BAB 2
LANDASAN TEORI

2.1 Gabah

Gabah adalah bulir padi. Secara definisi teknis perdagangan, gabah merupakan hasil panen dari tanaman padi yang telah dipisahkan dari tangkainya dengan cara perontokan. Dalam perdagangan komoditas, gabah merupakan tahap yang penting dalam pengolahan padi sebelum dikonsumsi karena perdagangan padi dalam partai besar dilakukan dalam bentuk gabah [12].

2.2 Komponen Mutu Gabah

Mutu gabah adalah sekumpulan sifat-sifat fisik individual (komponen mutu) yang menyusun gabah tersebut. Gabah dengan kualitas tinggi akan menghasilkan beras yang bermutu tinggi pula. Mutu gabah terdiri dari tiga komponen sebagai berikut [13].

1. Kadar Air

   Jumlah kandungan air dalam butir gabah yang dinyatakan dalam persentase dari berat basah.

2. Butiran Hampa

   Butir gabah yang tidak berkembang secara sempurna akibat serangan hama, penyakit, atau sebab lain sehingga tidak berisi butir beras meskipun kedua tungkup sekamnya tertutup ataupun terbuka. Butir gabah setengah hampa tergolong dalam butir hampa.

3. Kotoran

   Segala benda asing yang tidak tergolong bagian dari gabah, misalnya debu, butiran tanah, butiran pasir, batu kerikil, potongan kayu, potongan logam, tangkai padi, biji-bijian lain, bangkai serangga, dan lain sebagainya. Termasuk dalam kategori kotoran adalah butiran gabah yang telah terkelupas (beras pecah kulit) dan gabah patah.

2.3 Kualitas Gabah

Berdasarkan INPRES Nomor 5 Tahun 2015, kualitas gabah dibedakan menjadi dua, yaitu [13] :

1. GKG (Gabah Kering Giling)

   Gabah Kering Giling merupakan gabah yang mengandung kadar air maksimum sebesar 14% dan kadar hampa/kotoran maksimum 3%.

2. GKP (Gabah Kering Panen)

   Gabah Kering Panen merupakan gabah yang mengandung kadar air maksimum sebesar 25% dan kadar hampa/kotoran maksimum 10%.

2.4 Harga Gabah

Macam- macam penentuan harga gabah adalah sebagai berikut [13].

1. Harga di Tingkat Petani

   Harga yang disepakati pada waktu terjadinya transaksi antara petani dengan pedagang pengumpul/tengkulak/pihak penggilingan yang ditemukan pada hari dilaksanakannya observasi dengan kualitas apa adanya, sebelum dikenakan ongkos angkut pasca panen.

2. Harga di Tingkat Penggilingan

   Harga di tingkat petani ditambah dengan besarnya biaya ke penggilingan terdekat. Lokasi terjadinya transaksi gabah, menyebabkan perbedaan cara penghitungan harga di tingkat petani dan penggilingan. Kemungkinan yang terjadi adalah sebagai berikut :

a. Bila transaksi gabah antara petani (produsen) dan pembeli terjadi di sawah/gudang petani, maka harga di tingkat penggilingan adalah harga di tingkat petani ditambah dengan perkiraan besarnya biaya ke penggilingan.
b. Bila transaksi gabah antara petani (produsen) dan pembeli dilakukan oleh pihak penggilingan (terjadi di gudang penggilingan), maka harga gabah di tingkat petani adalah harga di tingkat penggilingan dikurangi besarnya biaya ke penggilingan dari lokasi sebelum adanya ongkos angkut pasca panen siap jual.

3. Harga Pembelian Pemerintah (HPP)

Harga minimal yang harus dibayarkan pihak penggilingan/pembeli kepada petani sesuai dengan kualitas gabah sebagaimana yang telah ditetapkan Pemerintah dalam SK Inpres. Penetapan harga dilakukan secara kolektif antara Departemen Pertanian, Menko Bidang Perekonomian, dan Bulog.

2.5 Prediksi

Prediksi atau peramalan adalah suatu kegiatan memprediksi masa depan menggunakan kondisi ataupun data dimasa lalu. Peramalan merupakan kegiatan dalam memperkirakan apa yang akan terjadi pada masa yang akan datang, atau lebih tepatnya peramalan adalah kegiatan mencoba menduga perubahan yang akan terjadi [14]. Hasil ramalan adalah situasi/kondisi yang diperkirakan akan terjadi pada masa yang akan datang.

Peramalan dibagi menjadi dua bagian yaitu [14]:

1. Peramalan Kualitatif

Metode peramalan kualitatif lebih mendasarkan kualitatif dimasa lalu yaitu berdasarkan pemikiran yang bersifat intuisi, pendapat dan pengetahuan serta pengalaman. Metode ini banyak digunakan dalam banyak pengambilan keputusan sehari-hari. Dalam hal ini ramalan dikatakan baik atau tidak bergantung dari banyak hal antara lain pengalaman,perkiraan dan pengetahuan yang didapat.

2. Peramalan Kuantitatif

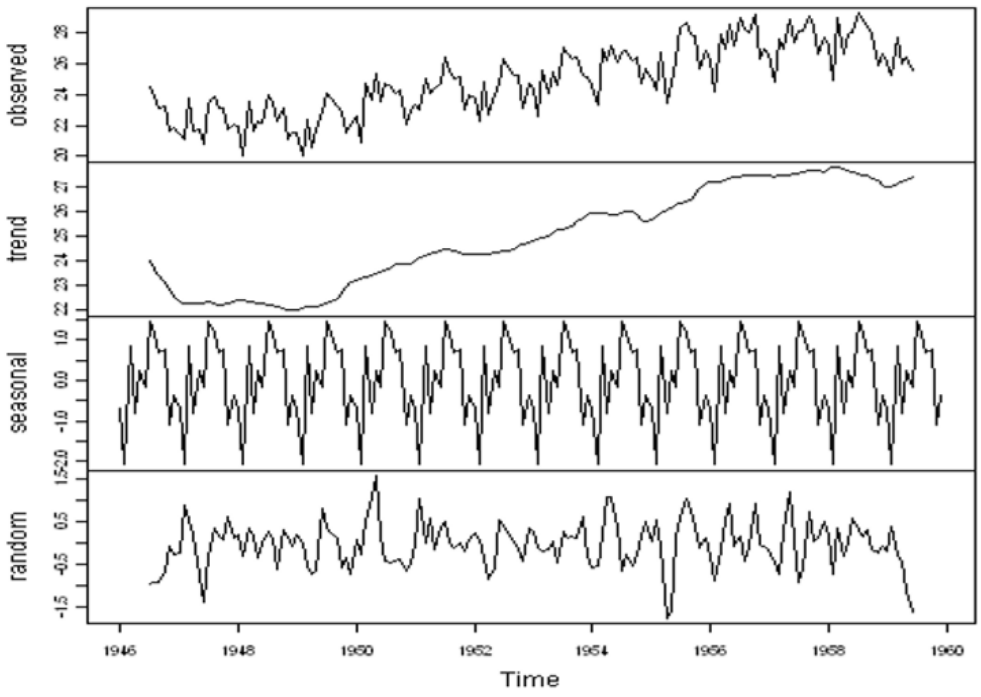
Metode peramalan kuantitatif merupakan peramalan yang didasarkan pada data kuantitatif dimasa lalu. Peramalan dengan menggunakan metode kuantitatif dapat diterapkan apabila terdapat tiga kondisi berikut yaitu [15]:

a. Adanya informasi tentang keadaan yang lain.
b. Informasi tersebut dapat dikuantifikasikan dalam bentuk data numerik.
c. Diasumsikan pola masa lalu akan berkelanjutan pada masa yang akan datang.

2.6 Prediksi Time Series

Prediksi secara *time series* merupakan peramalan yang dilakukan dengan menggunakan data tertentu yang disebut data *time series*. Menurut *Australia Bureau of Statistics*, data *time series* adalah nilai suatu variabel yang disusun berdasarkan urutan/deret waktu bisa dalam satuan harian, mingguan, bulanan, atau tahunan tergantung keperluan terhadap perilaku data yang akan diobservasi. Data *time series* yang digunakan adalah data historis yang diukur berdasarkan suatu pengamatan tertentu. Karena data yang digunakan adalah data yang terukur, maka peramalan secara *time series*, termasuk ke dalam peramalan kuantitatif. Metode prediksi *time series* beranggapan bahwa data atau kejadian masa lalu akan cenderung berulang dimasa yang akan datang. Fokus prediksi secara *time series* adalah apa yang akan terjadi, bukan mengapa hal itu terjadi. Hal yang harus diperhatikan sebelum melakukan prediksi *time series* adalah [16]:

1. Adanya ketergantungan masa depan dengan masa lampau.
2. Aktivitas masa yang akan datang mengikuti pola masa lalu.
3. Hubungan keterkaitan masa lalu dan masa kini dapat ditentukan melalui pengamatan.



Gambar 2. 1 Pola data *time series*

Berikut empat pola data *time series* [16]:

1. Pola Data Konstan

Pola data konstan terjadi apabila data berfluktuasi secara stabil. Pola data konstan berupa garis horizontal. Pola data konstan umumnya terdapat pada data yang memiliki periode waktu jangka pendek atau menengah.

2. Pola Data Trend

Pola data trend terjadi apabila data cenderung meningkat atau menurun dari waktu ke waktu dalan jangka panjang. Pola ini disebabkan antara lain oleh bertambahnya populasi, perubahan pendapat, dan pengaruh budaya.

3. Pola Data Musiman

Pola data musiman terjadi apabila data berfluktuasi berulang-ulang secara teratur dalam setiap periode tentu, misalnya tahunan, semesteran, kuartalan, bulanan atau mingguan. Pola ini berhungan dengan faktor iklim / cuaca atau faktor yang dibuat manusia seperti liburan.

4. Pola Data Siklus

Pola data siklus terjadi apabila data dipengaruhi oleh fluktuasi ekonomi jangka panjang, seperti daur hidup bisnis. Perbedaan utama antara pola data musiman dan siklus adalah pola musiman mempunyai panjang gelombang yang dtetap dan terjadi pada jarak waktu tetap, sedangkan pola siklus memiliki durasi yang lebih panjang dan bervariasi.

5. Pola Data Residual atau Variasi Acak

Pola data residu terjadi apabila fluktuasi data tidak teratur sama sekali. Data yang bersifat residu tidak dapat digambarkan.

2.7 Particle Swarm Optimization

Particle Swarm Optimization (PSO) merupakan salah satu algoritma optimasi metaheuristik dan merupakan bagian dari swarm intelligence. PSO memiliki konsep yang sederhana, efisien, dan cepat dalam mencapai solusi yang konvergen [17]. PSO didasarkan pada perilaku kawanan hewan untuk mencapai solusi yang optimal. Tindakan suatu individu dipengaruhi oleh memori posisi

terbaik yang pernah dilalui dan informasi posisi terbaik dari seluruh individu dalam kawanan untuk mencapai solusi yang optimal.

Dalam sebuah algoritma PSO terdiri dari sekumpulan partikel dimana setiap partikel mewakili sebuah solusi potensial. Jika dianalogikan ke dalam paradigma komputasi evolusioner, sebuah swarm sama dengan sebuah populasi, sedangkan sebuah partikel sama dengan sebuah individu. Sederhananya, partikel-partikel terbang melalui sebuah ruang pencarian multidimensi dimana posisi dari setiap partikel diubah menurut pengalaman mereka sendiri dan tetangga mereka. Jika $x_i(t)$ merupakan posisi partikel $_i$ dalam ruang pencarian di waktu $_t$, maka posisi partikel berubah dengan menambah kecepatan $v_i(t)$ ke dalam posisi saat ini dengan cara

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2.1)$$

dengan $x_i(0), \sim U(x_{min}, x_{max})$ merupakan nilai acak dengan range batas bawah dan batas atas dimensi partikel.

Vektor kecepatan mengarahkan proses optimasi dan mencerminkan pengetahuan yang dialami partikel dan perubahan informasi dari partikel tetangga secara sosial. Pengetahuan yang dialami partikel umumnya disebut sebagai komponen kognitif, yang proporsional dengan jarak partikel dari posisi terbaiknya (disebut sebagai posisi terbaik individu partikel) ditemukan sejak langkah pertama. Perubahan informasi secara sosial disebut juga sebagai komponen sosial dari persamaan kecepatan [17].

2.7.1 Global Best PSO

Untuk *global best* PSO, tetangga setiap partikel adalah seluruh swarm. Jaringan sosial yang digunakan oleh *global best* PSO menggambarkan topologi bintang. Untuk topologi tetangga bintang, komponen sosial dari pembaruan kecepatan partikel menggambarkan informasi yang diperoleh dari seluruh partikel dalam *swarm* [18]. Dalam kasus ini, informasi sosial merupakan posisi terbaik yang ditemukan *swarm* yang dinotasikan dengan $\hat{y}(t)$.

Untuk *global best* PSO, kecepatan partikel $_i$ dihitung menggunakan persamaan

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 j(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_2 j(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (2.2)$$

dimana $v_{ij}(t)$ merupakan kecepatan partikel i dalam dimensi j = 1,..., n_x dalam waktu ke $_t$, $x_{ij}(t)$ merupakan posisi partikel $_i$ dalam dimensi $_j$ dalam waktu ke $_t$, c_1 dan c_2 merupakan konstanta akselerasi yang digunakan untuk memberi skala kontribusi komponen kognitif dan sosial, dan nilai acak dengan *range* [0,1].

Posisi terbaik individu (y_i) yang berhubungan dengan partikel $_i$, merupakan posisi terbaik partikel yang telah dikunjungi sejak waktu pertama. Dengan mempertimbangkan masalah minimalisasi, posisi terbaik individu dalam waktu selanjutnya, t+1, adalah

$$y_i(t+1) = \begin{cases} yi(t) & \text{jika } f(xi(t+1)) \geq f(yi(t)) \\ xi(t+1) & \text{jika } f(xi(t+1)) < f(yi(t)) \end{cases} \quad (2.3)$$

Dimana $f : R^{nx} \rightarrow R$ merupakan fungsi objektif yang dipetakan ke dalam bilangan real. Seperti pada algoritma evolusi, fungsi objektif mengukur seberapa dekat solusi dengan optimum, contohnya fungsi objektif mengukur performansi atau kualitas partikel.

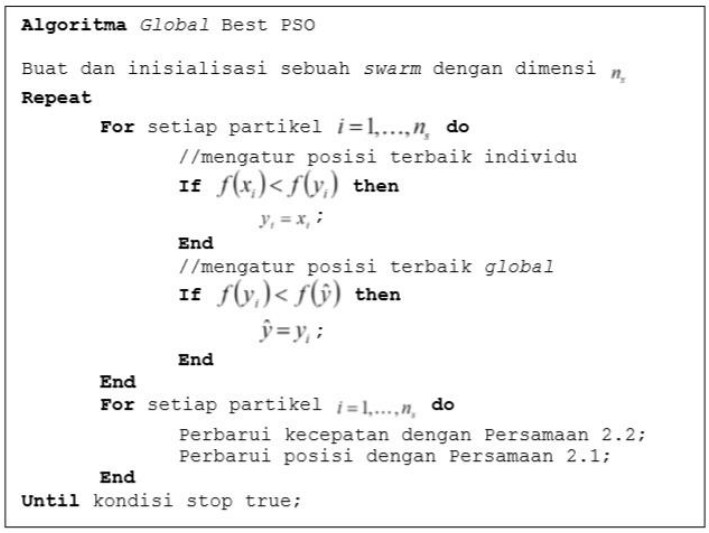
Posisi partikel global, $\hat{y}(t)$ dalam waktu t didefinisikan sebagai berikut

$$\hat{y}(t) \in \{y_o(t),...,y_{ns}(t)\} | f(\hat{y}(t)) = \min\{ f(y_o(t)),...,f(y_{ns}(t))\} \quad (2.4)$$

dimana n_s merupakan jumlah total partikel dalam swarm. Posisi terbaik global juga dapat diperoleh dari partikel swarm saat ini, dimana

$$\hat{y}(t) = \min\{ f(x_o(t)),...,f(x_{ns}(t))\} \quad (2.5)$$

Secara singkat, algoritma global best PSO dijelaskan sebagai berikut [15].



Gambar 2. 2 Algoritma Global Best PSO

2.7.2 Inisialisasi Partikel

Diasumsikan bahwa sebuah partikel untuk setiap dimensinya harus berada dalam domain yang didefinisikan oleh dua vektor, x_{min} dan x_{max} yang mewakili batas bawah dan batas atas di setiap dimensi. Metode inisialisasi yang efisien untuk posisi partikel adalah [18].

$$x(0) = x_{min,j} + r_j(x_{max,j} - x_{min,j}),\ \forall j = 1,...,n_x\ \forall i = 1,...,n_s \qquad (2.6)$$

Dimana rj ~ U(0,1) merupakan nilai acak dengan range [0,1]. Kecepatan awal dapat diinisialisasikan menjadi nol.

$$v_i(0) = 0 \qquad (2.7)$$

Posisi terbaik individu untuk setiap partikel diinisialisasikan dengan posisi partikel partikel pada waktu t = 0.

$$Yi(0) = xi(0) \qquad (2.8)$$

2.7.3 Kondisi Berhenti

Kondisi berhenti merupakan kriteria yang digunakan untuk mengakhiri proses iterasi pencarian. Ketika memilih kriteria untuk mengakhiri iterasi, 2 aspek penting yang harus dipertimbangkan, yaitu

1. Kondisi berhenti harus tidak menyebabkan PSO konvergen dini karena solusi suboptimal akan diperoleh.

2. Kondisi berhenti harus menjaga *oversampling* fungsi objektif. Jika kondisi berhenti membutuhkan penghitungan berkala fungsi objektif, kompleksitas komputasi proses pencarian akan meningkat secara signifikan.

Kondisi berhenti yang dapat digunakan adalah sebagai berikut [18].

a. Ketika jumlah maksimum iterasi telah dilampaui.
b. Ketika sebuah solusi yang memenuhi kriteria telah ditemukan.
c. Ketika tidak ada peningkatan yang dipantau dalam jumlah iterasi.
d. Ketika nilai radius *swarm* yang dinormalisasi yang didapatkan dari nilai maksimum radius *swarm* dibagi dengan diameter *swarm* awal, mendekati nol.
e. Ketika grafik fungsi objektif mendekati nol.

2.7.4 Pembatasan Kecepatan

Pembaruan kecepatan pada Persamaan 2.2 terdiri dari 3 bagian yang berkontribusi ke posisi partikel. Dalam implementasi awal PSO, ditemukan bahwa kecepatan partikel bergerak ke nilai yang besar dengan cepat, terutama untuk posisi partikel yang jauh dari posisi tetangga dan individu terbaik. Konsekuensinya, partikel memiliki posisi terbaru yang besar, yang menyebabkan partikel meninggalkan batas ruang pencarian. Untuk mengontrol eksplorasi global partikel, kecepatan dibatasi untuk tetap dalam batasan yang didefinisikan pada Persamaan 2.9. Jika kecepatan partikel melebihi kecepatan maksimum yang telah ditetapkan, kecepatan partikel diatur ke dalam kecepatan maksimum [18]. Jika $v_{max,j}$ merupakan kecepatan maksimum yang diperbolehkan dalam dimensi j, maka kecepatan partikel disesuaikan sebelum pembaruan posisi menggunakan

$$v_{ij}(t+1) = \begin{cases} v'ij(t+1) & \text{jika } v'ij(t+1) < Vmax,j \\ Vmax,j & \text{jika } v'ij(t+1) \geq Vmax,j \end{cases} \quad (2.9)$$

dimana v’ij dihitung dengan persamaan 2.2

Nilai $V_{max,j}$ sangat penting mengingat nilai tersebut mengontrol granularitas pencarian dengan membatasi kenaikan kecepatan.Nilai $V_{max,j}$ yang besar menyebabkan eksplorasi global, sedangkan nilai lebih kecil menyebabkan eksplorasi lokal. Jika $V_{max,j}$ terlalu kecil, swarm tidak dieksplorasi melampaui

wilayah yang baik secara lokal. Demikian juga, nilai yang terlalu kecil menambah jumlah langkah untuk mencapai optimum dengan tidak ada jalan keluar. Disamping itu, nilai $V_{max,j}$ yang terlalu besar memberikan resiko kemungkinan hilangnya wilayah yang memiliki solusi optimal. Partikel akan melompat ke luar wilayah yang memiliki solusi optimal dan melanjutkan pencarian ke dalam wilayah ruang pencarian yang tidak menghasilkan solusi yang optimal. Partikel bergerak lebih cepat ketika nilai $V_{max,\ j}$ besar karena partikel akan melompat melampaui solusi optimal. Untuk menyelesaikan masalah ini, $V_{max,\ j}$ dipilih sebagai bagian dari domain setiap dimensi ruang pencarian dengan cara

$$V_{max,j} = \delta\ (x_{max,j} - x_{min,j}) \qquad (2.10)$$

Dimana xmax,j dan xmin,j merupakan nilai maksimum dan minimun domain x dalam dimensi j dan $\delta \in [0,1]$ merupakan nilai acak dengan *range* $0 < \delta < 1$.

Ada 2 aspek penting dalam pendekatan pembatasan kecepatan yang harus diperhatikan [18]:

1. Pembatasan kecepatan tidak membatasi posisi partikel, hanya ukuran langkah yang ditentukan dari kecepatan partikel.
2. Dalam Persamaan 2.10, secara eksplisit diimplementasikan kedalam dimensi j. Kecepatan maksimum dihubungkan dengan setiap dimensi, sesuai dengan domain dimensi tersebut. Sebagai pembuktian alasan tersebut, dimisalkan semua dimensi dibatasi oleh $V_{max,\ j}$ yang sama. Kemudian jika dalam sebuah dimensi j terdapat $x_{max,j} - x_{min,j} << V_{max,j}$, partikel akan melebihi nilai optimum dalam dimensi j.

2.7.5 Bobot Inersia

Bobot inersia dikenalkan oleh Shi dan Eberhart (1998) dengan tujuan sebagai sebuah mekanisme untuk mengontrol kemampuan eksplorasi dan ekploitasi *swarm* dan sebuah mekanisme untuk menghilangkan penggunaan batas kecepatan. Bobot inersia berhasil mencapai tujuan pertama tapi tidak berhasil menghilangkan penggunaan batas kecepatan sepenuhnya. Bobot inersia, w mengontrol momentum partikel dengan memberikan bobot kontribusi kecepatan sebelumnya dan pada dasarnya, mengontrol seberapa besar pengetahuan dari arah

terbang sebelumnya akan mempengaruhi kecepatan baru [17]. Untuk *global best* PSO, persamaan pembaruan kecepatan berubah menjadi

$$v_{ij}(t+1) = wv_{ij}(t) + c_1r_1j(t)[y_{ij}(t) - x_{ij}(t)] + c_2r_2j(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (2.11)$$

Nilai w sangat penting untuk memastikan perilaku konvergen dan untuk menukar eksplorasi dan eksploitasi secara optimal. Untuk $w \geq 1$, kecepatan

bertambah melampaui waktu dan menuju kecepatan maksimum (diasumsikan bahwa pembatasan kecepatan digunakan) dan *swarm* berubah. Partikel gagal mengubah arah untuk kembali menuju area yang kemungkinan memiliki solusi. Untuk w < 1, kecepatan berkurang hingga kecepatan mendekati nol (bergantung pada nilai koefisien akselerasi). Nilai w yang besar mengakibatkan eksplorasi dengan peningkatan keberagaman. Nilai w yang kecil mengakibatkan eksploitasi lokal. Bagaimanapun nilai yang terlalu kecil menghilangkan kemampuan eksplorasi *swarm*. Momentum kecil disimpan dari langkah sebelumnya yang mengakibatkan perubahan cepat terhadap arah. Nilai w yang lebih kecil akan lebih melakukan kontrol komponen kognitif dan sosial dalam pembaruan posisi.

Nilai w dapat dicari dengan cara [18]:

1. Penyesuaian acak, dimana bobot inersia dipilih secara acak pada setiap iterasi, contohnya menggunakan distribusi Gaussian

   $$w \sim N(0{,}72,\sigma) \quad (2.12)$$

   dimana σ cukup kecil untuk memastikan w tidak lebih besar dari satu. Alternatif lain, menggunakan

   $$w = (c_1r_1 + c_2r_2) \quad (2.13)$$

2. Penggunaan secara linier, dimana sebuah bobot inersia besar dinisialisasi (biasanya 0,9) dan dikurangi secara linier ke nilai kecil (biasanya 0,4).

   $$w(t) = (w(0) - w(n_t))\frac{(nt-t)}{nt} + w(n_t) \quad (2.14)$$

3. Pengurangan secara tidak linier, dimana sebuah nilai besar yang diinisialisasi dikurangi secara tidak linier ke nilai yang lebih kecil. Metode pengurangan tidak linier memiliki waktu ekplorasi yang lebih kecil

dibandingkan metode pengurangan linier dengan lebih banyak waktu yang digunakan untuk memperbaiki solusi. Metode tersebut didefinisikan sebagai berikut:

$$w(t+1) = \frac{(w(t)-0,4)(nt-t)}{nt + 0,4} \quad (2.15)$$

dengan w(0) = 0,9

4. *Fuzzy* adaptif inersia, dimana bobot inersia disesuaikan berdasarkan himpunan dan aturan fuzzy secara dinamis. Sebuah sistem *fuzzy* didefinisikan oleh Shi dan Eberhart yang terdiri dari:
   a. Dua masukan, yang pertama mewakili *cost* posisi terbaik *global* dan yang kedua mewakili bobot inersia saat ini.
   b. Satu keluaran mewakili perubahan bobot inersia.
   c. Tiga himpunan *fuzzy*, *low, medium, dan high* diimplementasikan sebagai fungsi keanggotaan segitiga.
   d. Sembilan aturan *fuzzy* dari perubahan inersia yang dihitung. Berikut ini aturan *fuzzy* yang dibentuk.

Tabel 2. 1 Aturan *Fuzzy* Bobot Inersia

Cost* Posisi Terbaik *Global	**Bobot Inersia**	**Perubahan Bobot Inersia**
Low	Low	Medium
Low	Medium	Low
Low	High	Low
Medium	Low	High
Medium	Medium	Medium
Medium	High	Low
High	Low	High
High	Medium	Medium
High	High	Low

5. Penambahan inersia, dimana bobot inersia ditambahdari 0,4 ke 0,9 secara linear.

2.7.6 Binary PSO

Sebenarnya, PSO dikembangkan untuk ruang pencarian bernilai kontinu. Kennedy dan Eberhart mengembangkan PSO diskrit pertama untuk beroperasi dalam ruang pencarian biner. *Binary* PSO dapat diaplikasikan ke masalah

optimasi bernilai real karena domain bernilai real dapat ditransformasikan ke dalam domain bernilai biner dengan mudah menggunakan pengkodean standar biner atau pengkodean Gray [18]. Pada *binary* PSO, partikel mewakili posisi pada ruang biner. Setiap elemen pada sebuah vektor posisi partikel diambil dari nilai biner 0 atau 1. Pada dasarnya, perubahan posisi partikel mendefinisikan mutasi bit dengan membalik sebuah bit dalam satu nilai ke yang lain.

Kecepatan baru partikel binary PSO dinormalisasikan menggunakan fungsi sigmoid.

$$v'_{ij}(t) = sig\,(v_{ij}(t)) = \frac{1}{1 + e^{-vij(t)}} \tag{2.16}$$

Dimana $v_{ij}(t)$ didapatkan dari persamaan 2.2, dari persamaan diatas, pembaruan posisi partikel menjadi

$$x_{ij}(t+1) = \begin{cases} 1 & \text{jika } r_{3j}(t) < sig(v_{ij}(t+1)) \\ 0 & \text{lainnya} \end{cases} \tag{2.17}$$

dimana $r_{3j}(t) \sim U(0,1)$ merupakan nilai acak dengan *range* [0,1]. Secara singkat, algoritma *binary* PSO, dijelaskan sebagai berikut [18].

```
Algoritma Binary PSO

Buat dan inisialisasi sebuah swarm dengan dimensi n_x
Repeat
      For setiap partikel i = 1,...,n_s do
            //mengatur posisi terbaik individu
            If f(x_i) < f(y_i) then
                  y_i = x_i ;
            End
            //mengatur posisi terbaik global
            If f(y_i) < f(ŷ) then
                  ŷ = y_i ;
            End
      End
      For setiap partikel i = 1,...,n_s do
            Perbarui kecepatan dengan Persamaan 2.16;
            Perbarui posisi dengan Persamaan 2.17;
      End
Until kondisi stop true;
```

Gambar 2. 3 Algoritma Binary PSO

2.7.7 Fungsi Objektif PSO

Fungsi objektif digunakan untuk mengukur seberapa optimal solusi yang telah dihasilkan. Ada 2 tipe fungsi objektif yaitu *fitness* dan *cost*. Semakin besar nilai *fitness*, maka solusi yang dihasilkan semakin baik. Semakin kecil nilai *cost*, maka solusi yang dihasilkan semakin baik. Pada permasalahan optimasi algoritma yang memiliki tujuan untuk meminimalkan error, digunakan *cost* sebagai fungsi objektifnya. Prediksi akurasi dan jumlah fitur yang dipilih adalah 2 kriteria yang digunakan untuk merancang fungsi objektif sehingga partikel dengan akurasi prediksi tinggi dan jumlah fitur sedikit akan menghasilkan *error* yang kecil. Nilai *mean absolute percentage error* (MAPE) menjadi bagian dari kriteria pada fungsi objektif ini. Partikel dengan nilai *error* prediksi rendah memiliki kemungkinan tinggi untuk memberikan efek ke posisi partikel lain iterasi berikutnya [8]:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}|\text{PE}| \tag{2.18}$$

$$\text{PE} = \left|\frac{\text{Ai-Fi}}{\text{Ai}}\right| \tag{2.19}$$

dimana n merupakan banyak data, Ai merupakan nilai aktual dan Fi merupakan nilai prediksi pada data ke i,

2.8 Normalisasi Data

Metode normalisasi ini menskalakan ulang secara linear data dari satu rentang nilai untuk rentang baru nilai-nilai, seperti [0,1] atau [-1,1] [19]. Metode Min-max menormalkan nilai atribut X sesuai dengan nilai minimum dan maksimum, x_{min} dan x_{max}. Ini dikonversi ke nilai v_i (i = 1,2,3, ..., n) dari atribut X ke v_i' dalam rentang yang telah ditetapkan [new_x_{min}, new_x_{max}] dengan menghitung :

$$v_i' = \frac{\text{Vi - xmin}}{\text{xmax - xmin}} (new_x_{max} - new_x_{min}) + new_x_{min} \tag{2.20}$$

Meskipun Min-max normalisasi memiliki keuntungan dari menjaga hubungan antara nilai-nilai data asli, itu sangat sensitif terhadap nilai ekstrim dan

outlier, sebagai hasilnya mungkin akan didominasi oleh nilai tunggal yang besar. Ini akan menghasilkan "out-of-bound " kesalahan jika kasus masukan masa depan untuk normalisasi jatuh di luar rentang data asli dari atribut X.

2.9 Support Vector Regression

Support Vector Machines (SVM) merupakan algoritma *machine learning* yang diusulkan oleh V. Vapnik di tahun 1998 [20]. SVM digunakan untuk mendeskripsikan masalah klasifikasi. Di samping itu, dengan penggunaan fungsi *insensitive loss*, SVM dikembangkan untuk menyelesaikan masalah regresi yang disebut sebagai *Support Vector Regression* (SVR).

Dimisalkan terdapat sebuah *dataset* latih {xi,yi}, i=1,...,l $x_i \in R^d$; $y_i \in R^d$ dimana xi merupakan vektor variabel masukan dan yi merupakan nilai skalar keluaran (target). Tujuan dari SVR adalah mengembangkan sebuah fungsi f(x) yang dapat memprediksi nilai keluaran berdasarkan variabel masukan pada data uji dengan akurat. Untuk mencapai tujuan tersebut, SVR mencari sebuah fungsi yang dinamakan fungsi ε-SVR yang mayoritas deviasi ε berasal dari nilai keluaran sebenarnya yang diperoleh untuk semua data latih dan sebisa mungkin flat dalam artian fungsi sebisa mungkin mengikuti pola fluktuasi data. Dalam kasus fungsi linier, f(x) dapat berupa

$$f(x) = w \bullet x + b \tag{2.21}$$

dimana $\bullet$ merupakan notasi *dot product*, w merupakan vektor bobot, dan x merupakan *bias*. *Flatness* yang ada dalam kasus Persamaan 2.21 dapat diperoleh dengan meminimalkan *Euclidean* vektor bobot. Masalah ini didefinisikan sebagai sebuah masalah optimasi konveks yang merupakan masalah meminimalkan fungsi konveks dalam himpunan konveks

$$\text{min. } \frac{1}{2}||w||^2$$

$$\text{dengan syarat } \begin{cases} yi - f(xi) \leq \varepsilon \\ f(xi) - yi \leq \varepsilon \end{cases} \tag{2.22}$$

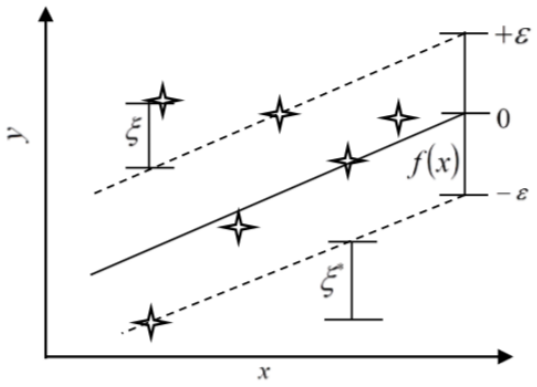
Terkadang, ada kasus dimana sebuah fungsi yang tidak mampu mengakomodasi semua data dengan nilai yang presisi. Oleh karena itu, terdapat variabel slack ξ_i dan ξ_i^* untuk mengatasi kendala masalah optimasi yang tidak mungkin diselesaikan dengan cara diatas. Algoritma SVR tidak hanya meminimalkan error pelatihan dengan meminimalkan jumlah ξ_i dan ξ_i^* , tetapi juga meminimalkan ||w|| untuk meningkatkan *flatness* fungsi. Masalah optimasi ini didefinisikan menjadi

$$\text{min. } \frac{1}{2}\| w^2\| + C \sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

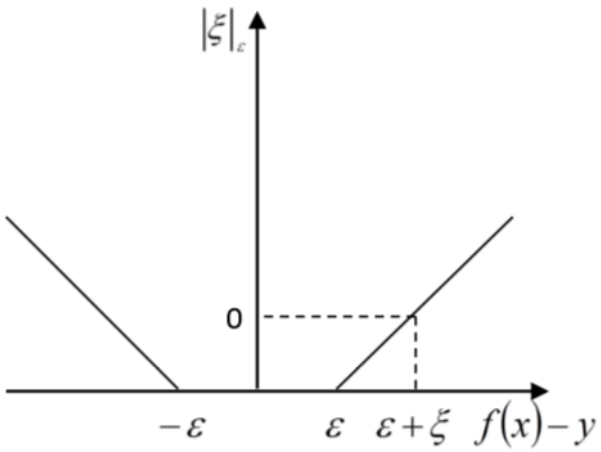
$$\text{dengan syarat } \begin{cases} yi\text{-}f(xi) \leq \varepsilon + \xi_i \\ f(xi)\text{-}yi \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, C > 0 \end{cases} \quad (2.23)$$

dimana C merupakan sebuah konstanta yang menentukan penukaran *flatness* fungsi dengan *error* pelatihan. Hal ini berkaitan dengan penanganan fungsi *ε–insensitive loss.*

$$|\xi|_\varepsilon = \begin{cases} 0, & |f(x)\text{-}y| \leq \varepsilon \\ |f(x)\text{-}y| - \varepsilon, & \text{lainnya} \end{cases} \quad (2.24)$$



Gambar 2. 4 Grafik Pengaturan Loss untuk SVR Linear



Gambar 2. 5 Grafik Fungsi *e- insensitive loss*

Gambar 2.4 menggambarkan pengaturan loss dan Gambar 2.5 menggambarkan fungsi *e- insensitive loss*. Masalah optimasi konveks yang terdapat pada Persamaan 2.23 dapat diselesaikan lebih mudah dalam bentuk *dual*. Berikut ini Persamaan 2.23 yang ditransformasi ke dalam bentuk fungsi objektif *dual*

$$\text{max.} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)(x_i \bullet y_i)+$$

$$\sum_{i=1}^{l} yi(\alpha_i-\alpha_i^*)\text{-}\varepsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*) \qquad (2.25)$$

dengan syarat $\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0, \qquad 0\le\alpha_i,\ \alpha_i^*\le C$

dimana α_i dan α_i^* merupakan Lagrange Multiplier untuk pelatihan ke-i dan diperoleh dengan menyelesaikan Persamaan 2.25. Hanya beberapa koefisien $(\alpha_i-\alpha_i^*)$ bernilai tidak nol dan vektor masukan x_i disebut dengan *support vector*. *Support vector* dan *Lagrange multiplier* ini memberikan nilai pada vektor bobot yag didefinisikan sebagai berikut

$$w=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)xi \qquad (2.26)$$

dengan mengkombinasikan persamaan 2.21 dan 2.26 didapatkan

$$f(x)=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)(x \bullet xi)+b \qquad (2.27)$$

dimana parameter *bias* b didapatkan dari kondisi Karush-Kuhn-Tucker (KKT) berikut

$$\alpha_i(\varepsilon + \xi_i - y_i + (w, x_i) + b) = 0$$

$$\alpha_i^*(\varepsilon + \xi_i^* + y_i - (w, x_i) - b) = 0 \quad (2.28)$$

dan

$$(C - \alpha_i)\,\xi_i = 0$$

$$(C - \alpha_i^*)\,\xi_i^* = 0 \quad (2.29)$$

Nilai b dapat dihitung menggunakan nilai α atau α^* yang ditunjukkan pada persamaan 2.30 yang menggunakan nilai α.

$$b = y_k - \varepsilon - w^T\Phi(X_k)$$
$$= yk - \varepsilon - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)(xi \bullet xk)$$
$$= yk - \varepsilon - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)k(xi \bullet xk) \quad (2.30)$$

dimana data x_k merupakan data latih ke- k. y_k merupakan target data latih ke- k. Salah satu data x_k dan y_k merupakan *support vector*, yang berada pada *boundary* atau di luar *tube*. Persamaan 2.27 merupakan model linier yang dikembangkan menggunakan metode ini. Di sisi lain, terdapat ketidaklinieran antara variabel masukan dan keluaran. Model SVR tidak linier dapat dikembangkan dengan cara memetakan variabel masukan ke dalam ruang fitur berdimensi tinggi F , misalnya $\Phi : R^d \rightarrow F$. Fungsi tidak linear dibentu menjadi

$$f(x) = w \bullet \emptyset(x) + b \quad (2.31)$$

Fungsi tidak linear didapatkan dengan proses yang sama untuk menyelesaikan fungsi linear yaitu

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)(\phi(x) \bullet \phi(xi)) + b \quad (2.32)$$

Di sisi lain, seiring dengan bertambahnya dimensi masukan, dimensi dalam ruang fitur bertambah menjadi berlipat-lipat sehingga proses pemetaan menjadi masalah yang tidak mungkin diselesaikan dengan komputasi. Masalah ini dapat diatasi dengan mendefinisikan fungsi *kernel* yang sesuai ke dalam *dot product* vektor masukan di ruang fitur berdimensi tinggi. Fungsi *kernel* diekspresikan sebagai

$$K(x,x_i) = \phi(x) \bullet \phi(x_i) \quad (2.33)$$

Fungsi regresi didapatkan dengan mengkombinasikan persamaan 2.32 dan 2.33 sehingga menjadi

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x, xi) + b \quad (2.34)$$

Saat ini, beberapa fungsi *kernel* telah didefinisikan dan digunakan seperti fungsi linier, fungsi polinomial, dan *radial basis function* (RBF). Di antara semua fungsi kernel, RBF menjadi metode yang sering digunakan dan didefinisikan sebagai berikut

$$K(x,x_i) = \exp\left(-\frac{||x\text{-}xi||^2}{2\sigma^2}\right) \quad (2.35)$$

dimana $\frac{1}{2\sigma^2}$ merupakan lebar dari kernel RBF. Nilai σ berpengaruh pada kinerja SVR. Jika nilainya terlalu kecil, SVR akan cenderung *overfit* pada data latih. Hal ini menyebabkan tingkat akurasi SVR berkurang karena model regresi yang dibentuk tidak mampu memprediksi data uji secara akurat. Jika nilainya terlalu besar, SVR menjadi tidak fleksibel dalam penghitungan fungsi yang kompleks [21].

2.9.1 Formula Alternatif untuk Bias

Nilai bias dapat diformulasikan ke dalam bentuk lain yaitu dengan menggunakan penambahan dimensi untuk vektor pola seperti $x' = (x_1,..., x_N,\lambda)$ dan berhubungan dengan bias dalam vektor bobot $w' = (w_1,..., w_N,b/\lambda)$ dimana λ

merupakan sebuah konstanta skalar. Fungsi tidak linier yang menggunakan vektor masukan tambahan λ didefinisikan sebagai berikut [22].

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x, xi) + \lambda^2 \quad (2.36)$$

Terjadi perbedaan pendefinisian *Lagrange multiplier* (α_i dan α_i^*) pada fungsi tidak linier yang digunakan Vijayakumar (1999) dan Smola & Scholkopf (2004) yang ditunjukkan pada Persamaan 2.34 dan 2.36. Hal ini terjadi karena bergantung pada inisialisasi letak α_i dan α_i^* pada grafik fungsi *loss* SVR.

2.9.2 Sequential Learning

Pada setiap penghitungan fungsi SVR, terdapat proses sequential learning. Algoritma sequential untuk regresi dapat dilihat di bawah ini [22]:

Nomenklatur yang digunakan pada algoritma ini adalah

$[R]_{ij}$ merupakan matriks *kernel*

E_i merupakan nilai *error*

y_i merupakan nilai kelas *support vector* (berupa nilai aktual data uji)

γ merupakan *learning rate* yang didapatkan dari

$$\frac{\text{konstanta learning rate}}{\max(\text{diagonal matriks kernel})} \quad (2.37)$$

Dimana konstanta *learning rate* akan dibagi dengan nilai maksimum dari diagonal matriks *kernel*.

1. Inisialisasi $\alpha_i = 0$ dan $\alpha_i^* = 0$. Hitung

$$[R]_{ij} = K(x_i,x_j) + \lambda^2 \quad (2.38)$$

Untuk i,j = 1,...,n

2. Untuk setiap data latih (x_i), i = 1 sampai n, Hitung

2.1 $$E_i = y_i - \sum_{j=1}^{n} (\alpha_i^* - \alpha_i) R_{ij} \quad (2.39)$$

2.2 $$\delta\alpha_i^* = \min\{\max[\gamma(E_i - \varepsilon), -\alpha_i^*], C - \alpha_i^*\} \quad (2.40)$$

2.3 $$\delta\alpha_i = \min\{\max[\gamma(E_i - \varepsilon), -\alpha_i], C - \alpha_i\} \quad (2.41)$$

2.4 $\alpha_i^* = \alpha_i^* + \delta\alpha_i^*$ (2.42)

2.5 $\alpha_i = \alpha_i + \delta\alpha_i$ (2.43)

3. Jika proses pelatihan telah mengalami konvergensi, maka iterasi berhenti. Jika tidak, maka kembali ke langkah 2.

Proses pelatihan dikatakan telah mengalami konvergensi apabila jumlah iterasi mencapai jumlah maksimal yang telah didefinisikan atau dengan memantau pembaruan dari tiap *Lagrange multiplier* ($\delta\alpha_i^*$dan $\delta\alpha_i$).

2.9.3 Denormalisasi

Denormalisasi dapat memberikan atau mengembalikan data, sehingga didapatkan *predicted values* [21]. Adapun rumus denormalisasi dalam range [0,1] adalah:

$x_i = y\,(x_{max} - x_{min}) + x_{min}$ (2.44)

Dimana : x_i = Nilai data normal

y = hasil output jaringan

x_{max} = data dengan nilai maksimum

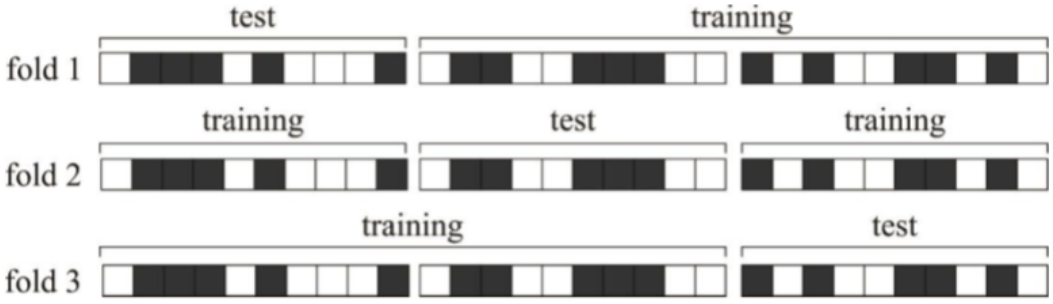
x_{min} = data dengan nilai minimum

2.10 K-Fold Cross Validation

Tujuan utama membangun sebuah model prediksi adalah membangun sebuah model yang dapat memprediksi nilai target dengan akurat. Oleh karena itu, perlu dilakukan estimasi tingkat *error* model prediksi untuk mengukur apakah model prediksi yang telah dibangun dapat memprediksi data uji berdasarkan data latih yang digunakan. Metode yang paling mudah dan banyak digunakan untuk estimasi tingkat *error* prediksi adalah *cross-validation*. Jika tersedia data yang cukup maka sebuah *validation set* dapat diatur dan digunakan untuk mengetahui kinerja model prediksi. Tetapi hal ini tidak mungkin dilakukan, karena data sering kali sulit didapatkan. Untuk mengatasi tersebut, *k-fold cross validation* menggunakan bagian dari data yang tersedia untuk melatih model dan bagian lain untuk mengujinya. Dimisalkan ada m dataset, proses *k-fold cross validation* adalah sebagai berikut [23]:

1. Acak urutan data dalam dataset.
2. Bagi dataset ke dalam k *fold*.
3. Untuk i = 1,..., k :
    a. Latih model menggunakan data yang tidak termasuk pada *fold* i.
    b. Uji model dengan semua data pada *fold* i .

Proses di atas dapat diilustrasikan dengan Gambar 2.6 untuk *3-fold cross validation* dengan data sejumlah 30 buah.



Gambar 2. 6 K-Fold Cross Validation

2.11 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi *console*, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup *compiler*, SDK, Integrated Development Environment (IDE), dan dokumentasi yang umumnya berupa MSDN Library. *Compiler* yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk Microsoft Intermediate Language di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework) [24].

2.12 Basis Data

Basis Data dapat didefinisikan dalam sejumlah sudut pandang seperti himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Basis Data dapat diartikan pula sebagai kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan [25].

Operasi dalam basis data:

1. Pembuatan basis data baru (*create database*).
2. Penghapusan basis data (*drop database*).
3. Pembuatan file/tabel baru ke suatu basis data (*create table*).
4. Penghapusan file/tabel dari suatu basis data (*drop table*).
5. Penambahan/pengisian data baru ke sebuah file/tabel di sebuah basis data (*insert*).
6. Pengambilan data dari sebuah file/tabel (*retrieve/search*).
7. Pengubahan data dari sebuah file/tabel (*update*).
8. Penghapusan data dari sebuah file/tabel (*delete*).

Operasi yang berkenaan dengan pembuatan objek (basis data dan tabel) merupakan operasi awal yang hanya dilakukan sekali dan berlaku seterusnya. Sedang operasi-operasi yang berkaitan dengan isi tabel (data) merupakan operasi rutin yang akan berlangsung berulang-ulang dan karena itu operasi-operasi inilah yang lebih tepat mewakili aktifitas pengelolaan dan pengolahan dalam basis data.

Database Management System (DBMS) atau dalam bahasa Indonesia sering disebut sebagai Sistem Manajemen Basis Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal sebagai berikut:

1. Menyediakan fasilitas untuk mengelola akses data.
2. Mampu menangani integritas data.
3. Mampu menangani akses data yang dilakukan secara bersamaan.
4. Mampu menangani *backup* data.

Database MySQL merupakan sistem manajemen basis data SQL yang sangat terkenal dan bersifat *open source*. MySQL dibangun, didistribusikan, dan didukung oleh MySQL AB. MySQL AB merupakan perusahaan komersial yang dibiayai oleh pengembang MySQL. MySQL mempunyai dua macam lisensi yaitu lisensi yang bersifat *open source* dengan menggunakan GNU (*General Public License*) dan lisensi kedua berupa *Standard Commercial License* yang dapat dibeli dari MySQL AB.

Dalam konteks bahasa SQL, informasi disimpan dalam tabel-tabel yang secara logis merupakan struktur dua dimensi yang tersimpan atas baris-baris data (*row* atau *record*) yang berada dalam satu atau lebih kolom (*column*). Baris pada tabel disebut *insance* dari data sedangkan kolom sering disebut sebagai *atributes* atau *field*. MySQL yang berjalan di sistem operasi *Windows* terbukti sangat stabil. MySQL versi *Windows* mempunyai fitur yang sama dengan MySQL versi *Unix* [25].

2.13 Diagram Konteks

Diagram Konteks adalah sebuah diagram yang mencakup masukan-maskan dasar, system umum dan keluaran. Diagram ini merupakan tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses. Menunujukan system secara keseluruhan, diagram konteks tidak memuat penyimpanan dan penggambaran aliran data yang sederhana. Semua entitas luar yang ditunjukan pada diagram konteks berikut aliran data-aliran data utama yang menuju dan dari system [25].

2.14 Data Flow Diagram (DFD)

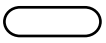
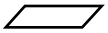
Data Flow Diagram (DFD) adalah representasi grafik dari sebuah system. DFD menggambarkan komponen-komponen sebuah system, aliran-aliran data dari komponen-komponen yang ada mulai dari aliran asal, tujuan, dan penyimpanan dari data tersebut [25].

DFD digunakan untuk dua hal utama, yaitu untuk membuat dokumentasi dari system yang ada atau untuk menyusun dokumentasi untuk system yang baru.

DFD merupakan alat bantu dari pengembangan sebuah bantu dari pengembangan sebuah system yang dibangun secara terstruktur atau procedural.urutan iteraksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

2.15 Flowchart

Sebuah flowchart adalah representasi bergambar dari suatu algoritma. Ini berfungsi sebagai sarana catatan, menganalisis, dan mengkomunikasikan informasi masalah [26]. Programmer sering menggunakan flowchart sebelum menulis sebuah program. Hal ini tidak selalu wajib untuk menggambar flowchart. Dalam prakteknya, kadang-kadang, menggambar dari flowchart dan menulis kode dalam bahasa tingkat tinggi berjalan secara berdampingan.

Dua jenis flowchart yang digunakan, yaitu program flowchart dan sistem flowchart. Sebuah program flowchart (juga disebut diagram alur) menunjukkan langkah-langkah preprocesssing rinci dalam satu program komputer dan urutan di mana langkah-langkah harus dijalankan. Simbol yang berbeda digunakan dalam flowchart untuk menunjukkan operasi yang berbeda yang terjadi dalam sebuah program. Simbol terminal  menunjukkan dengan jelas awal dan akhir program. Simbol  menandakan operasi input / output. Setiap memanipulasi atau preprocessing data dalam komputer dinyatakan dengan simbol  pengolahan. Dalam flowchart simbol keputusan digunakan untuk  menentukan cabang bersyarat atau langkah pengambilan keputusan. Simbol konektor  digunakan dalam flowchart untuk menunjukkan keluar atau masuk dari bagian lain dari flowchart.

Sebuah sistem flowchart menunjukkan prosedur yang terlibat dalam mengkonversi data pada media input data dalam bentuk output. Penekanan ditempatkan pada aliran data masuk atau keluar dari program komputer, bentuk-bentuk masukan dan bentuk-bentuk output. Sebuah sistem flowchart tidak akan mencoba untuk menggambarkan berorientasi fungsi langkah preprocessing dalam sebuah program. Sebuah sistem flowchart dapat dibangun oleh analis sistem

sebagai bagian dari definisi masalah. Namun, algoritma dalam struktur data selalu disajikan dalam bentuk diagram alur.