terkecil.

**Tahap 4**

Nilai gain untuk setiap attribut akan diperhitungkan dengan formula (2.3), nilia dengan gain tertinggi akan dijadikan cabang dalam pohon keputusan.

(2.3)

**Tahap 5**

Setelah cabang pohon keputusan terbentuk, perhitungan dilakukan kembali seperti pada tahap 1 sampai 4. Namun jika cabang telah mencapai maksimal cabang yang diperbolehkan, daun akan terbentuk dengan nilai mayoritas dari nilai data.

#### Pruning dalam pohon keputusan

Untuk mendapatkan data yang benar benar sejenis, maka akan terbentuk banyak sekali cabang dalam pohon keputusan. Data yang terlalu beragam ataupun data acak akan membuat struktur pohon keputusan menjadi terlalu rumit. Di dalam pohon keputusan dikenal istilah *pruning*yaitu memangkas cabang yang tidak terlalu besar pengaruhnya agar diagram ang dihasilkan lebih akurat dan simple.

Ada dua pendekatan *pruning* yang digunakan :

1. *Prepruning*menghentikan proses pembuatan cabang pada titik tertentu. Semakin besar perulangan pembuatan cabang yang diperbolehkan, semakin besar pula kompleksitas dari pohon keputusan yang didapat jika data beragam, namun jika jumlah perulangan terlalu kecil, diagram pohon yang dihasilkan menjadi kurang akurat.
2. *Postpruning*memotong cabang pohon yang kurang mereprensentasikan data setelah sebuah pohon keputusan terbentuk. Kelas yang diberikan akan diukur dari jumlah persebaran label yang ada pada cabang tersebut.

Proses *pruning* dalam pohon keputusan dapat dilihat pada Gambar dibawah. Gambar tersebut menjukkan model pohon keputusan sebelum dan sesudah pruning. Pemotongan cabang ini selain mempermudab model sehingga terlihat lebih sederhana, namun juga dapat meningkatkan akurasi. Selain diterapkan pada cabang yang kurang akurat, *pruning* juga dapat diterapkan pada pohon yang cabangnya terulang beberapa kali. Contoh pohon dengan attribute A1 yang berulang dapat dilihat pada gambar dibawah menunjukkan perulangan pada cabang yang lebih besar dimana semua cabang dari attribute *credit­\_rating* berulang dengan aturan yang sama.

Algoritma C4.5 menggunakan *pessimisticpruning*yang mampu mengkalkulasi tingkat error yang digunakan sebagai acuan dalam pemangkasan cabang pohon keputusan. Baik *postpruning* dan *prepruning* dapat dikombinasikan karena tidakada teknik yang lebih baik antara keduanya. Walaupun pohon keputusan yang muncul setelah *pruning* akan lebih singkat, namun terkadang masih muncul repetisi dan replikasi cabang. Pengunaan kombinasi atribut dalam pemecahan cabang dapat mengatasi masalah perulangan ini.



Gambar 3   
Pohon keputusan yang dipangkas dengan *postprunnin*g



Gambar 4   
Contoh repetisi pohon keputusan pada atribut A1



Gambar 5   
Perulangan dalam pohon keputusan

#### Contoh penggunaan algoritma C4.5

Dengan menggunakan *information gain*, data dari Tabel berikut akan digunakan sebagai *data training*. Untuk memprediksi pelanggan yang membeli komputer. Untuk dapat mengukur nilai informasi setiap atribut, maka perlu diukur nilai informasi yang diperlukan untuk klasifikasi tuple. Dari data tersebut 9 data yadan 5 data tidak*.*

Tabel 1   
Data pembelian komputer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Umur | pendapatan | pelajar | *credit\_rating* | Pembelian |
| 1 | Muda | tinggi | bukan | cukup | Tidak |
| 2 | Muda | tinggi | bukan | baik | Tidak |
| 3 | Menengah | tinggi | bukan | cukup | Ya |
| 4 | Senior | menengah | bukan | cukup | Ya |
| 5 | Senior | rendah | ya | cukup | Ya |
| 6 | Senior | rendah | ya | baik | Tidak |
| 7 | Menengah | rendah | ya | baik | Ya |
| 8 | Muda | menengah | bukan | cukup | Tidak |
| 9 | Muda | rendah | ya | cukup | Ya |
| 10 | Senior | menengah | ya | cukup | Ya |
| 11 | Muda | menengah | ya | baik | Ya |
| 12 | Menengah | menengah | bukan | baik | Ya |
| 13 | Menengah | tinggi | ya | cukup | Ya |
| 14 | Senior | menengah | bukan | baik | Tidak |

Kemudian satu persatu atribut akan diuji dan untuk dilihat nilai gain terbesar, berikut cara penghitungan atribut umur. Didalam atribut tersebut untuk nilai muda2 membeli komputer dan 3 tidak,semua umurmenengah membeli komputer ( 4 ya), senior 3 membeli komputer dan 2 tidak.

Maka nilai gain akan menjadi

Dengan cara yang sama nilai gain untuk masing masing atribut dihitung, hasil perhitungan nilai gain untuk pendapatan = 0.029 bits, pelajar = 0.151 bits, dan *credit*\_*rating* = 0.048 bits. Karena nilai*gain*dari atribut umur yang paling tinggi, maka percabangan pertama menggunakan nilai dari atribut umur. Atribut umur sendiri terdiri dari 3 jenis data yaitu muda, menengah dan senior, oleh karena itu diagram pohon yang terbentuk memiliki tiga cabang. Karena semua data dengan umur menengah membeli komputer, maka cabang ini dapat dijadikan node daun dengan nilai “ya”. Diagram pohon fase pertama digambarkan pada Gambar 2.9.



Gambar 6   
keputusan setelah dicabangkan dengan *age*

Contoh diatas menggambarkan data yang dengan nilai diskrit, untuk percabangan data nominal perlu dihitung nilai perpecahan (*split\_point*) untuk menentukan nilai terbaik dalam pemecahan cabang pohon keputusan. Nilai perpecahan didapat dari pengurutan semua data dalam satu attribut dari data terkecil sampai terbesar, kemudian dirata-ratakan setiap tahap, setiap angka muncul akan dihitung nilai informasinya (2), nilai informasi yang paling kecil akan menjadi *split*\_*point*.



Gambar 7   
Pemilihan titik perpecahan dalam atribut

Perhitungan pada tahap selanjutnya mengikuti langkah sebelumnya, pohon keputusan akan berhenti ketika semua data sudah teridentifikasi menjadi kelas, atau sampai pada batasan cabang yang diperbolehkan. Jika cabang yang terbentuk terlalu besar, ataupun berulang, dapat dilakukan *pruning* agar pohon menjadi lebih mudah dimengerti.

## 

## Naives Bayes

Pendekatan Bayesian digunakan untuk menentukan kemungkinan terhadap asumsi disekitarnya. Dalam statistik Bayesian, parameter dipertimbangkan terhadap variabel yang acak dan data dipertimbangkan terhadap hasil kemungkinan. Pendekatan Bayesian pertama kali dilakukan oleh Reverend Thomas Bayes (1702-1761) pada *“Essay Towards Solving a Problem in the Doctrine of Chances”* yang dipublikasikan tahun 1763.

Naive Bayes merupakan *machine learning* yang menggunakan perhitungan probabilitas yang menggunakan konsep pendekatan Bayesian. Tahapan dalam metode Naive Bayes, yakni:

1. Penentuan kategori data

Naive Bayes melakukan klasifikasi pada nilai probabilitas p(C=ci | D=dj) yaitu probabilitas ci dan data dj. Klasifikasi dilakukan agar dapat menentukan c ɛ C dari data d ɛ D dimana C={c1, c2, c3, ..., ci} dan D={d1, d2, d3, d... dj} (Trisedya & Jais, 2009). Penentuan dari kategori sebuah data dilakukan dengan mencari nilai maksimal dari p(C=ci | D=dj) pada P={ p(C=ci | D=dj) | c ɛ C dan d ɛ D}. Nilai probabilitas p(C=ci | D=dj) dapat dihitung dengan persamaan Mitchell tahun 2005:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | .....(1) |

Dengan p(D=dj | C=ci) merupakan nilai probabilitas dari kemunculan dokumen dj jika diketahui data tersebut berkategori ci , p(C=ci) adalah nilai probabilitas kemunculan kategori ci dan p(D= dj) adalah nilai probabilitas kemunculan dokumen dj.

1. Menghitung probabilitas

Naive Bayes menganggap sebuah data sebagai sekumpulan dari kata-kata yang menyusun data tersebut, dan tidak memperhatikan urutan kemunculan kata pada data. Sehingga perhitungan probabilitas p(D=dj | C=ci) dapat dianggap sebagai hasil perkalian dari probabilitas kemunculan kata-kata pada data dj. Perhitungan probabilitas p(C=ci | D=dj) dapat dituliskan sebagai berikut:

|  |  |  |
| --- | --- | --- |
|  |  | .....(2) |
|  |  |  |

Dengan adalah hasil perkalian dari probabilitas kemunculan semua kata pada data dj.

1. Proses klasifikasi

Proses klasifikasi dilakukan dengan membuat model probabilistik dari data training, yaitu dengan menghitung nilai p(wk | c). Untuk wkj diskrit dengan wkj ɛ V = {v1, v2, v3, ..., vm} maka nilai p(wk | c) dicari untuk seluruh kemungkinan nilai wkj dan didapatkan dengan melakukan perhitungan:

|  |  |  |
| --- | --- | --- |
|  |  | dan |

Dengan Db(wk = wkj, c) adalah fungsi yang mengembalikan jumlah dokumen b pada kategori c yang memiliki nilai kata wk = wkj, Db(c) adalah fungsi yang mengembalikan jumlah dokumen b yang memiliki kategori c dan |D| adalah jumlah selurung training data. Persamaan Db(wk = wkj, c) sering kali dikombinasikan dengan Laplacion Smoothing untuk mencegah persamaan Db(wk = wkj, c) dituliskan sebagai:

|  |  |  |
| --- | --- | --- |
|  |  | .....(3) |

1. Melakukan pengkategorian nilai

Dengan |V| merupakan jumlah kemungkinan nilai dari wkj. Pemberian kategori dari sebuah data dilakukan dengan memilih nilai c yang memiliki nilai p(C=ci | D=dj) maksimum, dan dinyatakan dengan:

|  |  |  |
| --- | --- | --- |
|  |  | .....(4) |

Kategori c\* merupakan kategori yang memiliki nilai p(C=ci | D=dj) maksimum. Nilai p(D=dj) tidak mempengaruhi perbandingan karena untuk setiap kategori nilainya akan sama. P untuk setiap variabel dapat dirumuskan sebagai berikut:

E:\tesis\method\Naive_Bayes_classifier_files\96db23de1dcc85d3b7ae0681b3ce6519.png.....(5)

#### Contoh penggunaan algoritma Naives Bayes

Data software defect pada tabel dibawah akan digunakan sebagai contoh perhitungan algoritma Naives Bayes. Faktor terjadinya defect disebabkan oleh baris code, jumlah operator, level programmer yang akan digunakan untuk melakukan prediksi defect pada software. Parameter dari variabel sebagai berikut:

Tabel 2  
Contoh data prediksi dengan Naive Bayes

|  |  |  |  |
| --- | --- | --- | --- |
| **jml baris code** | **jml operator** | **lvl programmer** | **defect** |
| 500 | 100 | 1 | f |
| 542 | 130 | 2 | f |
| 550 | 150 | 3 | f |
| 558 | 170 | 7 | t |
| 575 | 150 | 4 | f |
| 592 | 190 | 6 | t |
| 592 | 165 | 5 | t |
| 600 | 180 | 7 | t |

Dari data Tabel 3, kemudian dilakukan pencarian rata-rata dan *variance* dari setiap variabel bebas (*independent*) yakni jumlah baris code, jumlah operator, level programmer pada variabel tergantung (*dependent*) yakni defect.

Tabel 3   
rata-rata dan variance dengan Naive Bayes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **rata(code)** | **variance(code)** | **rata(opt)** | **variance(opt)** | **rata(lvl)** | **variance(lvl)** |
| **t** | 537,50 | 667,67 | 137,50 | 891,67 | 3,25 | 6,916667 |
| **f** | 589,75 | 110,92 | 171,25 | 306,25 | 5,50 | 1,666667 |

Pada contoh data, probabilitas untuk defect t dan defect f adalah masing-masing ada 5 dari seluruh 10 data. Maka dari itu p(t) = 0,5 dan p(f) = 0,5.

Apabila terdapat data yang akan diuji sebagai berikut dengan jumlah baris code 600, jumlah operator 165, dan level programmer 3. Prediksi yang dilakukan terhadap data tersebut untuk menentukan variabel defect mengandung kondisi true atau false.

Dengan formula P pada variabel jumlah baris code pada variabel true menghasilkan:

P(code | t) = 0,015792835 P(code | f) = 0,002235154

P(opt | t) = 0,021509323 P(opt | f) = 0,006558072

P(lvl | t) = 0,001311554 P(lvl | f) = 0,286763398

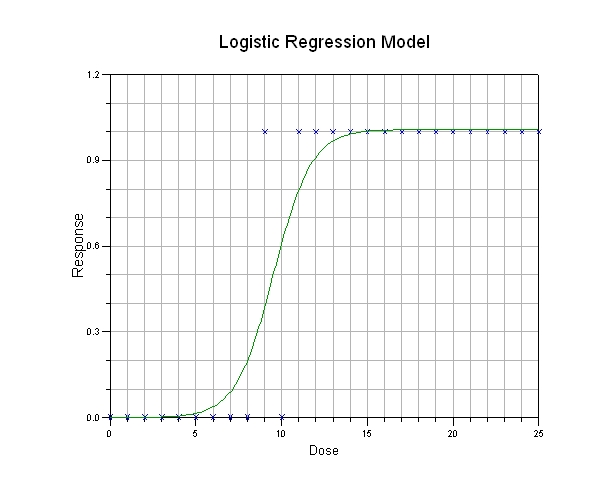
Untuk menentukan data tersebut merupakan defect t atau defect f adalah dari nilai Posterior yang paling besar. Maka dari data dengan jumlah baris code 600, jumlah operator 165 dan level programmer 3 adalah defect f (*false*).

## Logistic Regression

Logistic Regression dipresentasikan untuk prediksi dengan menggunakan lebih dari satu Linear Regression. Logistic Regression menampilkan persamaan linear yang saling berhubungan antara beberapa variabel acak, dimana variabel yang tergantung (*dependent*) adalah variabel yang berkelanjutan. Metode ini merupakan perluasan metode Linear Regression yang menggunakan lebih dari satu variabel. Dalam beberapa kasus, variabel yang tergantung menunjuk kepada dua nilai / kategori tidak dapat menggunakan Linear Regression, tetapi dapat melakukan pendekatan yang serupa yang dapat disebut juga Multiple Linear Logistic Regression. Model Logistic Regression merupakan probabilitas dari beberapa peristiwa, metode ini menggunakan fungsi linear untuk perhitungan prediksi pada beberapa variabel.

Prediksi dengan menggunakan metode Logistic Regression dapat digunakan pada variabel yang ditentukan atau varibel yang sudah dikategorikan menjadi 2 variabel. Seperti pada prediksi hidup atau mati, sakit atau tidak sakit, menang atau kalah, dan yang lainnya.

Metode Logistic Regression untuk melakukan prediksi memiliki kurva yang disebut Dose-Response Curve. Kurva ini berbentuk huruf S (Sigmoidal) yang digambarkan sebagai berikut:



Gambar 8   
Kurva Dose-Response pada metode Logistic Regression

Dimana nilai dari Response adalah antara nilai 0 dengan 1. Dan nilai dari Dose adalah dari 0 sampai 25. Nilai Dose 9 kebawah memiliki nilai Response 0. Dan nilai Dose 10 keatas memiliki nilai Response 1.

Tahapan pada metode Logistic Regression adalah sebagai berikut:

1. Penentuan bobot variabel

Metode ini diawali dengan menentukan bobot variabel. Bobot variabel akan digunakan dalam formula agar dapat menghasilkan nilai probabilitas.

1. Memasukan nilai dari variabel

Dari formula yang ada, dimasukan nilai variabel agar dapat dikalkulasi. Proses menghitung nilai dari variabel dan bobot variabel yang sudah ditentukan pada tahap sebelumnya.

1. Menghitung nilai probabilitas

Perhitungan nilai probabilitas dari formula yang ada, kemudian dihasilkan nilai probabilitas. Nilai probabilitas dapat disimpulkan bahwa dari nilai tersebut apakah menghasilkan ya atau tidak (*true* atau *false*).

Formula atau rumus dari metode Logistic Regression, sebagai berikut:

f(z) = probabilitas prediksi

β = bobot variabel

x = nilai dari variabel

n = jumlah variabel  
β0 = intercept (nilai penduga)

e = exp()

#### Contoh penggunaan algoritma Logistic Regression

Contoh perhitungan prediksi dengan logistic regression misalkan pada kasus defect software. Faktor terjadinya defect disebabkan oleh tingkat kesulitan, baris code, level programmer yang akan digunakan untuk melakukan prediksi defect pada software. Parameter dari variabel sebagai berikut:

β0 = - 5

β1 = 1.2

β2 = -1

β3 = 2

x1 = level programmer ( 1 – 10 ), dengan standar level 5

x2 = tingkat kesulitan, dengan nilai 0 untuk tidak sulit dan 1 untuk sulit

x3 = baris code, dengan standar 50

Kemudian dengan formulasi yang sudah ditentukan untuk kasus ini dengan rumus:

Apabila diuji coba pada contoh pada software yang dikerjakan oleh programmer dengan level 7, tingkat kesulitan software yang tidak sulit, dan jumlah code yang telah dihasilkan programmer tersebut adalah 50 baris maka dapat diprediksikan pada software tersebut.

= -5 + (1,2)( 7 – 5 ) + (-1)(0) + (2)(50 – 50)

= -2,6

Resiko defect yang akan terjadi sebesar 0,07 atau 7% pada software yang dikerjakan oleh programmer tersebut. Karena persentasenya sangat kecil, dapat disimpulkan tidak terjadi defect pada software tersebut.

## Neural Network

*Neural Network* adalah *processor* yang terdistribusi paralel, terbuat dari unit-unit yang sederhana, dan memiliki kemampuan untuk menyimpan pengetahuan yang diperoleh secara eksperimental dan siap pakai untuk berbagai tujuan.*Neural Network* ini meniru otak manusia dari sudut:

* Pengetahuan diperoleh oleh *network* dari lingkungan, melalui suatu proses pembelajaran.
* Kekuatan koneksi antar unit yang disebut *synaptic weights*, berfungsi untuk menyimpan pengetahuan yang telah diperoleh oleh *network* tersebut.

Pada tahun 1943, Mc. Culloch dan Pitts memperkenalkan model matematika yang merupakan penyederhanaan dari struktur sel saraf yang sebenarnya

Korelasi antara ketiga komponen pada persamaan di atas yaitu:

Signal x berupa vektor berdimensi n (x1 , x2 ,…,xn) T akan mengalami penguatan oleh *synapse* w (w1, w2 ,…,wn)T. Selanjutnya akumulasi dari penguatan tersebut akan mengalami transformasi oleh fungsi aktifasi f. Fungsi f ini akan memonitor, bila akumulasi penguatan signal itu telah melebihi batas tertentu, maka sel *neuron* yang semula berada dalam kondisi “0”, akan mengeluarkan signal “1”. Berdasarkan nilai *output* tersebut (y), sebuah *neuron* dapat berada dalam dua status: “0” atau “1”. *Neuron* disebut dalam kondisi *firing* bila menghasilkan output bernilai “1”.



**Gambar 9  
Mc. Culloch and Pitts Neuron Model**

Gambar 2-5 di atas memperlihatkan bahwa sebuah *neuron* memiliki tiga komponen:

1. *synapse* (w1 , w2 ,…,wn) T

*2. alat penambah (adder)*

3. *fun*gsi aktifasi (f)

Sebuah *Neural Network* dapat dianalisa dari dua sisi:

1. Bagaimana *neuron-neuron* tersebut dirangkaikan dalam suatu jaringan (arsitektur)

2. Bagaimana jaringan tersebut dilatih agar memberikan *output* sesuai dengan yang dikehendaki (algoritma pembelajaran). Algoritma pembelajaran ini menentukan cara bagaimana nilai penguatan yang optimal diperoleh secara otomatis.

Berdasarkan arsitekturnya, *Neural Network* dapat dikategorikan, antara lain, *single layer Neural Network*, *multi layer Neural Network*, dan *recurrent Neural Network*. Berbagai algoritma pembelajaran antara lain *Hebb’s law*, *Delta rule*, *Backpropagation algorithm*, dan *Self Organizing Feature Map*. Berawal dari diperkenalkannya model matematika *neuron* oleh Mc. Culloch dan Pitts, penelitian di bidang *Neural Network* berkembang cukup pesat, dan mencapai puncak keemasan pertama pada era tahun 60, dan puncak kedua pada pertengahan tahun 80-an.

Neuron adalah unit pemrosesan informasi yang merupakan dasar dari operasi jaringan syaraf tiruan.Sel – sel syaraf tiruan ini dirancang berdasarkan sifat – sifat dari neuron biologis.Sel syaraf tiruan ini biasa disebut juga sebagai *processing elements, unit atau node.*

Hubungan antar node diasosiasikan dengan suatu nilai yang disebut dengan bobot atau weight. Setiap node pasti memiliki output, error dan weightnya masing - masing. Output merupakan keluaran dari suatu node. Error merupakan tingkat kesalahan yang terdapat dalam suatu node dari proses yang dilakukan. Weight merupakan bobot dari node tersebut ke node yang lain pada layer yang berbeda. Nilai weight berkisar antara -1 dan 1. Bobot – bobot atau weight yang tersimpan di dalam jaringan syaraf tiruan ini disebut sebagai bobot interkoneksi. Nilai bobot yang baik akan memberikan keluaran yang sesuai, dalam arti mendekati keluaran yang diharapkan (target output) untuk suatu input yang diberikan. Bobot awal dalam suatu jaringan syaraf tiruan biasanya diperoleh secara random dan sebaiknya di inisialisasi dengan nilai yang relatif kecil, yaitu berkisar antara -0,1 sampai 0,1. Baru dalam tahap  
pelatihan, bobot tersebut akan mengalami penyesuaian melalui suatu proses perhitungan matematik agar tercapai nilai bobot yang sesuai.

Pada dasarnya untuk membentuk suatu sistem neural, hanya diperlukan 3 *tahap, yaitu forward phase, backward propagation, dan update weight.*

1. Node

Node adalah sebuah sel neuron yang di setiap nodenya memiliki output, error, dan weight. Jadi di setiap node, dimanapun itu pasti memiliki ketiga unsur tersebut. Output merupakan keluaran (hasil) dari suatu node. Error merupakan tingkat kesalahan yang terdapat dalam suatu node dari proses yang dilakukan.

1. Input Hidden Layer

Input layer merupakan layer tempat sebuah input dimasukkan(inisialisasi input), dan dari layer ini dilakukan proses-proses selanjutnya. Hidden layer disini berfungsi untuk membantu proses, semakinbanyak hidden layer yang digunakan maka semakin bagus dan semakincepat pula didapat output yang diinginkan, tetapi waktu training akan semakin lama.

1. Forward Propagation

Forward propagation bertujuan untuk menentukan output dari suatu node. Output yang dimaksud di sini adalah output dari output layer.  
Karena masing-masing node tersebut memiliki output.

**Gambar 10  
Neural Network**

Komponen dalam Neural Network :

Layer

Secara umum, setiap *Neural network* memiliki setidaknya tiga lapisan: *Input Layer, Hidden Layer*, dan *Output Layer*. Meskipun hanya ada satu *input layer* dan satu *output layer*, akan ada beberapa *hidden layer*. Lapisan pertama adalah *Input Layer*. Jumlah neuron pada *Input Layer* sama dengan nomor input dari sistem. Setiap neuron sesuai dengan salah satu input.

Lapisan kedua adalah *Hidden Layer*. Lapisan ini melakukan pengolahan menengah. angka tersebut neuron di setiap *Hidden Layer* dapat bervariasi. Biasanya jumlah neuron di lapisan ini lebih besar dari jumlah neuron di kedua lapisan input dan lapisan output. Jumlah *Hidden Layer* biasanya di simpan untuk salah satu untuk mengendalikan kompleksitas.

Lapisan terakhir adalah *Output Layer.* Dalam lapisan ini, jumlah neuron adalah sama dengan jumlah output sesuai dengan yang diberikan sistem ini. Setiap neuron sesuai dengan output tertentu di sistem. Setiap representasi node pada gambar itu sendiri merupakan neuron buatan. Masing-masing node tidak bekerja penambahan agregat dan aktivasi. Hal ini dapat dilihat pada Gambar 10

*Weights*

*Weight* merupakan berat dari node tersebut ke node yang lain (beda layer), besarnya weight adalah berkisar antara –1 sampai dengan 1. Seperti yang ditunjukan pada Gambar 11 dibawah ini.

Weight

Out/Err

**Gambar 11  
Node (Sel Neuron)**

Proses belajar suatu neural network terdiri dari proses *Forward, Backward*, dan *Update Weight*. Sekali melewati 3 tahap itu disebut dengan 1 kali training (1 *cycle).* Semakin banyak training yang dilakukan maka akan semakin kecil pula tingkat error yang dihasilkan di output layer-nya, dan dengan demikian semakin kecil juga error suatu sistem. Ada dua metode learning dalam *neural network*, yaitu :

1. ***Supervised Learning***

*Supervised Learning* adalah suatu metode dimana *neural network* belajar dari pasangan data input dan target, pasangan ini disebut training pair. Biasanya jaringan dilatih dengan sejumlah training pair, dimana suatu input vektor diaplikasikan, menghasilkan nilai di output, lalu hasil di output tersebut akan dibandingkan dengan target output. Selisihnya akan dikembalikan ke jaringan, dihitung error-nya, melalui error ini akan didapatkan selisih yang terdapat di weight-nya. Maka itu terdapat weight baru yang cenderung memiliki error yang lebih kecil, jadi akan didapat error yang lebih minimum dari error yang pertama. Vektor – vektor dalam training set diaplikasikan seluruhnya secara berurutan, error dihitung, weight disesuaikan sampai seluruh training set menghasilkan error yang sekecil – kecilnya. Sebenarnya konsep ini belajar dengan menggunakan konsep human brain. Model Neural Network yang menggunakan metode supervised learning diantaranya adalah sebagai berikut:

1. *Model Back Propagation*
2. *Model Biderectional Associative Memory*
3. *Hopfield Network*
4. ***Non-Supervised (Unsupervised) Learning***

*Unsupervised Learning* dianggap sebagai model dalam konsep sistem biologis. Teori ini dikembangkan oleh Kohonen (1984) dan beberapa ilmuwan lainnya. Dalam *unsupervised learning* tidak diperlukan target output, training hanya terdiri dari vektor – vektor input,  
tanpa pasangan target. Algoritma training merubah weight jaringan untuk menghasilkan output yang konsisten. Aplikasi dari vektor – vektor yang cukup serupa akan menghasilkan pola output yang sama. Dengan demikian proses training akan menghasilkan sifat – sifat statistik dalam  
bentuk pengelompokan vektor – vektor dalam beberapa kelas. Dengan mengaplikasikan suatu vektor dari suatu kelas sebagai input akan menghasilkan vektor output yang spesifik.

*Multi Layer Neural Network adalah neural network yang* memiliki karakteristik multi layer dimana setiap node pada suatu layer terhubung dengan setiap node pada layer di depannya. Ber-arsitektur umpan maju atau (feed forward network), menggunakan metode *supervised learning.*

Model ini merupakan model yang paling sering dipakai dalam pengembangan sistem neural dan memiliki kinerja yang sangat baik dalam sisi keakuratan.Model ini mempunyai dua fase dalam pelatihannya yaitu fase forward dan backward. Cara kerja jaringan ini adalah, setelah input masuk ke input layer maka data akan diolah dan diteruskan ke masing – masing bagian di depannya sampai ke output layer. Nilai di output layer akan dibandingkan dengan nilai target, lalu akan dihasilkan sinyal error bagi masing node di output layer. Sinyal ini ditransmisikan balik (back propagation) dari lapisan keluaran ke masing – masing sel pada lapisan sebelumnya.



**Gambar 12  
Multilayer Network**

Dewasa ini, *Neural Network* telah diaplikasikan di berbagai bidang. Hal ini dikarenakan *Neural Network* memiliki kelebihan-kelebihan sebagai berikut:

* Dapat memecahkan problema *non-linear* yang umum dijumpai di aplikasi
* Kemampuan memberikan jawaban terhadap *pattern* yang belum pernah dipelajari (*generalization)*

Dapat secara otomatis mempelajari data numerik yang diajarkan pada jaringan tersebut.

Algoritma pemrogramannya adalah:

**Tahap 1**

Membentuk model *neural network* beserta bobot setiap node. Tambahkan juga node bias dan bobot kedalam node bias. Bobot ditentukan secara acak, namun diusahakan agar bobot memiliki nilai kecil dengan nilai positif atau negative antara 0 sampai 2 (Larose, 2005, p. 128). Pemberian bobot dengan nilai kecil agar proses pembelajaran menjadi lebih mudah. Karena nilai momentum dan *learning rate* berada diantara 0 sampai 1.

**Tahap 2**

Mengambil nilai atribut dan class pada data yang akan dijadikan data training. Lalu menghitung bobot keluaran jaringan neuron        

Fungsi f yang disebut juga fungsi aktivasi *sigmoid* digunakan agar hasil perhitungan tetap pada nilai 0 sampai 1. Fungsi f sendiri dijabarkan dalam persamaan berikut:

Nilai e merupakan bilangan natural dengan nilai konstan yaitu 2.718281828, x merupakan nilai hasil penjumlahan dari bobot dikalikan dengan masukan attribute.

**Tahap 3**

Tahap berikutnya disebut tahap propagasi mundur (*back propagation*) Tahap ini membandingkan nilai output yang dihasil dengan dengan nilai output sesungguhnya, perbedaan nilai antara kenyataan dengan hasil perhitungan disebut dengan error, dan dari error ini bobot setiap node akan disesuaikan agar model menjadi lebih baik. Perhitungan bobot dilakukan dengan persamaan :

Dimana merupakan nilai perubahan yang didapat dari kalkulasi *learning rate* dengan (delta) dikalikan dengan nilai masukan. Jika ada nilai momentum, nilai momentum akan dikalikan dengan nilai perubahan sebelumnya. Persamaan dapat dilihat pada persamaan berikut:

Perhitungan (delta) dibedakan menjadi dua macam, untuk *output layer* dan untuk *hidden* *layer*, untuk *hidden layer* perhitungan juga menambahkan nilai delta dari *output layer.*

merupakan hasil penjumlahan antara nilai node setelahnya dikalikan dengan bobot node yang dihitung dalam node setelahnya ditambahkan dengan nilai sebelumnya.

Perhitungan untuk bobot bias juga menggunakan perhitungan yang sama, namun tidak menggunakan momentum, serta nilai bias yang konstan dengan nilai satu sehingga menjadi:

**Tahap 4**

Setelah semua bobot baru diketahui, tahap dua dijalankan kembali. sampai pada jumlah pengulangan yang telah ditentukan.

#### Contoh penggunaan Neural Network

Studi kasus untuk pembuatan model neural network akan dilakukan pada dataset yang terdiri dari 3 attribut yang merupakan attribut warna, dan kelas yang mewakili warna tulisan. Attribut warna merupakan warna latar yang dibuat secara acak, dan warna text merupakan warna text (hitam atau putih) yang akan dicetak pada warna latar tersebut. Pemilihan warna hitam atau putih didasarkan dari kemudahan membaca pada warna attributnya. Contoh data dapat dibaca pada tabel 4 seperti dibawah ini:

Tabel 4   
Data pengenalan warna

|  |  |  |  |
| --- | --- | --- | --- |
| R | G | B | Text |
| 20 | 20 | 20 | 1 |
| 255 | 0 | 0 | 1 |
| 140 | 200 | 150 | 0 |
| 255 | 255 | 255 | 0 |

Dari data tersebut, *hidden layer* akan dikalkulasikan dengan perhitungan dengan ( jumlah attribut + hasil ) / 2 + 1 yaitu 3 hidden layer, bobot untuk *hidden layer* pada awalnya diberikan secara acak, model dengan bobot yang sudah diberikan secara acak dapat dilihat pada gambar 13. Model akan dibentuk dengan momentum sebesar 0.1 dan learning rate 0.3.



Gambar 13 *Prototype* Model awal

Setiap data pada dataset kemudian diterapkan kedalam model yang terbentuk, setelah itu dihitung nilai setiap node *hidden layer* dan node *output*. Contoh penghitungan dimulai dari data pertama dengan nilai R 20, G 20, dan B 20. Pada hidden layer node pertama nilai y adalah:

Karena nilai untuk node *hidden layer* kedua melebihi nilai 1, maka fungsi f membuat nilai tersebut menjadi 0.224, demikian pula untuk node ke 3 *hidden layer*. Setelah semua node *hidden layer* dihitung, Nilai untuk output dihitung berdasarkan bobot node output dengan nilai dari *hidden layer*. Perhitungan dapat dililhat pada perhitungan berikut:

Seperti terlihat pada Gambar 14 dibawah ini:



Gambar 14  
Nilai untuk data percobaan pertama

Setelah diketahui nilai node output, nilai ini dibandingkan dengan nilai output sesungguhnya dan ditambahkan dengan nilai error terdahulu untuk mendapatkan nilai error, setelah didapat nilai *error*, nilai error ini akan dijumlahkan dengan nilai *error* sebelumnya dan akan dijadikan untuk menghitung nilai delta. Karena model saat ini berada pada tahap pertama, maka error mula mula bernilai kosong.



Gambar 15  
Error dan nilai delta untuk setiap node.

Setelah didapatkan nilai error dan nilai delta, bobot untuk setiap node disesuaikan kembali, Untuk bobot dari attribute R ke hidden layer node pertama, bobot attribut pertama dihitung, nilai perubahan sebelumnya adalah kosong, lalu dihitung nilai perubahan baru dan didapat hasil 0.014. Nilai perubahan ini ditentukan oleh *momentum* dan *learning rate*, untuk tahap pertama ini *learning rate* tidak berpengaruh karena perubahan sebelumnya masih kosong. Nilai perubahan ini ditambahkan dengan nilai bobot sebelumnya untuk menemukan nilai bobot baru. proses penyesuaian dapat dilihat pada perhitungan berikut:



Gambar 16  
Bobot setelah training data pertama

Perhitungan dilanjutkan dengan menerapkan model yang sudah disesuaikan pada data kedua. Data kedua dimasukkan kedalam variable input, lalu disesuaikan bobotnya sampai menjadi output, kemudian dihitung nilai error, dan didapatkan nilai delta untuk output, kemudian nilai delta disesuaikan kedaam node hidden layer dan kemudian dihitung bobot baru, keseluruhan proses pembelajaran data kedua digambarkan pada Gambar 2.13



Gambar 17  
Detail Error, Delta dan bobot setiap node setelah data kedua

Perhitungan dilanjutkan dengan data ketiga, dan seterusnya. Setelah semua data dihitung. Model akan dihitung kembali dengan menggunakan data pertama sebanyak jumlah perulangan yang diperbolehkan, batasan nilai perulangan ini dapat diatur dengan parameter *maximum\_epoch*. Setelah model yang terbentuk tidak mengalami perubahan selama beberapa putaran, atau jika batas maksimum perulangan telah dicapai, model akan selesai terbentuk dan dapat diujikan pada data baru.

#### Pemilihan Parameter

Pemilihan parameter memainkan peran sentral dalam *machine learning*. Ide utama dari pemilihan parameter adalah memilih subset dari parameter yang relevan untuk membangun model. Ada  
banyak potensi manfaat dari pemilihan parameter yaitu memfasilitasi visualisasi data dan pemahaman data, mengurangi ukuran dan persyaratan penyimpanan, mengurangi pelatihan dan pemanfaatan waktu, menentang kutukan dimensi untuk meningkatkan kinerja dari prediksi. Ada beberapa teknik parameter seleksi yang bersifat *adhoc*, tetapi ada juga metode pendekatan yang lebih perspektif.

Dari sudut pandang perspektif, dapat ditunjukkan bahwa pemilihan parameter yang optimal memerlukan pencarian yang lengkap dari semua bagian himpunan yang mungkin dari parameter. Jika jumlah parameter yang tersedia cukup besar, hal ini tidak praktis. Untuk mesin pembelajaran, pencarian adalah satu set untuk meningkatkan parameter bukan sebagai set optimasi. Banyak sekali penelitian dalam mesin belajar telah difokuskan pada pengembangan algoritma pemilihan parameter otomatis dan dataset dengan puluhan atau ratusan ribu variabel yang tersedia.

# Algoritma Optimasi

Optimasi adalah tentang menemukan nilai-nilai parameter yang optimal untuk suatu objek atau sistem yang meminimalkan tujuan (biaya) dan fungsi. Dalam optimasi, kita diberi sebuah fungsi, yang dikenal sebagai fungsi tujuan. Tujuannya adalah untuk meminimalkan atau memaksimalkan nilai dari fungsi tujuan dengan menyesuaikan parameter berbagai. Masing-masing kombinasi parameter menandai solusi yang mungkin baik atau buruk, tergantung pada nilai fungsi tujuan. Soft-teknik komputasi menghasilkan set parameter terbaik yang memberikan nilai-nilai terbaik dari fungsi tujuan mengingat kendala dari waktu.

## Algoritma Genetika

Menurut Han (Han & Kamber, Data Mining: Concepts and Techniques 2nd Edition, 2007, hal. 351), metode Genetic Algorithm merupakan gabungan atau kumpulan dari konsep evolusi alam, yang biasanya digunakan untuk inisialiasi individu secara aturan acak pada populasi. Teknik Genetic Algorithm menampilkan ulang teknik identifikasi untuk optimisasi dalam masalah pencarian (Gorunescu, 2011, hal. 289). Genetic Algorithm juga merupakan bagian dari Evolutionary Algorithm. Metode ini menggunakan proses seleksi secara acak seperti hal proses evolusi. Metode yang sering digunakan untuk meyelesaikan masalah optimisasi yang kompleks dan sulit dengan metode yang konvensional.

Genetic Algortihm biasa digunakan untuk klasifikasi dan juga masalah optimisasi (Han & Kamber, Data Mining: Concepts and Techniques 2nd Edition, 2007, hal. 351). Dalam data mining, metode ini juga digunakan untuk melakukan evaluasi terhadap nilai fitness pada sebuah algoritma. Nilai fitness yang semakin besar akan menghasilkan optimisasi yang lebih baik. Metode ini diawali dengan dataset solusi atau populasi. Individual pada suatu populasi merepresentasikan solusi untuk meyelesaikan suatu masalah. Individu pada suatu populasi disebut chromosom, yang merepresentasikan suatu solusi atas permasalahan. Chromosom akan berevolusi dengan cara perulangan yang berkelanjutan, yang disebut generasi. Selama tiap generasi, chromosom dievaluasi menggunakan nilai *fitness*  yang paling tinggi. Chromosom dengan fitness yang tinggi akan menghasilkan probabilitas yang tinggi pula. Generasi berikutnya atau chromosom baru yang dihasilkan disebut *offspring*. *Offspring* dapat dibentuk melalui penyatuan dua chromosom dari generasi awal menggunakan perkawinan silang (*crossover*) atau hasil modifikasi chromosom menggunakan mutasi (*mutation*). Generasi baru yang dibentuk melalui proses seleksi induk (*parent*) dan anak (*offspring*), berdasarkan dengan nilai *fitness*, dan juga melalui eliminasi chromosom lainnya agar ukuran populasi tidak berubah atau konstan. Chromosom yang sesuai dengan tujuan memiliki kemungkinan tertinggi untuk dipilih. Setelah beberapa generasi, Genetic Algorithm menghasilkan chromosom-chromosom terbaik yang diharapkan dapat menjadi solusi optimal atau suboptimal atas permasalahan untuk optimisasi.

Langkah-langkah dalam Genetic Algorithm sebagai berikut:

**begin**

x ← 0;

initialize P(x);

evaluate P(x);

**while** (not termination condition) **do**

recombine P(x) to yield O(x);

evaluate O(x);

select P(x + ) from P(x) and O(x);

x++;

**end**

**end**

Dimana: x = suatu generasi

P(x) = Chromosom parent

O(x) = Chromosom Offspring

Crossover merupakan operator kombinasi terhadap chromosom, yang dioperasikan pada dua chromosom pada suatu waktu dan menghasilkan offspring dengan mengkombinasikan pola-pola kedua chromosom. metodenya adalah dengan membangkitkan suatu nilai secara acak yang mana besar maksimal nilai tersebut tidak lebih dari panjang chromosom. Nilai tersebut menunjukan titik pertukaran. Seluruh chromosom dari kedua induk setelah titik tersebut kemudian saling ditukar. Cara termudah untuk melakukan crossover adalah dengan pemilihan secara acak dan membentuk *offspring* dengan mengkombinasikan segmen sebelah kiri dari seleksi salah satu parent dengan segmen sebelah kanan dari seleksi parent lainnya. Kinerja metode Genetic Algorithm tergantung atas proses operator crossover yang digunakan. Setelah melakukan proses crossover, *offspring* akan masuk ke dalam proses mutasi, dimana ada proses perubahan terhadap gen pada chromosom. Mekanisme perubahan susunan gen akibat adanya faktor alam yang disebut dengan mutasi. Mutasi direpresentasikan sebagai proses berubahnya satu atau lebih nilai gen dalam chromosom dengan suatu nilai acak. Semua proses tersebut akan berulang secara terus-menerus sampai menghasilkan chromosom yang paling bagus.

Tahapan yang diperlukan dalam optimisasi dengan Genetic Algorithm adalah dengan:

1. Menentukan populasi awal

Diperlukan populasi awal sebelum optimisasi dilakukan. Populasi awal dilakukan seperti melakukan pendeklarasian dalam sebuah program.

1. Evaluasi nilai fitness

Nilai fitness dievaluasi dengan tujuan agar setiap chromosom memiliki nilai baik atau tidak. Kemudian semua nilai fitnes ditentukan probabilitasnya masing-masing. Dari hasil probabilitas tertinggi, dihasilkan bahwa chromosom 1 mempunyai nilai fitness paling tinggi. Maka chromosom 1 juga mempunyai kesempatan paling besar dalam proses seleksi selanjutnya dengan Roullete Wheel.

1. Seleksi Chromosom induk

Proses dimana dipilih chromosom yang akan dijadikan chromosom induk dalam populasi yang akan dihitung. Proses seleksi chromosom yang digunakan dengan Roullete Wheel.

1. Melakukan crossover (perkawinan silang)

Dalam crossover juga melanjutkan ke langkah selanjutnya menggunakan bilangan acak R antara 0 sampai 1. Setelah melakukan pemilihan parent, proses selanjutnya adalah menentukan posisi crossover. Setelah didapatkan posisi crossover maka chromosom parent akan dipotong mulai gen posisi crossover kemudian potongan gen tersebut saling ditukarkan antar parent.

1. Mutasi Chromosom

Jumlah chromosom yang mengalami mutasi dalam satu populasi ditentukan oleh persentase *p mutation*. Proses mutasi dilakukan dengan cara mengganti satu gen yang terpilih secara acak dengan suatu nilai baru yang didapat secara acak. Chromosom tersebut kemudian diuji bila belum sesuai tujuan, maka populasi ini belum memiliki chromosom yang ingin dicapai. Chromosom-chromosom pada populasi ini akan mengalami proses yang sama seperti generasi sebelumnya yaitu proses evaluasi, seleksi, crossover dan mutasi yang kemudian akan menghasilkan chromosom-chromosom baru untuk generasi yang selanjutnya. Proses ini akan berulang sampai sejumlah generasi yang telah ditetapkan sebelumnya.

Seleksi Roda Rolet atau *Roulette Wheel Selection* merupakan metode seleksi yang digunakan pada Genetic Algorithm. Metode roda rolet bersifat proporsional yang mana besarnya kemungkinan suatu individu terseleksi adalah dengan membandingkan *fitness* individu tersebut dengan total *fitness* yang ada pada populasi. Merupakan metode yang paling banyak digunakan dan dapat diilustrasikan. Kemungkinan masing-masing chromosom dapat dicari dengan membagi *fitness* masing-masing chromosom dengan total *fitness* dalam populasi. Seleksi dengan roda rolet ini adalah berdasarkan skala *fitness*, semakin besar nilai fitness dari suatu chromosom maka semakin besar kemungkinan chromosom itu terpilih. Karena terpilihnya suatu chromosom dalam populasi untuk dapat bereproduksi adalah sebanding dengan *fitness* chromosom tersebut, maka akan terjadi kecenderungan chromosom dengan *fitness* yang baik akan berlipat ganda dalam iterasi yang cukup cepat sehingga mencegah algortima genetik untuk melakukan eksplorasi lebih lanjut. Misalnya, jika persentasi *fitness* salah satu chromosom adalah 90% dari keseluruhan roda rolet, maka pada iterasi selanjutnya seluruh populasi akan didominasi oleh chromosom yang sama dikarenakan chromosom-chromosom yang terpilih untuk menjadi induk hampir selalu sama dan hasil persilangan dari dua chromosom induk yang sama adalah chromosom induk itu sendiri.

Gambar 2.7  
Roulette Wheel

Contoh Optimisasi dengan Genetic Algorithm dalam optimisasi prediksi *Software Defect*, bila prediksi *Software Defect* berdasarkan dari 3 variabel (level programmer, jumlah operator, dan jumlah baris code). Dihasilkan rumus 3a+2b+c=1000, dimana:

1. variabel a sebagai level programmer,
2. variabel b sebagai jumlah operator,
3. variabel c sebagai jumlah baris code

variabel a paling berpengaruh, dan variabel c kurang berpengaruh terhadap hasil yang akan dicapai. Berdasarkan tahapan dalam metode Genetic Algorithm berlangsung sebagai berikut:

1. Menentukan populasi awal

C[1] 🡪 {6 , 185 , 700}

C[2] 🡪 {5 , 165 , 590}

C[3] 🡪 {1 , 100 , 500}

C[4] 🡪 {3 , 150 , 550}

C[5] 🡪 {4 , 150 , 575}

1. Evaluasi nilai fitness

Dengan rumus yang sudah ditentukan sebelumnya 3a+2b+c=1000 maka akan menghasilkan nilai fitness pada setiap chromosom.

Fitness C[1] = 1 / 1 + | (3a+2b+c) – 1000 |

= 1 / 1 + | ((3\*6)+(2\*185)+590) – 1000 |

= 1 / 1 + 22

= 0,043

Fitness C[2] = 1 / 1 + | (3a+2b+c) – 1000 |

= 1 / 1 + | ((3\*5)+(2\*165)+590) – 1000 |

= 1 / 1 + 65

= 0,015

Fitness C[3] = 1 / 1 + | (3a+2b+c) – 1000 |

= 1 / 1 + | ((3\*1)+(2\*100)+500) – 1000 |

= 1 / 1 + 297

= 0,003

Fitness C[4] = 1 / 1 + | (3a+2b+c) – 1000 |

= 1 / 1 + | ((3\*3)+(2\*150)+550) – 1000 |

= 1 / 1 + 141

= 0,007

Fitness C[5] = 1 / 1 + | (3a+2b+c) – 1000 |

= 1 / 1 + | ((3\*4)+(2\*150)+575) – 1000 |

= 1 / 1 + 113

= 0,008

Total fitness = (0,043+0,015+0,003+0,007+0,008) = 0,077

Probabilitas masing-masing chromosom menjadi:

P C[1] = 0,043 / 0.077 = 0,558

P C[2] = 0,015 / 0.077 = 0,194

P C[3] = 0,003 / 0.077 = 0,038

P C[4] = 0,007 / 0.077 = 0,090

P C[5] = 0,008 / 0.077 = 0,103

Dari hasil probabilitas tertinggi, dihasilkan bahwa chromosom 1 mempunyai nilai fitness paling tinggi. Maka chromosom 1 juga mempunyai kesempatan paling besar dalam proses seleksi selanjutnya dengan Roullete Wheel.

1. Seleksi Chromosom induk

Untuk proses seleksi digunakan Roullete Wheel, untuk itu diperlukan nilai kumulatif probabilitasnya dari setiap chromosom, yakni sebagai berikut:

C[1] = 0,558 = 0,558

C[2] = 0,558 + 0,194 = 0,762

C[3] = 0,558 + 0,194 + 0,038 = 0,800

C[4] = 0,558 + 0,194 + 0,038 + 0,090 = 0,890

C[5] = 0,558 + 0,194 + 0,038 + 0,090 + 0,103 = 1

Langkah selanjutnya dengan menggunakan bilangan acak R antara 0 sampai 1. Bilangan acak yang digunakan sebanyak jumlah chromosom, misalkan sebagai beikut:

R[1] = 0,793

R[2] = 0,396

R[3] = 0,376

R[4] = 0,904

R[5] = 0,622

Memilih chromosom ke x sebagai parent dengan syarat C[x-1] < R < C[x]. Angka acak R[1] < nilai kumulatif dari C3, sehingga C1 nanti akan dilakukan crossover dengan C3. Hasil seleksi Roullete Wheel pada populasi ini untuk crossover menjadi:

C[1] menjadi C[3] 🡪 {1 , 100 , 500}

C[2] menjadi C[1] 🡪 {7 , 180 , 600}

C[3] menjadi C[1] 🡪 {7 , 180 , 600}

C[4] menjadi C[5] 🡪 {4 , 150 , 575}

C[5] menjadi C[2] 🡪 {5 , 165 , 590}

1. Melakukan crossover (perkawinan silang)

Dalam crossover juga melanjutkan ke langkah selanjutnya menggunakan bilangan acak R antara 0 sampai 1. Bilangan acak yang digunakan sebanyak jumlah chromosom, misalkan sebagai beikut:

R[1] = 0,937

R[2] = 0,236

R[3] = 0,076

R[4] = 0,841

R[5] = 0,469

Misal kita tentukan crossover probability (pr) adalah sebesar 0,5 = 50%, maka diharapkan dalam satu generasi ada 2 atau 3 chromosom dari satu generasi mengalami proses crossover. Maka chromosom ke y akan dipilih sebagai induk jika R[y] < ρr, dari bilangan acak R diatas maka yang dijadikan parent, yaitu C2, C3 dan C5.

Setelah melakukan pemilihan parent, proses selanjutnya adalah menentukan posisi crossover. Ini dilakukan dengan cara menggunakan bilangan acak dengan batasan 1 sampai panjang chromosom, dalam kasus ini bilangan acak yang dibangkitkan adalah 1 – 3. Misalkan didapatkan posisi crossover adalah 1 maka chromosom parent akan dipotong mulai gen ke 1 kemudian potongan gen tersebut saling ditukarkan antar parent.

C[2] >< C[3]

C[3] >< C[5]

C[5] >< C[2]

Posisi cut-point crossover dipilih menggunakan bilangan acak 1-3 sebanyak jumlah crossover yang terjadi, misal

cp(C[2]) = 2

cp(C[3]) = 3

cp(C[5]) = 1

Offspring[1] = C[2] >< C[3] dengan cp(C[2])

= [7 , 180 , 600] >< [7 , 180 , 600]

= [7 , 180 , 600]

Offspring[2] = C[3] >< C[5] dengan cp(C[3])

= [7 , 180 , 600] >< [5 , 165 , 590]

= [7 , 180 , 590]

Offspring[3] = C[5] >< C[2] dengan cp(C[5])

= [5 , 165 , 590] >< [7 , 180 , 600]

= [7 , 165 , 590]

Populasi baru terbentuk dengan hasil crossover yang telah terjadi menjadi:

C[1] 🡪 [1 , 100 , 500]

C[2] 🡪 [7 , 180 , 600]

C[3] 🡪 [7 , 180 , 590]

C[4] 🡪 [4 , 150 , 575]

C[5] 🡪 [7 , 165 , 590]

1. Mutasi Chromosom

Jumlah chromosom yang mengalami mutasi dalam satu populasi ditentukan oleh persentase *p mutation*. Proses mutasi dilakukan dengan cara mengganti satu gen yang terpilih secara acak dengan suatu nilai baru yang didapat secara acak.

Total gen = (gen dalam chromosom) \* jumlah chromosom

= 3 \* 5

= 15

Dalam menentukan posisi gen yang mengalami mutasi dilakukan dengan cara menggunakan bilangan acak antara 1 sampai total gen, yakni antara 1 sampai 15. Misal ρm kita tentukan 10% maka jumlah gen yang mengalami mutasi adalah 10% dari 15 yaitu 1,5 atau 1 gen.

Kemudian menggunakan bilangan acak dan terpilih posisi gen 12 yang mengalami mutasi. Dengan demikian yang akan mengalami mutasi adalah chromosome ke-4 gen nomor 3. Maka nilai gen pada posisi tersebut kita ganti dengan bilangan acak 0-1000. Misalkan bilangan acak yang digunakan adalah 698, maka chromosom ke-4 berubah menjadi [4 , 150 , 698]. Populasi pada generasi pertama menjadi:

C[1] 🡪 [1 , 100 , 500]

C[2] 🡪 [7 , 180 , 600]

C[3] 🡪 [7 , 180 , 590]

C[4] 🡪 [4 , 150 , 575]

C[5] 🡪 [7 , 165 , 590]

Chromosom ke-4 kemudian di uji pada rumus 3a+2b+c=1000, agar menjadi chromosom yang ingin dicapai. (3\*4)+(2\*150)+698 1000, maka populasi ini belum memiliki chromosom yang ingin dicapai. Chromosom-chromosom pada populasi ini akan mengalami proses yang sama seperti generasi sebelumnya yaitu proses evaluasi, seleksi, crossover dan mutasi yang kemudian akan menghasilkan chromosom-chromosom baru untuk generasi yang selanjutnya. Proses ini akan berulang sampai sejumlah generasi yang telah ditetapkan sebelumnya.

## Particle Swarm Optimization

*Particle Swarm Optimization* merupakan teknik optimisasi berbasis populasi yang dikembangkan oleh Eberhart dan Kennedy pada tahun 1995. PSO mensimulasikan perilaku dari sekelompok burung yang mencari makanan. Perilaku tersebut digambarkan sebagai berikut:

Sekelompok burung sedang mencari sepotong makanan didalam suatu area. Semua burung dalam kelompok tersebut tidak tahu dimana letak makanan tersebut, dalam proses pencarian makanan beberapa burung secara mendadak berpisah dari kawanan dan membentuk kawanan baru dan kembali berkelompok. Proses pengelompokkan burung ini bertujuan untuk menjaga jarak optimum antara makanan dengan burung tersebut dan kawanan burung lainnya. Cara yang paling efektif untuk menemukan potongan makanan tersebut adalah dengan mengikuti burung yang jaraknya paling dengan dengan makanan tersebut.

Perilaku yang sama juga digambarkan pada kelompok hewan lainnya seperti ikan. Dengan berkelompok, kawanan ikan mengalam persaingan dalam membagi makanan, namun kawanan tersebut juga menjadi lebih mudah untuk menemukan makanan baru didalam daerah yang persebaran makanannya tidak diketahui (Kennedy & Eberhart, 1995, p. 123). Metode tersebut diadopsi dalam algoritma PSO sendiri yaitu melakukan perulangan untuk mengoptimasi sebuah masalah dengan memberikan nilai kualitas. Dengan begitu dalam data yang besar dapat diketahui posisi terbaik dalam perhitungan pengolahan data.

Algoritma PSO menggunakan teknik perhitungan evolusioner (Hu, Eberhart, & Shi, 2003, p. 534):

**Tahap 1**

Partikel awal PSO akan didefinisikan secara acak pada awal algoritma, diasumsikan x merupakan lokasi partikel, dan v merupakan kecepatan partikel. Setiap partikel dalam data akan diukur nilai terbaiknya (*fitness*) yang merupakan representasi dari dari seberapa baiknya partikel tersebut dalam memecahkan masalah. Dalam perulangan yang terjadi, posisi partikel terbaik sebelumnya disebut Pbest dengan indeks Gbest. Setelah itu semua partikel akan bergerak kepada posisi terbaik (Pbest) dan akan memulai kumpulan yang baru. Berikut persamaan dalam mengupdate ketinggian dan kecepatan partikel :

( 2.5 )

**Tahap 2**

PSO mencari solusi yang optimum dengan memperbaharui generasi

**Tahap 3**

Perkembangan populasi berdasarkan pada generasi sebelumnya. Setiap partikel yang bukan merupakan partikel terbaik akan terbang bergerak menuju partikel terbaik, pergerakan akan dihitung dengan formula 2.6

**( 2.6 )**

Dimana w adalah nilai kualitas, c1 dan c2 adalah konstanta percepatan, Rand() nilai acak antara 0 dan 1. x k + 1 adalah nilai pada partikel yang baru, sedangkan x k adalah nilai partikel sebelum ditambah dengan *velocity*.

#### Studi Kasus

Contoh perhitungan PSO yang menggunakan 5 buah partikel. Kelima partikel akan terdiri dari dua buah input yang terdiri yang dideklarasikan secara acak ( antara 20 sampai 30), partikel tersebut akan mencari kombinasi angka yang bila dijumlahkan menghasilkan angka 40. Kelima partikel awal yang terbentuk secara acak adalah :

1. 28 + 24 = 52
2. 29 + 25 = 54
3. 22 + 27 = 49
4. **24 + 22 = 46**
5. 25 + 25 = 50

Dari kelima partikel tersebut nilai yang paling dekat dengan target adalah 46, Setiap partikel dilakukan perhitungan untuk menentukan *velocity* masing masing dengan formula (2.5). Berikut contoh perhitungan *velocity* untuk partikel pertama.

Pada perhitungan diatas 0.56 dan 0.19 yang dicetak tebal merupakan nilai yang muncul saat random, sedangkan 52 adalah nilai best partikel pertama, yang juga sama dengan nilai yang muncul saat ini, sedangkan 46 ada nilai best dari keseluruhan. Partikel yang lainnya ikut dihitung pula.

Hasil *velocity* dari partikel ke4 adalah 0 maka partikel 4 tidak bergerak, sedangkan hasil *velocity* dari partikel 2 adalah -15.98, lebih besar daripada maksimum *velocity*, oleh karena itu *velocity* partikel ke 2 dijadikan *velocity* maksimum. Nilai partikel diupdate dengan nilai *velocity* menjadi :

1. 26 + 22 = 48 (*velocity* dibulatkan menjadi -2)
2. **19 + 15 = 34** (*velocity* menggunakan *velocity* maksimum yaitu -10)
3. 21 + 26 = 47 (*velocity* dibulatkan menjadi -1)
4. **24 + 22 = 46** (tidak bergerak)
5. **23 + 23 = 46** (*velocity* dibulatkan menjadi -2)

Setelah nilai partikel berubah, ditemukan bahwa partikel dengan nilai terbaik adalah partikel pertama dan keempat dan kelima. Perhitungan *velocity* baru digunakan kepada setiap partikel dengan hasil. Namun nilai terbaik untuk setiap partikel berubah menjadi nilai baru ( karena semua nilai partikel saat ini lebih dekat kepada nilai 40), berikut perhitungan serta posisi terakhir setiap partikel

1. 21 + 22 = 42 (*velocity* dibulatkan menjadi -5)
2. 10 + 6 = 16 (*velocity* dibulatkan menjadi -9)
3. **18 + 23 = 41** (*velocity* dibulatkan menjadi -3)
4. 24 + 22 = 46 (tidak bergerak)
5. 21 + 21 = 42 (*velocity* dibulatkan menjadi -2)

Hanya salah satu nilai dari partikel satu yang bergerak, hal ini disebabkan salh satu nilainya telah sama dengan nilai pada partikel best, sehingga hanya nilai berbeda yang bergerak mengikuti *velocity*. Setelah semua partikel bergarak, kini partikel ketigalah yang paling dekat dengan target, maka perhitungan menggunakan nilai partikel ke 3 sebagai nilai terbaik dalam perhitungan, dan hasil seperti dibawah. Pada gambar 2.12 a merupakan *velocity*  awal, c menggambarkan arah terhadap partikel terbaik, dan b merupakan *velocity* akhir.



Gambar 2.12  
Pergerakan partikel 1 dan 3

1. 14 + 16 = 30 (*velocity* dibulatkan menjadi -6)
2. 20 + 16 = 36 (*velocity* menggunakan *velocity* maksimum yaitu 10)
3. 18 + 23 = 41 (*velocity* dibulatkan menjadi -3)
4. **21 + 19 = 40** (*velocity* dibulatkan menjadi -3)
5. 18 + 18 = 36 (*velocity* dibulatkan menjadi -3)

Terlihat bahwa partikel 4 mencapai target dalam 3 iterasi, partikel kedua bergerak dengan *velocity* 10 karena 10 adalah nilai maksimum *velocity*, sedangkan, partikel ketiga tidak bergerak walaupun memiliki *velocity* -3. Hal ini disebabkan kedua nilai tersebut sama dengan nilai partikel terbaik pada iterasi kedua.

# Segmentasi

Tujuan dari algoritma segmentasi adalah dengan memecahkan setiap data dalam dataset menjadi kelompok kelompok yang homogen. Kelompok data ini biasanya disebut sebagi segmen atau *cluster*. Setiap segmen yang terbentuk akan terdiri dari data yang sejenis dan berbeda dengan data pada segmen lainnya (Vercellis, 2009 : p. 298). Pengelompokkan ini sama dengan cara kerja otak manusia, dimana ilmu pengetahuan dikelompokkan dalam setiap bidangnya. Dengan adanya pengelompokkan, data yang dapat diolah dengan lebih spesifik sesuai dengan tujuan penelitian.

Pemecahan data kedalam segmen data juga diterapkan pada tahap pengolahan awal data dalam proses *data mining*, sehingga dapat diterapkan metode data mining *data mining* kedalam setiap segmen data. Proses segmentasi juga dapat mengurangi jumlah ataupun dimensi data yang diolah. Beberapa alasan dalam pemilihan algoritma segmentasi:

**Flexibilitas**

Algoritma segmentasi harus dapat mengolah berbagai jenis data, sehingga dapat diterapkan dalam ruang lingkup yang lebih besar.

**Handal**

Algoritma segmentasi harus dapat menghasilkan segmen yang stabil sehingga tidak menimbulkan banyak perbedaan dalam setiap observasi yang dilakukan. Algoritma juga tidak boleh mudah terganggu oleh data yang tidak relevan. Attribut dan ciri dari setiap segmen harus tetap stabil jika diteliti dengan cara yang sama.

**Efisiensi**

Algoritma segmentasi harus dapat mengolah data baik dalam jumlah besar dengan waktu pengolahan yang sedikit. Algoritma segmentasi juga harus dapat menyesuaikan jumlah attribute yang digunakan dalam dataset agar waktu pengolahan dapat diminimalisasi, namun tetap memiliki hasil yang konsisten.

Metode segmentasi dapat diklasifikasikan menjadi beberapa jenis berdasarkan logika segmentasi(Vercellis, 2009 : p. 302):

1. Metode partisi

Baik dalam mengelompokkan data menjadi kelompok yang sudah ditentukan lebih dahulu

1. Metode hierarki

Memecah data kedalam segmen dengan struktur hirarki. Data setiap segmen tetap homogen, namun memiliki tingkatan antara satu segmen dengan segmen lainnya

1. Metode berbasis kepadatan

Merupakan perpaduan kedua metode sebelumnya, metode ini memecah data kedalam partisi berdasarkan jarak data terhadap setiap segmen. Namun setiap segmen memiliki batasan jarak, sehingga nilai tidak boleh lebih kecil dari nilai minimum segmen.

1. Metode grid

Segmen terbentuk bersarakan sturktur ruangan yang seperti sel. Metode ini dapat mengolah data besar dengan cepat, namun memiliki akurasi yang rendah

## K-Means

*K-means* pertama kali dipublikasikan oleh Stuart Lloyd pada tahun 1984 dan merupakan algoritma clustering yang banyak digunakan. *K-means* bekerja dengan mensegmentasi objek yang ada kedalam kelompok atau yang disebut dengan segmen sehingga objek yang berada dalam masing-masing kelompok lebih serupa satu sama lain dibandingkan dengan objek dalam kelompok yang berbeda. Algoritma Clustering adalah meletakkan nilai yang serupa dalam satu segmen, dan meletakkan nilai yang berbeda dalam cluster yang berbeda (Wu & Kumar, 2009: p. 22). *K-Means* memisahkan data dengan optimal dengan perulangan yang memaksimalkan hasil dari partisi hingga tidak ada perubahan data dalam setiap segmentasi.

*K-Means* bekerja dengan pendekatan *Top-Down*karena memulai dengan segmentasi yang sudah ditentukan terlebih dahulu (Myatt, 2007: p. 121). Sehingga hasil data sebuah segmen tidak mungkin tercampur antara satu segmen dengan segmen lainnya (Xu & Wunsch II, 2009: p. 68). Pendekatan ini juga mempercepat proses komputasi untuk data dalam jumlah besar.

Algoritma *K-means* diterapkan pada objek yang diwakili dalam bentuk titik didalam ruangan vektor berdimensi-d. *K-means* mengcluster semua data didalam setiap dimensi dimana titik dalam segmentasi yang sama diberi custer ID. Nilai dari k adalah masukan dasar dari algoritma yang menentukan jumlah segmentasi yang ingin dibentuk. Partisi akan dibentuk dari sekumpulan objek n kedalam cluster k sehingga terbentuk kesamaan objek dalam setiap segmentasi k.

Untuk menghasilkan cluster yang maksimal, titik awal partisi merupakan salah satu faktor yang berpengaruh. Karena *k-means* memecahkan data kedalam segmentasi berdasarkan nilai lokal maksimum. Karena itu pemilihan titik awal harus beralasan, ada beberapa metode yang diusulkan oleh peneliti seperti genetic *k-means*, pemilihan titik awal secara acak sebanyak beberapa kali, metode yang paling baik adalah dengan mengukur nilai titik tengah segmentasi berdasarkan jumlah jarak terpendek antar anggota kelas tersebut.

Beberapa kelemahan dari algoritma *k-means* antara lain:

1. Perlu mengetahui jumlah segmen yang ingin dibentuk, penentuan jumlah segmentasi ini membutuhkan pengalaman dalam melihat dan menilai jumlah segmentasi yang ada dalam data.
2. Mudah terganggu dengan data yang tidak valid, karena cara kerja k-means merata-rata nilai dalam setiap segmen, maka data yang tidak relevan dapat mengacaukan pusat segmen.
3. Hasil partisi dari *k-means* tidak memiliki nilai hirarki, sehingga tidak ada segmentasi yang lebih baik dari segmentasi lainnya.

Meskipun memiliki beberapa kelemahan, algoritma k-means merupakan algoritma yang banyak dipakai dalam menentukan cluster (Wu & Kumar, 2009: p. 32-33). Algoritma k-means popular karena mudah digunakan, memiliki perhitungan yang pasti dan mudah dimodifikasi untuk memenuhi kebutuhan penggunaan.

#### Tahapan algoritma k-means

Algoritma pembuatan partisi untuk algoritma *k-means* adalah :

**Tahap 1**

Membuat partisi sejumlah K sebagai prototype dari segmentasi yang akan terbentuk dengan persamaan

**Tahap 2**

Mengisi setiap obyek dalam dataset kedalam segmen terdekat

(2.4)

Dalam algoritma pengelompokkan, sebuah data dikatakan serupa dengan mengukur jarak nilai dari satu data dengan data lain (Myatt, 2007: p. 125). Proses pengukutan jarak antara dua objek ini dinamakan *Euclidean distance* atau jarak *Euclidean* dengan persamaan sebagai berikut :

(2.5)

Dimana p dan q adalah dua nilai yang sedang diukur. Jarak *Eucledian* hanya mampu mengukur nilai kontinyu, karena itu untuk mengukur jarak data yang biner digunakan jarak Jaccard yang ditulis dalam persamaan :

(2.6)

Dimana adalah jumlah semua variable yang bernilai 1 pada observasi pertama dan 1 pada observasi ke-2, adalah jumlah semua variable yang bernilai 0 pada observasi pertama dan 1 pada observasi ke-2, dan seterusnya.

**Tahap 3**

Kalkulasi ulang setiap segmentasi yang terbentuk

(2.7)

**Tahap 4**

Mengulangi langkah 2 dan 3 hingga data di dalam setiap segmentasi tidak berubah

Untuk menggambarkan penerapak algoritma diatas, diasumsikan k = 3, maka algoritma *k-means* akan memilih titik tengah untuk setiap segmentasi objek, setiap objek akan ditarik kedalam segmentasi sesuai dengan jarak objek tersebut dengan pusat segmen (Gambar 2.10). Pusat segmen kemudian disempurnakan dengan rekalkulasi dari nilai rata-rata dari kumpulan objek yang sudah ada dalam segmen pertama (Gambar 2.11). Secara terus menerus objek dikelompokkan kembali untuk meningkatkan hasil pastisi hingga tidak adalagi objek yang mengalami pengelompokkan kembali.



Gambar 2.11  
Pengambilan titik tengah setiap segmen



Gambar 2.12  
Pusat segmen bergeser, dan objek yang ada dalam segmentasi disesuaikan hingga tidak ada perubahan

#### Studi Kasus Segmentasi dengan k-means

Contoh Partisi dengan *k-means*, data pada Tabel 2.2 akan disegmentasi menjadi tiga kelas. Maka data tersebut secara acak ( dalam contoh ini digunakan D, K, dan M) akan dipilih sebagai data awal dalam membentuk segmentasi seperti pada Gambar 2.9. Setelah itu data lainnya akan dikalkulasi berdasarkan jarak *Euclidean* dari data tersebut terhadap data yang dipilih. Setelah perhitungan ke setiap titik euclidian dilakukan, maka data akan mengikuti segmentasi dengan titik euclidian yang terdekat.

Contoh perhitungan jarak *Euclidean* dari A ke D

Tabel 2.3  
Data yang akan disegmentasi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Variable 1 | Variable 2 | Variable 3 | Variable 4 | Variable 5 |
| A | 7 | 8 | 4 | 5 | 2 |
| B | 6 | 8 | 5 | 4 | 2 |
| C | 8 | 9 | 7 | 8 | 9 |
| D | 6 | 7 | 7 | 7 | 8 |
| E | 1 | 2 | 5 | 3 | 4 |
| F | 3 | 4 | 5 | 3 | 5 |
| G | 7 | 8 | 8 | 6 | 6 |
| H | 8 | 9 | 6 | 5 | 5 |
| I | 2 | 3 | 5 | 6 | 5 |
| J | 1 | 2 | 4 | 4 | 2 |
| K | 3 | 2 | 6 | 5 | 7 |
| L | 2 | 5 | 6 | 8 | 9 |
| M | 3 | 5 | 4 | 6 | 3 |
| N | 3 | 5 | 5 | 6 | 3 |



Gambar 2.13  
Data acak yang akan digunakan sebagai data awal segmen

Perhitungan yang sama dilakukan untuk semua titik kedalam semua segmentasi hingga didapat data yang tersebar seperti Gambar 2.12. Untuk menyempurnakan data dalam segmentasi, dihitunglah nilai titik tengah untuk setiap segmentasi dengan merata-rata variable dalam setiap segmentasi seperti pada Tabel 2.3. setelah itu jarak *Euclidian* diukur kembali dari setiap titik ke nilai tengah segmentasi yang didapat. Data akan dipindah ke segmentasi dengan nilai *Euclidian* yang terdetak. Proses ini diulangi sampai tidak ada lagi data yang berpindah dari satu segmentasi ke segmentasi lainnya.



Gambar 2.14  
Kumpulan data awal untuk setiap segmen

Tabel 2.4  
Nilai titik tengah untuk setiap segmentasi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Segmen 1 |  |  |  |  |  |
| Name | Variable 1 | Variable 2 | Variable 3 | Variable 4 | Variable 5 |
| C | 8 | 9 | 7 | 8 | 9 |
| D | 6 | 7 | 7 | 7 | 8 |
| G | 7 | 8 | 8 | 6 | 6 |
| H | 8 | 9 | 6 | 5 | 5 |
| L | 2 | 5 | 6 | 8 | 9 |
| Center | 6.2 | 7.6 | 6.8 | 6.8 | 7.4 |
| **Segmen 2** |  |  |  |  |  |
| Name | Variable 1 | Variable 2 | Variable 3 | Variable 4 | Variable 5 |
| E | 1 | 2 | 5 | 3 | 4 |
| F | 3 | 4 | 5 | 3 | 5 |
| I | 2 | 3 | 5 | 6 | 5 |
| K | 3 | 2 | 6 | 5 | 7 |
| Center | 2.25 | 2.75 | 5.25 | 4.25 | 5.25 |
| **Segmen 3** |  |  |  |  |  |
| Name | Variable 1 | Variable 2 | Variable 3 | Variable 4 | Variable 5 |
| A | 7 | 8 | 4 | 5 | 2 |
| B | 6 | 8 | 5 | 4 | 2 |
| J | 1 | 2 | 4 | 4 | 2 |
| M | 3 | 5 | 4 | 6 | 3 |
| N | 3 | 5 | 5 | 6 | 3 |
| Center | 4 | 5.6 | 4.4 | 5 | 2.4 |

Tabel 2.4 menggambarkan data setelah pusat segmen baru dihitung, algoritma akan berulang kembali dengan menghitung *euclidean distance* untuk setiap data terhadap pusat segmen yang baru. Namun dalam perhitungan yang dilakukan, tidak ada data yang berpindah kedalam segmen lainnya. Karena terdiri dari data yang sama, pusat segmen tidak akan bergesar dan perhitungan tidak dilanjutkan.