BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Masalah

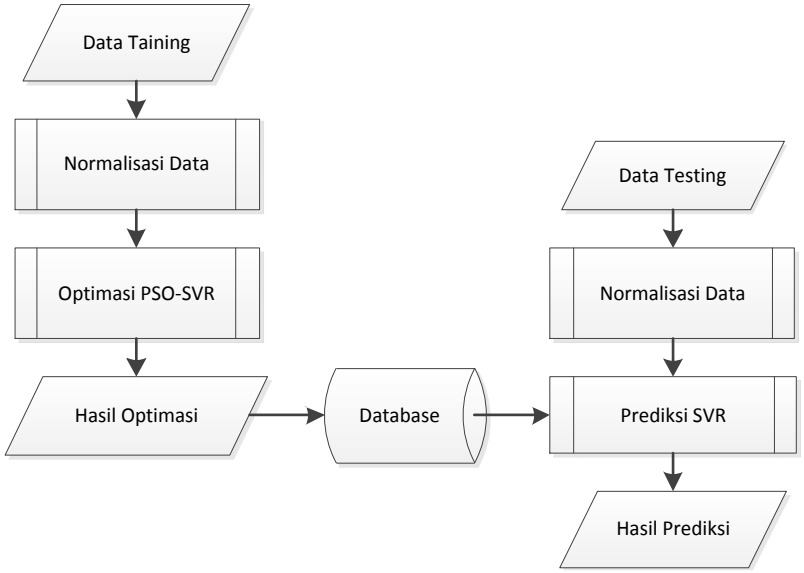
Dalam penggunaan metode Support Vector Regression (SVR), ada dua hal yang dapat mempengaruhi performansinya. Pertama, memilih fitur masukan yang optimal karena ketepatan dalam menentukan jumlah fitur masukan akan menghasilkan estimasi yang bagus dan komputasi yang lebih sederhana. Kedua adalah bagaimana mendapatkan nilai parameter SVR terbaik. Fungsi *kernel* SVR yang umum digunakan adalah *radial basis function* (RBF). Pada SVR-RBF, parameter *error penalty*, *radial basis*, dan *insensitive loss* mempunyai dampak yang signifikan terhadap performansi metode sehingga pencarian nilai parameter yang optimal menjadi masalah penting dalam penggunaan metode tersebut [6]. Dimana penggunan SVR dengan kernel RBF menghasilkan rata-rata tingkat error 5,40. Kedua hal tersebut sangat krusial karena pemilihan fitur akan mempengaruhi nilai parameter SVR begitu juga sebaliknya [7], sehingga proses pemilihan fitur dan parameter yang optimal harus dilakukan secara simultan.

Model SVR dengan optimasi PSO diharapkan dapat mengatasi hal tersebut. Optimasi menggunakan PSO dapat meningkatkan kinerja SVR karena PSO memiliki konsep yang sederhana dan efisien dalam mencapai solusi global sehingga membutuhkan waktu komputasi yang lebih sedikit [17].

3.2 Analisis Sistem

Dalam penelitian ini akan dilakukan optimasi metode *Support Vector Regression* (SVR) dengan algoritma *Particle Swarm Optimization* (PSO) dalam memprediksi harga gabah. Output yang diharapkan adalah mendapatkan nilai prediksi yang akurat dari kombinasi fitur masukan dan nilai parameter terbaik. Langkah yang pertama adalah menentukan parameter awal dan masukan data yang akan digunakan. Langkah selanjutnya adalah melakukan *preprocessing* data menggunakan normalisasi min-max. Kemudian melakukan proses optimasi PSO pada fungsi SVR menggunakan parameter yang telah didefinisikan dan data yang

telah dinormalisasi. Terakhir fitur masukan dan nilai parameter yang optimal dari hasil proses optimasi PSO akan digunakan untuk prediksi menggunakan SVR. Berikut adalah arsitektur dari sistem yang akan dibangun.



Gambar 3. 1 Alur Sistem

Dari gambar diatas, proses prediksi harga gabah dapat dijabarkan sebagai berikut.

1. Data gabah training akan dilakukan proses normalisasi data.
2. Data training normalisasi akan dilakukan optimasi menggunakan metode PSO-SVR.
3. Hasil optimasi yang didapatkan akan disimpan ke database.
4. Data gabah testing juga dinormalisasi terlebih dahulu.
5. Kemudian data testing normalisasi dan hasil optimasi PSO-SVR akan digunakan dalam pengujian.
6. Dalam pengujian dilakukan prediksi harga gabah menggunakan metode SVR.

3.3 Analisis Data Masukan

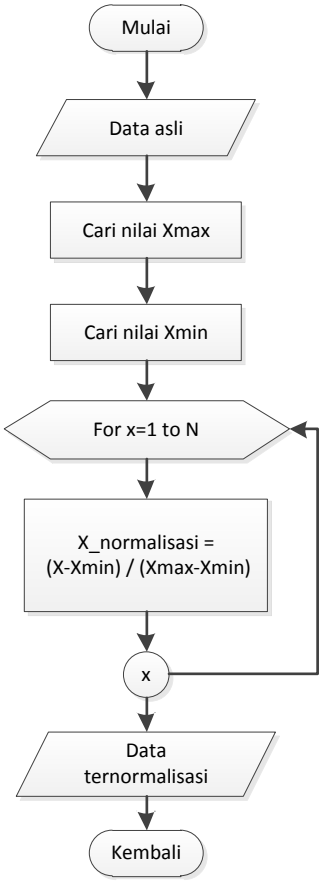
Dataset yang digunakan dalam penelitian ini diambil dari situs BPS Indonesia (https://www.bps.go.id/linkTabelStatis/view/id/1589). Dataset terdiri dari data tahun 2008-2014 dari publikasi tahunan harga produsen gabah dan data

tahun 2015-2016 berasal dari release BRS (Berita Resmi Statistik) setiap bulan. Data tersebut berisi data rata-rata harga gabah menurut kualitas, komponen mutu, dan HPP di tingkat petani. Dalam penelitian ini, atribut yang digunakan adalah data harga gabah kering giling (GKG) setiap bulannya. Data yang digunakan untuk prediksi *time series* adalah data harga gabah pada bulan sebelumnya, yaitu data (t-3) sebagai data 3 bulan sebelumnya, data (t-2) sebagai data 2 bulan sebelumnya dan data (t-1) sebagai data 1 bulan sebelumnya. Dalam perhitungan akan digunakan 25 data sampel. Data tersebut dimulai dari April 2008 sampai dengan April 2010. Berikut adalah contoh sampel data yang akan digunakan.

Tabel 3. 1 Sampel Data

Data	(t-3)	(t-2)	(t-1)	Aktual
1	2822	2770	2610	2507
2	2770	2610	2507	2690
3	2610	2507	2690	2800
4	2507	2690	2800	2863
5	2690	2800	2863	2827
6	2800	2863	2827	2838
7	2863	2827	2838	2897
8	2827	2838	2897	2905
9	2838	2897	2905	2973
10	2897	2905	2973	3036
11	2905	2973	3036	3129
12	2973	3036	3129	3006
13	3036	3129	3006	2638
14	3129	3006	2638	2984
15	3006	2638	2984	3019
16	2638	2984	3019	2947
17	2984	3019	2947	2971
18	3019	2947	2971	3020
19	2947	2971	3020	3117
20	2971	3020	3117	3021
21	3020	3117	3021	3080
22	3117	3021	3080	3476
23	3021	3080	3476	3553
24	3080	3476	3553	3516
25	3476	3553	3516	3186

Sebelumnya data akan dilakukan proses normalisasi agar nilai yang dihitung dapat digunakan dalam perhitungan. Berikut adalah alur dari normalisasi data.



Gambar 3. 2 Alur Normalisasi Data

Adapun proses normalisasi data yang ditunjukkan pada Gambar 3.2 dapat dijabarkan sebagai berikut.

1. Masukkan data asli yang akan dinormalisasikan.
2. Mencari nilai maksimum dan minimum data.
3. Hitung normalisasi data dengan Persamaan 2.20

Dari data sampel didapatkan nilai minimum adalah 2507, sedangkan nilai maksimumnya adalah 3553. Kemudian, melakukan perhitungan normalisasi dengan range [new_x_{max}, new_x_{min}] adalah 0-1. Berikut adalah contoh penghitungannya, misalnya diambil 1 data dari data sampel pada fitur pertama yaitu nomer 1 yang bernilai 2822. Tabel 3.2 merupakan tabel yang berisi hasil normalisasi data.

$$v_1' = \frac{\text{Vi-Xmin}}{\text{Xmax-Xmin}} (1\text{-}0) + 0 = \frac{2822\text{-}2507}{3553\text{-}2507} = 0{,}3010$$

Tabel 3. 2 Hasil Normalisasi Data

No	F1	F2	F3	Aktual
1	0,3010	0,2510	0,0984	0,0000
2	0,2510	0,0984	0,0000	0,1745
3	0,0984	0,0000	0,1745	0,2794
4	0,0000	0,1745	0,2794	0,3403
5	0,1745	0,2794	0,3403	0,3059
6	0,2794	0,3403	0,3059	0,3163
7	0,3403	0,3059	0,3163	0,3727
8	0,3059	0,3163	0,3727	0,3798
9	0,3163	0,3727	0,3798	0,4457
10	0,3727	0,3798	0,4457	0,5051
11	0,3798	0,4457	0,5051	0,5940
12	0,4457	0,5051	0,5940	0,4769
13	0,5051	0,5940	0,4769	0,1250
14	0,5940	0,4769	0,1250	0,4558
15	0,4769	0,1250	0,4558	0,4893
16	0,1250	0,4558	0,4893	0,4204
17	0,4558	0,4893	0,4204	0,4431
18	0,4893	0,4204	0,4431	0,4902
19	0,4204	0,4431	0,4902	0,5834
20	0,4431	0,4902	0,5834	0,4911
21	0,4902	0,5834	0,4911	0,5474
22	0,5834	0,4911	0,5474	0,9264
23	0,4911	0,5474	0,9264	1,0000
24	0,5474	0,9264	1,0000	0,9647
25	0,9264	1,0000	0,9647	0,6494

Pada Tabel 3.2, kolom F1, F2 dan F3 merupakan label yang mewakili tiap-tiap atribut dimana berisi nilai dari tiap tiap atribut dan kolom “Aktual” merupakan nilai harga gabah sebenarnya.

3.4 Analisis Metode

Dalam analisis metode ini terdiri dari proses optimasi PSO-SVR dan proses prediksi dengan SVR. Berikut adalah tahapan-tahapannnya.

3.4.1 Optimasi PSO-SVR

Data yang digunakan dalam proses ini adalah hasil normalisasi data pada Table 3.2. Sebelum melakukan optimasi PSO-SVR, Pertama adalah melakukan inisialisasi nilai parameter PSO. Dalam perhitungan ini menggunakan nilai parameter yang telah mendapatkan hasil yang cukup akurat dalam penelitian tentang kasus *software effort estimation* [27], yaitu sebagai berikut.

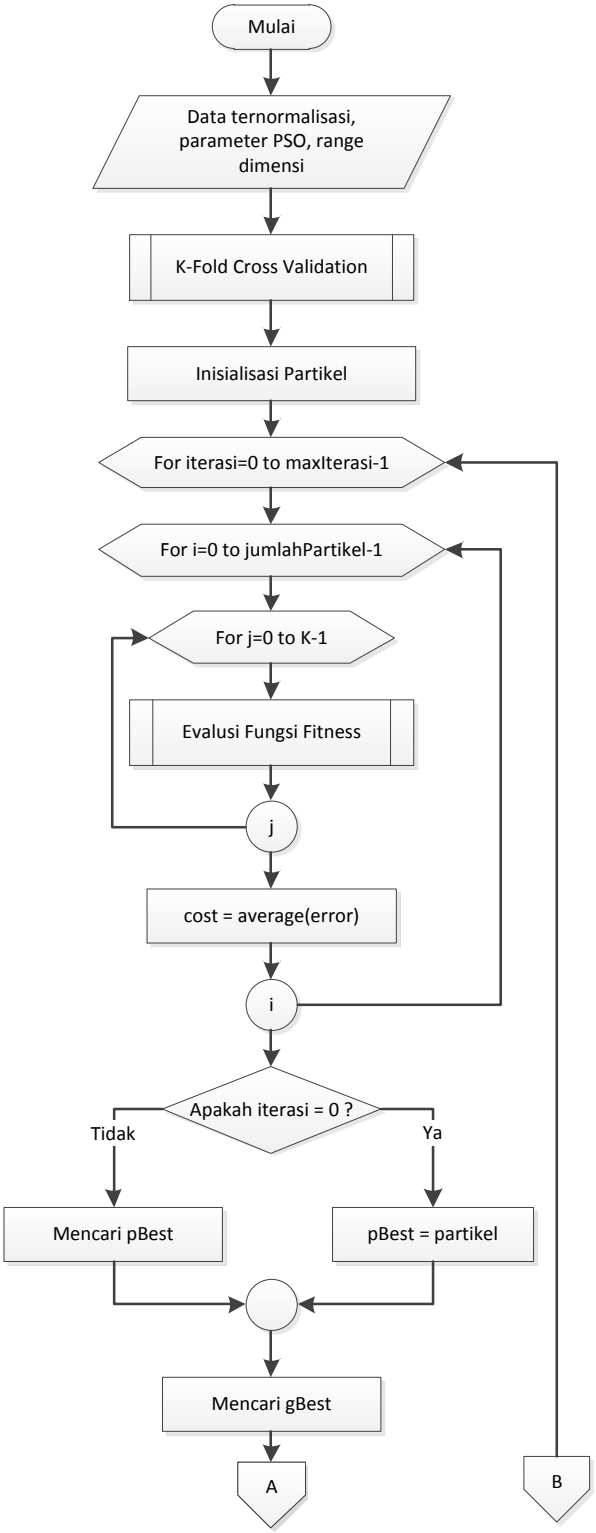
a. Jumlah partikel = 3
b. Jumlah iterasi maksimum = 10
c. Nilai batas bawah bobot inersia (w_{min}) = 0,4
d. Nilai batas atas bobot inersia (w_{max}) = 0,8
e. Nilai konstanta akselerasi (c_1 dan c_2) = 1
f. Jumlah *fold cross validation* (K) =5
g. Nilai batas bawah ruang pencarian tiap dimensi partikel
    i. Dimensi C ($x_{min,1}$) = 0,01
    ii. Dimensi ε ($x_{min,2}$) = 0,01
    iii. Dimensi σ ($x_{min,3}$) = 0,01
    iv. Dimensi *cLR* ($x_{min,4}$) = 0,01
    v. Dimensi λ ($x_{min,5}$) = 0,01
h. Nilai batas atas ruang pencarian tiap dimensi partikel
    i. Dimensi C ($x_{max,1}$) = 1000
    ii. Dimensi ε ($x_{max,2}$) = 0,09
    iii. Dimensi σ ($x_{max,3}$) = 10
    iv. Dimensi *cLR* ($x_{max,4}$) = 2
    v. Dimensi λ ($x_{max,5}$) = 5

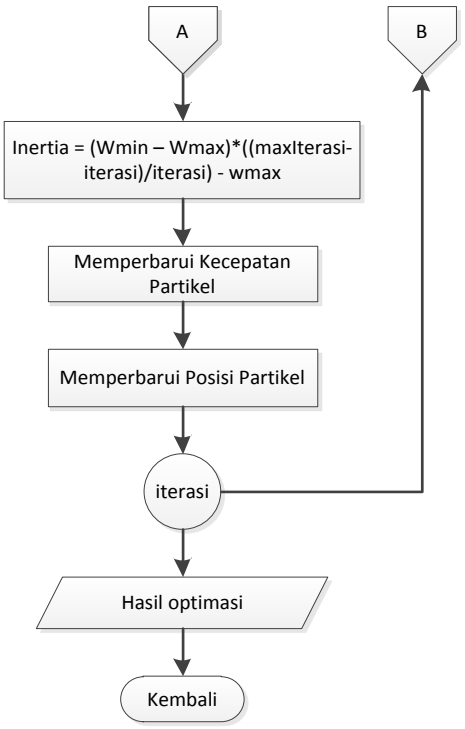
Dalam proses optimasi PSO-SVR ini terdiri dari beberapa tahap antara lain

1. K-Fold Cross Validation
2. Inisialisasi Partikel
3. Evaluasi Fungsi Fitness
4. Mencari Nilai pBest
5. Mencari Nilai gBest

6. Perbaruan Kecepatan Partikel
7. Perbaruan Posisi Partikel

Adapun alur dari proses tersebut dapat dilihat pada gambar dibawah in.





Gambar 3. 3 Alur PSO-SVR

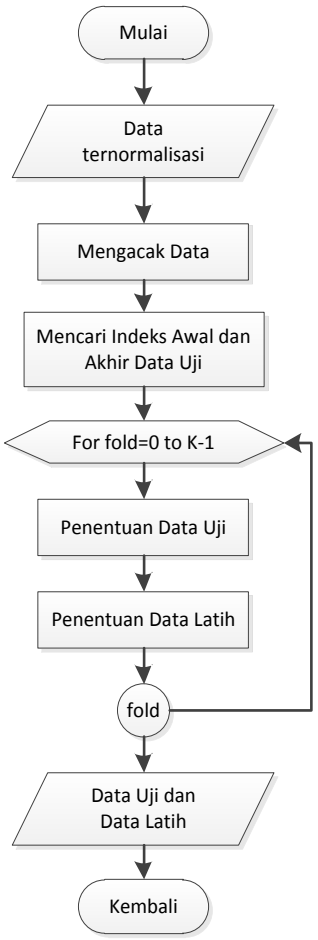
Adapun proses optimasi PSO-SVR yang ditunjukkan pada gambar diatas dapat dijabarkan sebagai berikut:

1. Proses K-*fold cross validation* merupakan proses mempersiapkan data untuk selanjutnya digunakan untuk melatih SVR. Proses ini dilakukan untuk membagi data sebanyak K dan mengulangi proses latihan sebanyak K.
2. Membangkitkan posisi partikel secara acak dengan kode dimensi C, ε , σ , *cLR*, λ dan fitur F1 sebagai (t-3), F2 sebagai (t-2), dan F3 sebgai (t-1) dari Tabel 3.5. Lima dimensi pertama bernilai kontinu. Sisa dimensi lain bernilai diskrit. Pembangkitan partikel bernilai kontinu menggunakan Persamaan 2.6. Kemudian memberikan nilai kecepatan partikel dengan Persamaan 2.7. Lalu menentukan v_{max} tiap dimensi dengan Persamaan 2.9 dan 2.10.
3. Membangkitkan perulangan dengan memberikan nilai iterasi = 0.
4. Membangkitkan perulangan dengan memberikan nilai i = 0.
5. Membangkitkan perulangan dengan memberikan nilai j = 0.
6. Melatih SVR dengan nilai parameter dan jumlah fitur yang diperoleh dari posisi partikel dan menghitung error dengan menggunakan Persamaan 2.19.

7. Jika nilai j = K-1, lanjut ke langkah 8. Jika sebaliknya, memberikan nilai j = j + 1 dan kembali ke langkah 6.
8. Menghitung rata-rata nilai error yang diperoleh dari tiap pelatihan untuk dijadikan nilai cost partikel.
9. Jika nilai i = jumlahPartikel -1, maka lanjut ke langkah 10. Jika sebaliknya, memberikan nilai i = i + 1 dan kembali ke langkah 5.
10. Jika nilai iterasi = 0, dilakukan pendefinisian nilai posisi individu terbaik (pBest) dengan nilai posisi partikel saat ini seperti pada Persamaan 2.7. Jika sebaliknya, dilakukan proses mencari posisi terbaik individu (pBest). Proses ini dilakukan dengan cara membandingkan nilai cost pBest dan partikel iterasi saat ini. Jika nilai cost pBest lebih kecil dari nilai cost partikel iterasi saat ini, nilai posisi pBest tetap. Jika nilai cost pBest lebih besar dari nilai cost partikel iterasi saat ini, nilai posisi pBest digantikan oleh nilai posisi partikel saat ini.
11. Mencari posisi terbaik global (gBest) dengan cara membandingkan nilai cost gBest dan pBest. Jika nilai cost gBest lebih kecil dari nilai cost pBest, nilai posisi gBest tetap. Jika nilai cost gBest lebih besar dari nilai cost pBest, nilai posisi gBest digantikan oleh nilai posisi pBest.
12. Menghitung bobot inersia menggunakan Persamaan 2.14.
13. Memperbarui kecepatan partikel menggunakan Persamaan 2.11 dan 2.16.
14. Memperbarui posisi partikel menggunakan Persamaan 2.1 dan 2.17.
15. Jika nilai iterasi = max Iterasi – 1, maka lanjut ke langkah 17. Jika sebaliknya, memberikan nilai iterasi = iterasi + 1 dan kembali ke langkah 4.
16. Mengambil hasil optimal parameter SVR dan jumlah fitur pilihan dari nilai gBest.

1. K-Fold Cross Validation

Proses *K-fold Cross Validation* dilakukan untuk mengestimasi tingkat kesalahan dengan cara mengelompokkan antara data latih dengan data uji dari keseluruhan data. Berikut adalah alur yang dilakukan dalam tahap ini.



Gambar 3. 4 Alur K-Fold Cross Validation

Adapun langkah-langkahnya adalah mengacak urutan hasil normalisasi data, mencari indeks awal dan akhir data uji, mendefinisikan *dataset* latih dan *dataset* uji sebanyak *K* kali. Hasil dari data acak dapat dilihat pada Tabel 3.7

Tabel 3. 3 Data Acak

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163
10	0,3403	0,3059	0,3163	0,3727

Indeks	F1	F2	F3	Aktual
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059
14	0,4204	0,4431	0,4902	0,5834
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

A. Mencari Indeks Awal dan Akhir Data Uji

Proses mencari indeks awal dan akhir data uji dilakukan untuk mencari nilai awal dan akhir data uji sebanyak *K*. Adapun langkah-langkahnya adalah mencari nilai interval data. Kemudian mencari indeks awal dan akhir data sebanyak *K* kali.

Pada penghitungan manual ini, dilakukan proses mencari indeks awal dan akhir data uji, dan berikut ini contoh penghitungannya. Pertama, hitung interval data.

$$\text{Interval} = \frac{\text{Jumlah Data}}{\text{K}} = \frac{25}{5} = 5$$

Kedua, mencari indeks awal dan akhir data uji berdasarkan Tabel 3.7. Pada Tabel 3.8 adalah hasil pencariannya. Sebagai contoh pada K=1, indeks data uji yang digunakan adalah dari 0-4.

Tabel 3. 4 Hasil Pencarian Indeks Data Uji

	Indeks Awal	Indeks Akhir
K=1	0	4
K=2	5	9
K=3	10	14
K=4	15	19
K=5	20	24

B. Penentuan Data Uji

Proses penentuan data uji dilakukan untuk menentukan data uji untuk setiap *fold*. Penentuan data uji tersebut berdasarkan indeks awal dan akhir yang telah ditentukan dari proses sebelumnya.

Pada penghitungan manual ini, dilakukan proses penentuan data uji, dan berikut ini contoh penghitungannya untuk *fold* ke-1. Pertama, hitung jumlah data uji.

$$\text{jumlah data uji} = (\text{indeks akhir } \textit{fold} \text{ ke-1}) - (\text{indeks awal } \textit{fold} \text{ ke-1}) + 1$$
$$= 4 - 0 + 1$$
$$= 5$$

Kedua, menentukan data uji berdasarkan indeks yang telah ditentukan. Tabel 3.5, 3.6, 3.7, 3.8 dan 3.9 berisi hasil penentuan data uji.

Tabel 3. 5 Data Uji Fold K=1

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457

Tabel 3. 6 Data Uji Fold K=2

Indeks	F1	F2	F3	Aktual
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163

Tabel 3. 7 Data Uji Fold K=3

Indeks	F1	F2	F3	Aktual
10	0,3403	0,3059	0,3163	0,3727
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059
14	0,4204	0,4431	0,4902	0,5834

Tabel 3. 8 Data Uji Fold K=4

Indeks	F1	F2	F3	Aktual
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051

Tabel 3. 9 Data Uji Fold K=5

Indeks	F1	F2	F3	Aktual
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

C. Penentuan Data Latih

Proses penentuan data latih dilakukan untuk menentukan data latih untuk setiap *fold*. Penentuan data latih pada *fold* tersebut berdasarkan ketentuan bahwa indeks data yang tidak termasuk indeks awal atau akhir yang telah ditentukan dari proses sebelumnya.

Pada penghitungan manual ini, dilakukan proses penentuan data latih, dan berikut ini contoh penghitungannya untuk *fold* ke-1. Pertama, hitung jumlah data latih.

Jumlah data latih = jumlah data – ((indeks akhir *fold* ke-1) –
(indeks awal *fold* ke-1) + 1)
= 25 – (4 – 0 + 1)
= 20

Kedua, menentukan data latih berdasarkan ketentuan bahwa indeks data yang tidak termasuk indeks awal atau akhir data uji. Sebagai contoh jika pada K=1 dimana indeks 0-4 adalah data uji, maka indeks 5-24 adalah data latih. Tabel 3.10, 3.11, 3.12, 3.13 dan 3.14 berisi hasil penentuan data latih.

Tabel 3. 10 Data Latih Fold K=1

Indeks	F1	F2	F3	Aktual
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163
10	0,3403	0,3059	0,3163	0,3727
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059
14	0,4204	0,4431	0,4902	0,5834
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

Tabel 3. 11 Data Latih Fold K=2

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457
10	0,3403	0,3059	0,3163	0,3727
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059
14	0,4204	0,4431	0,4902	0,5834
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902

Indeks	F1	F2	F3	Aktual
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

Tabel 3. 12 Data Latih Fold K=3

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

Tabel 3. 13 Data Latih Fold K=4

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163
10	0,3403	0,3059	0,3163	0,3727
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059

Indeks	F1	F2	F3	Aktual
14	0,4204	0,4431	0,4902	0,5834
20	0,3798	0,4457	0,5051	0,5940
21	0,4558	0,4893	0,4204	0,4431
22	0,4893	0,4204	0,4431	0,4902
23	0,5834	0,4911	0,5474	0,9264
24	0,4911	0,5474	0,9264	1,0000

Tabel 3. 14 Data Latih Fold K=5

Indeks	F1	F2	F3	Aktual
0	0,4457	0,5051	0,5940	0,4769
1	0,0984	0,0000	0,1745	0,2794
2	0,3010	0,2510	0,0984	0,0000
3	0,3059	0,3163	0,3727	0,3798
4	0,3163	0,3727	0,3798	0,4457
5	0,0000	0,1745	0,2794	0,3403
6	0,2510	0,0984	0,0000	0,1745
7	0,4769	0,1250	0,4558	0,4893
8	0,1250	0,4558	0,4893	0,4204
9	0,2794	0,3403	0,3059	0,3163
10	0,3403	0,3059	0,3163	0,3727
11	0,4431	0,4902	0,5834	0,4911
12	0,4902	0,5834	0,4911	0,5474
13	0,1745	0,2794	0,3403	0,3059
14	0,4204	0,4431	0,4902	0,5834
15	0,5474	0,9264	1,0000	0,9647
16	0,9264	1,0000	0,9647	0,6494
17	0,5051	0,5940	0,4769	0,1250
18	0,5940	0,4769	0,1250	0,4558
19	0,3727	0,3798	0,4457	0,5051

2. Inisialisasi Partikel

Proses inisialisasi partikel yaitu proses merepresentasikan setiap partikel dengan nilai C, ε , σ , *cLR*, dan λ dengan Persamaan 2.6 berdasarkan batas atas dan batas bawah ruang pencarian tiap dimensi yang telah ditentukan. Kemudian menginisialisasi dimensi fitur dengan nilai 1 dan 0 secara acak dengan *binary PSO*, dimana nilai 1 menyatakan bahwa fitur tersebut terpilih sedangkan 0 menyatakan fitur tidak terpilih. Inisialisasi partikel dilakukan sebanyak *jumlahPartikel*. Setelah itu memberikan nilai kecepatan partikel dengan

Persamaan 2.7 dan kecepatan maksimum partikel dengan Persamaan 2.10. Banyak dimensi kontinu (*dKontinu*) adalah 5 dan dimensi biner (*dBiner*) adalah 3.

Pada penghitungan manual ini, pertama, dilakukan proses inisialisasi partikel berdasarkan nilai parameter yang telah ditentukan sebelumnya diawal proses, dimana jumlah partikel yang digunakan adalah 3. Representasi posisi partikel adalah sebagai berikut.

$$\bar{x} = \begin{pmatrix} C \\ \varepsilon \\ \sigma \\ cLR \\ \lambda \\ F1 \\ F2 \\ F3 \end{pmatrix}$$

Dimana 5 dimensi awal adalah nilai yang terletak dalam batas-batas berikut pada Tabel 3.15. sedangkan untuk 3 dimensi berikutnya diinisialisasikan 0 atau 1.

Tabel 3. 15 Dimensi Parameter

No	Parameter	Xmin	Xmax
1	C	0,01	1000
2	E	0,01	0,09
3	Σ	0,01	10
4	cLR	0,01	2
5	Λ	0,01	5

Berikut ini adalah contoh penghitungan untuk partikel ke-1 dimensi ke-1 menggunakan Persamaan 2.6. Untuk nilai dari r_1 adalah nilai random dari 0-1.

$$\bar{X}_1,c = x_{min,c} + r_1(x_{max,c} - x_{min,c}) = 0,01 + 0,5853(1000 - 0,01) = 585,3681$$

Tabel 3. 16 Hasil Inisialisasi Posisi Partikel

$\bar{x}$	C	ε	σ	cLR	λ	F1	F2	F3
1	585,3681	0,0379	9,7027	0,4523	0,2421	0	1	1
2	801,4480	0,0602	2,6636	1,4660	2,4079	0	1	1
3	99,3700	0,0194	9,6105	0,5259	2,3299	1	0	1

Tabel 3.16 merupakan tabel yang berisi hasil inisialisasi posisi partikel. Nilai dari posisi partikel awal dilakukan secara random, tetapi masih sesuai dengan batas yang ditentukan. Setiap nilai dalam kolom mewakili posisi partikel di setiap dimensi. Lima dimensi pertama partikel mewakili solusi untuk nilai parameter yang optimal sedangkan 3 dimensi selanjutnya mewakili solusi sebagai fitur yang dipilih. Pengecekan jumlah fitur yang dihasilkan bertujuan untuk menjaga agar proses SVR tetap mendapatkan data latih dan uji. Jika jumlah fitur yang dipilih berjumlah 1 atau tidak ada yang dipilih, maka proses pengkodean fitur akan diulang kembali. Kedua, melakukan inisialisasi kecepatan partikel, dimana kecepatan awal partikel masih bernilai 0.

Tabel 3. 17 Inisialisasi Kecepatan Partikel

$\bar{x}$	C	ε	σ	cLR	λ	F1	F2	F3
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0

Tabel 3.17 merupakan tabel yang berisi hasil inisialisasi kecepatan partikel menggunakan Persamaan 2.7. Setiap nilai dalam kolom mewakili kecepatan partikel di setiap dimensi. Ketiga, melakukan inisialisasi v_{max} untuk partikel dengan dimensi bernilai kontinu yaitu lima dimensi pertama menggunakan Persamaan 2.10. Nilai kecepatan maksimum awal partikel digunakan untuk menentukan perubahan langkah dari posisi partikel. Berikut ini adalah contoh penghitungan untuk dimensi ke-1 dan hasil inisialisasi dimisalkan sebagai berikut. Nilai δ merupakan nilai random dari 0-1.

$$v_{max,c} = \delta\ (x_{max,c} - x_{min,c}) = 0{,}4953\ (1000 - 0{,}01) = 495{,}3954$$

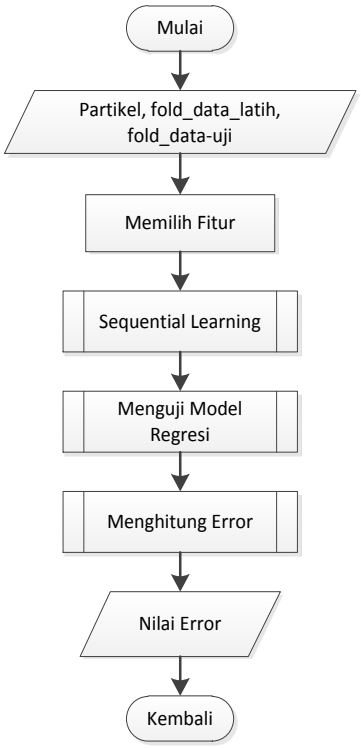
Tabel 3. 18 Hasil Inisialisasi Kecepatan Maksimum Partikel

	C	ε	σ	cLR	λ
V_{max}	495,3954	0,0392	7,3256	0,8147	1,6600

3. Evaluasi Fungsi Fitness

Setelah didapatkan nilai dari 5 dimensi pada tahap sebelumnya, yaitu nilai C, ε, σ, cLR, λ serta fitur yang dipilih, selanjutnya adalah melakukan evaluasi

fungsi *fitness* dengan pelatihan SVR Berikut adalah alur yang dilakukan dalam tahap ini.



Gambar 3. 5 Alur Pelatihan SVR dan Penghitungan Error

Adapun langkahnya adalah memilih fitur yang akan digunakan berdasarkan fitur yang dipilih, melakukan *sequential learning* dan menguji model regresi dengan data uji. Proses ini dilakukan sebanyak K kali.

A. Pemilihan Fitur

Proses memilih fitur yaitu proses menentukan fitur mana yang akan digunakan dalam penghitungan SVR berdasarkan fitur yang dipilih.

Pada penghitungan manual ini, dilakukan proses memilih fitur dan berikut ini contoh penghitungan untuk partikel ke-1 di iterasi ke-0 dimana fitur yang terpilih adalah F2 dan F3 berdasarkan proses sebelumnya.

Tabel 3. 19 Partikel ke-1 Iterasi ke-0

$\bar{x}$	C	E	σ	cLR	λ	F1	F2	F3
1	585,3681	0,0379	9,7027	0,4523	0,2421	0	1	1

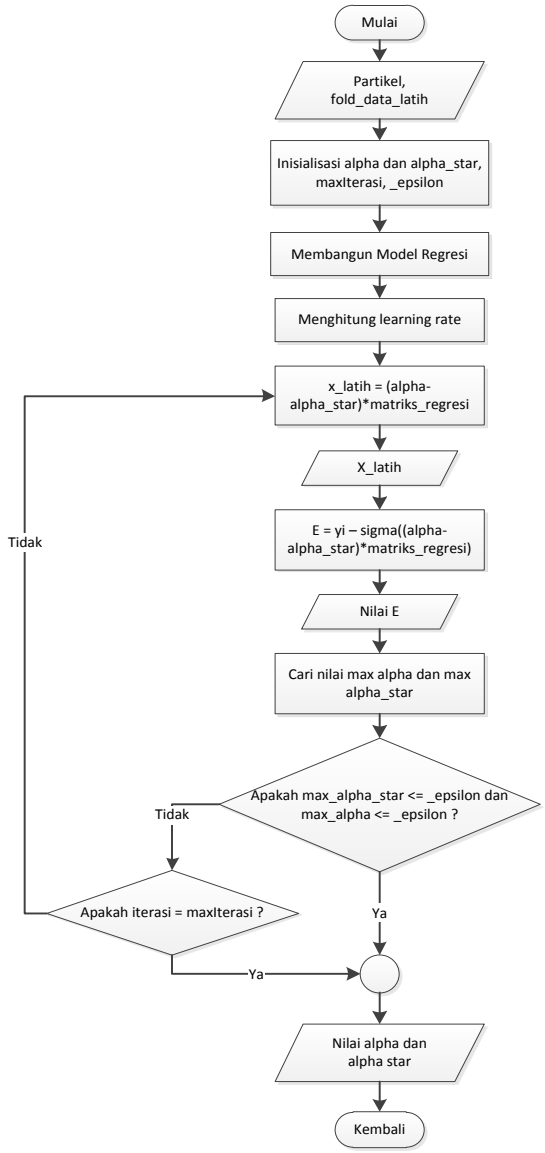
Pada proses K-fold cross validation, data dibagi menjadi 5 fold di mana masing-masing bagian telah ditentukan sebelumnya. Berikut adalah skenario hasil pemilihan fitur dapat dilihat pada Tabel 3.20

Tabel 3. 20 Skenario Data Uji dan Data Latih

Skenario	Data Uji	Data Latih
K=1	Indeks 0-4	Indeks 5-24
K=2	Indeks 5-9	Indeks 0-4 dan 10-24
K=3	Indeks 10-14	Indeks 0-9 dan 15-24
K=4	Indeks 15-19	Indeks 0-14 dan 20-21
K=5	Indeks 20-24	Indeks 0-19

B. Sequential Learning

Proses *sequential learning* yaitu proses mencari nilai *Lagrange multiplier* yang akan digunakan untuk pengujian model regresi. Berikut adalah alur dari proses sequential learning.



Gambar 3. 6 Alur Sequential Learning

Proses ini dimulai dengan menginisialisasi nilai $alpha_i$ = 0 dan vektor $alpha_star_i$ = 0 membangun model regresi, dan menghitung *learning rate*. Langkah selanjutnya adalah untuk setiap data latih, menghitung nilai *error*, menghitung nilai *delta_alpha* dan *delta_alpha_star* dan memperbarui nilai *alpha* dan *alpha_star*. Proses ini berhenti ketika jumlah iterasi mencapai jumlah maksimal yang telah didefinisikan atau nilai *error* lebih kecil dari nilai ε.

Pada penghitungan manual ini, dilakukan proses sequential learning dan berikut ini contoh penghitungan dengan partikel ke-1 di iterasi ke-0 pada dataset latih fold K=1.

1. Inisialisasi $alpha_i = 0$ dan $alpha_star_i = 0$ dengan nilai indeks i=1...n.
2. Membangun model regresi. Proses membangun model regresi yaitu proses membangun matriks yang digunakan untuk melatih data latih hasil proses sebelumnya. Proses ini dimulai dengan membentuk *matriks_x* yang berisi hasil penghitungan jarak antar data latih. Kemudian isi matriks tersebut digunakan untuk membentuk *matriks_kernel* yang berisi model regresi

Pada penghitungan manual ini, dilakukan proses membentuk model regresi dan berikut ini contoh penghitungan dengan partikel ke-1 di iterasi ke-0 pada *fold* K=1 untuk data latih ke-1 dengan data latih ke-2. Pertama, menghitung jarak antar data . Tabel 3.21 berisi hasil penghitungan jarak data latih pada *fold* K=1 dengan partikel ke-1 di iterasi ke-0.

$$||x_1 - x_2||^2 = (0{,}1745 - 0{,}0984)^2 + (0{,}2794 - 0{,}0000)^2$$
$$= 0{,}005791 + 0{,}078064 = 0{,}0839$$

Tabel 3. 21 Jarak Data Fold K=1 Partikel 1Iterasi 0

X	1	2	3	4	5	....	20
1	0,0000	0,0839	0,0336	0,1232	0,0282	**....**	0,5577
2	0,0839	0,0000	0,2085	0,3671	0,1521	**....**	1,0598
3	0,0336	0,2085	0,0000	0,1106	0,0688	**....**	0,3999
4	0,1232	0,3671	0,1106	0,0000	0,0470	**....**	0,1994
5	0,0282	0,1521	0,0688	0,0470	0,0000	**....**	0,4279
6	0,0186	0,1431	0,0522	0,0524	0,0013	**....**	0,4305
7	0,1921	0,4939	0,1497	0,0100	0,0995	**....**	0,1209
8	0,2120	0,4764	0,2114	0,0163	0,0934	**....**	0,1908
9	0,0147	0,1486	0,0372	0,0533	0,0049	**....**	0,4153
10	0,1166	0,3591	0,1024	0,0002	0,0445	**....**	0,2011
11	1,0846	1,6856	0,9384	0,4823	0,8253	**....**	0,1491
12	1,1511	1,7435	1,0246	0,5222	0,8692	**....**	0,2063
13	0,2150	0,4731	0,2204	0,0193	0,0936	**....**	0,2042
14	0,1153	0,1589	0,2333	0,1332	0,0514	**....**	0,6472
15	0,0698	0,2778	0,0650	0,0077	0,0211	**....**	0,2592
16	0,1245	0,3757	0,1053	0,0004	0,0508	**....**	0,1878
17	0,1190	0,3295	0,1340	0,0059	0,0353	**....**	0,2594
18	0,0873	0,3000	0,0874	0,0034	0,0252	**....**	0,2497
19	0,1721	0,4539	0,1424	0,0046	0,0811	**....**	0,1468
20	0,5577	1,0598	0,3999	0,1994	0,4279	**....**	0,0000

Kedua, membentuk model regresi berdasarkan jarak data yang telah diperoleh sebelumnya menggunakan Persamaan 2.38. Berikut ini contoh penghitungan menggunakan data latih ke-1 dan 2. Tabel 3.22 berisi hasil model regresi dengan partikel ke-1 di iterasi ke-0 pada *fold* K=1.

$$[R]_{1,2} = K(x_1,x_2) + \lambda^2 = \exp\left(-\frac{||x1\text{-}x2||^2}{2\sigma^2}\right) + \lambda^2$$

$$= \exp\left(-\frac{0{,}0839}{2*(9{,}7027)^2}\right) + 0{,}2421^2$$

$$= 1{,}0581$$

Tabel 3. 22 Model Regresi Partikel 1 Iterasi 0

X	1	2	3	4	5	....	20
1	1,0586	1,0581	1,0584	1,0579	1,0584	....	1,0556
2	1,0581	1,0586	1,0575	1,0566	1,0578	....	1,0530
3	1,0584	1,0575	1,0586	1,0580	1,0582	....	1,0565
4	1,0579	1,0566	1,0580	1,0586	1,0583	....	1,0575
5	1,0584	1,0578	1,0582	1,0583	1,0586	....	1,0563
6	1,0585	1,0578	1,0583	1,0583	1,0586	....	1,0563
7	1,0576	1,0560	1,0578	1,0585	1,0581	....	1,0579
8	1,0575	1,0561	1,0575	1,0585	1,0581	....	1,0576
9	1,0585	1,0578	1,0584	1,0583	1,0586	....	1,0564
10	1,0580	1,0567	1,0580	1,0586	1,0584	....	1,0575
11	1,0528	1,0497	1,0536	1,0560	1,0542	....	1,0578
12	1,0525	1,0494	1,0532	1,0558	1,0540	....	1,0575
13	1,0574	1,0561	1,0574	1,0585	1,0581	....	1,0575
14	1,0580	1,0577	1,0574	1,0579	1,0583	....	1,0552
15	1,0582	1,0571	1,0582	1,0585	1,0585	....	1,0572
16	1,0579	1,0566	1,0580	1,0586	1,0583	....	1,0576
17	1,0580	1,0568	1,0579	1,0586	1,0584	....	1,0572
18	1,0581	1,0570	1,0581	1,0586	1,0585	....	1,0573
19	1,0577	1,0562	1,0578	1,0586	1,0582	....	1,0578
20	1,0556	1,0530	1,0565	1,0575	1,0563	....	1,0586

3. Menghitung *learning rate*

Proses menghitung *learning rate* dimulai dengan mencari nilai maksimum dari diagonal matriks regresi. Kemudian hasil bagi nilai *cLR* dengan nilai maksimum yang diperoleh sebelumnya menjadi nilai *gamma* yang merupakan variabel yang berisi nilai *learning rate*.

Pada penghitungan manual ini, dilakukan proses menghitung *learning rate* menggunakan Persamaan 2.37 dan berikut ini contoh penghitungan dengan partikel ke- 1 di iterasi ke 0 pada data latih *fold* K=1.

$$\gamma = \frac{\text{konstanta learning rate}}{\max(\text{diagonal matriks regresi})}$$

$$= \frac{0{,}4523}{\max(1{,}0586,\ 1{,}0586,\ 1{,}0586\ 1{,}0586,\ldots,1{,}0586)}$$

$$= 0{,}4273$$

Menghitung nilai *error* menggunakan Persamaan 2.39. Berikut ini contoh penghitungan menggunakan data ke-1. Tabel 3.23 berisi hasil penghitungan nilai *error* dengan partikel ke-1 di iterasi ke-0 pada *fold* K=1.

$$E_1 = y_1 - \sum_{j=1}^{n} (\alpha_1^* - \alpha_1)\, R_{1j}$$

= 0,3403 – (((0-0)*1,0586) + ((0-0)*1,0581) + ((0-0)*1,0584) +
((0-0)*1,0579) + ((0-0)*1,0584) + ... + ((0-0)*1,0556))

= 0,3403

Tabel 3. 23 Nilai Error Partikel 1 Iterasi 0 Fold K=1

Data Ke	**1**	**2**	**3**	**4**	**5**	**....**	**20**
Ei	0,3403	0,1745	0,4893	0,4204	0,3163	....	1,0000

4. Menghitung *delta_alpha_star* dan *delta_alpha* menggunakan Persamaan 2.40 dan 2.41. Berikut ini contoh penghitungan menggunakan data ke-1. Tabel 3.24 berisi hasil penghitungan nilai *delta_alpha_star* dan *delta_alpha* dengan partikel ke-1 di iterasi ke-0 pada *fold K*=1.

$\delta\alpha_1^* = \min\{\max[\gamma(E_1 - \varepsilon), -\alpha_1^*], C - \alpha_1^*\}$

$= \min\{\max[0{,}4273(0{,}3403 - 0{,}0379), -0], 585{,}3681 - 0\}$

$= \min\{0{,}1292, 585{,}3681 - 0\} = 0{,}1292$

$\delta\alpha_1 = \min\{\max[\gamma(-E_1 - \varepsilon), -\alpha_1], C - \alpha_1\}$

$= \min\{\max[0{,}4273(-0{,}3403 - 0{,}0379), -0], 585{,}3681 - 0\}$

$= \min\{0, 585{,}3681 - 0\}$

$= 0$

Tabel 3. 24 Nilai δα* dan δα Partikel 1 Iterasi 0 Fold K=1

Data Ke	1	2	3	4	5	....	20
$\delta\alpha_1^*$	0,1292	0,0584	0,1929	0,1634	0,1189	....	0,4111
$\delta\alpha_1$	0,0000	0,0000	0,0000	0,0000	0,0000	....	0,0000

5. Memperbarui *alpha_star* dan *alpha* menggunakan Persamaan 2.42 dan 2.43. Berikut ini contoh penghitungan menggunakan data ke-1. Tabel 3.25 berisi hasil pembaruan nilai *alpha_star* dan *alpha* dengan partikel ke-1 di iterasi ke-0 pada *fold K*=1.

   $\alpha_1^* = \alpha_1^* + \delta\alpha_1^* = 0 + 0{,}1292 = 0{,}1292$

   $\alpha_1 = \alpha_1 + \delta\alpha_1 = 0 + 0 = 0$

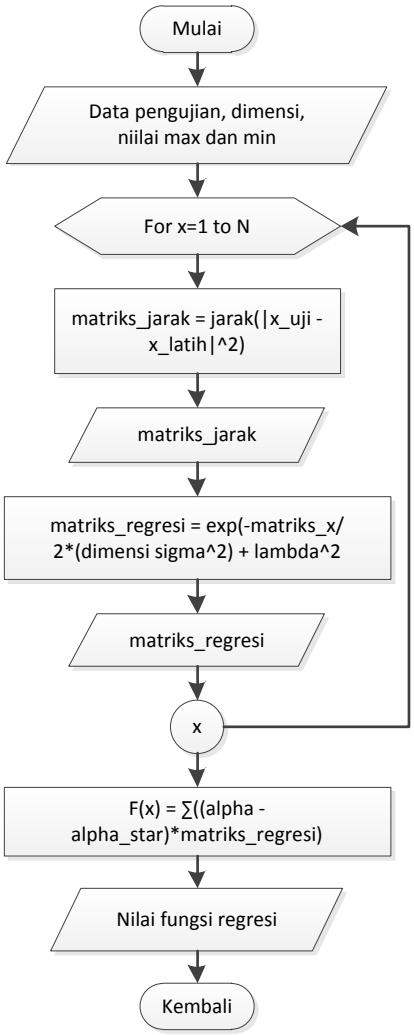
Tabel 3. 25 Nilai α* dan α Partikel 1 Iterasi 0 Fold K=1

Data Ke	1	2	3	4	5	....	20
α_1^*	0,1292	0,0584	0,1929	0,1634	0,1189	....	0,4111
α1	0,0000	0,0000	0,0000	0,0000	0,0000	....	0,0000

6. Mengecek apakah proses telah mengalami konvergensi dengan cara mencari nilai maksimum α^* dan α. Nilai maksimum α^* adalah 0,4111 dan nilai maksimum α adalah 0. Kemudian kedua nilai tersebut dibandingkan dengan nilai *epsilon* yang bernilai 0,000001. Apabila lebih kecil maka proses berhenti, sedangkan bila lebih besar maka akan dilakukan iterasi kembali ke langkah 3 hingga mendapatkan nilai lebih kecil atau maksimal iterasi. Asumsikan bahwa hasil tahap ini telah mencapai kondisi terminasi. Jadi nilai α^* dan α yang terdapat pada Tabel 3.25 akan digunakan dalam menguji model regresi.

C. Menguji Model Regresi

Adapun alur dalam menguji model regresi adalah sebagai berikut.



Gambar 3. 7 Alur Menguji Model Regresi

Proses menguji model regresi yaitu proses memperoleh hasil regresi dari model regresi yang telah dibangun. Proses ini dimulai dengan membentuk matriks regresi antara data latih dan uji. Langkah selanjutnya adalah menghitung fungsi regresi.

Pada penghitungan manual ini, dilakukan proses menguji model regresi dan berikut ini contoh penghitungan dengan partikel ke-1 di iterasi ke-0 pada data latih dan uji *fold K*=1. Pertama, menghitung jarak antara data latih dan data uji. Berikut ini contoh penghitungan menggunakan data uji ke-1 dan data latih ke-1.

Tabel 3.26 berisi hasil penghitungan jarak data uji dan latih pada *fold K*=1 dengan partikel ke-1 di iterasi ke-0.

$$||x_{1(uji)} - x_{1(latih)}||^2 = (0{,}5051 - 0{,}1745)^2 + (0{,}5940 - 0{,}2794)^2$$

$$= 0{,}109296 + 0{,}098973$$

$$= 0{,}1308$$

Tabel 3. 26 Jarak Data Uji dan Latih Fold K=1 Partikel ke-1 Iterasi ke-0

Uji \ Latih	1	2	3	4	5	...	20
1	0,2083	0,5182	0,1636	0,0134	0,1102	...	0,1123
2	0,0415	0,0401	0,0948	0,3069	0,1331	...	0,8650
3	0,0386	0,0330	0,1436	0,1947	0,0510	...	0,7734
4	0,0288	0,1864	0,0435	0,0331	0,0050	...	0,3600
5	0,0494	0,2195	0,0671	0,0189	0,0065	...	0,3293

Kedua, membentuk *matriks_kernel* berdasarkan jarak data yang telah diperoleh sebelumnya menggunakan Persamaan 2.36. Berikut ini contoh penghitungan menggunakan data uji ke-1 dan data latih ke-1. Tabel 3.27 berisi *matriks_kernel* dengan partikel ke-1 di iterasi ke-0 pada *fold K*=1.

$$[R]_{1,2} = K(x_1,x_1) + \lambda^2 = \exp\left(-\frac{||x1(uji)\text{-}x1(latih)||^2}{2\sigma^2}\right) + \lambda^2$$

$$= \exp\left(-\frac{0{,}2083}{2*(9{,}7027)^2}\right) + 0{,}2421^2$$

$$= 1{,}0575$$

Tabel 3. 27 Matriks Kernel Fold K=1 Partikel ke-1 Iterasi ke-0

Uji \ Latih	1	2	3	4	5	...	20
1	1,0575	1,0558	1,0577	1,0585	1,0580	...	1,0580
2	1,0584	1,0584	1,0581	1,0570	1,0579	...	1,0540
3	1,0584	1,0584	1,0578	1,0576	1,0583	...	1,0545
4	1,0584	1,0576	1,0584	1,0584	1,0586	...	1,0567
5	1,0583	1,0574	1,0582	1,0585	1,0586	...	1,0568

Ketiga, menghitung nilai regresi menggunakan persamaan 2.36 dan melakukan denormalisasi dengan persamaan 2.20. Berikut ini contoh penghitungan menggunakan data uji ke-1. Nilai *alpha* dan *alpha_star* diambil

dari hasil proses *sequential learning*. Tabel 3.28 berisi *f(x)* dan Hasil denormalisasi dengan partikel ke-1 di iterasi ke-0 pada *fold* K=1.

$$f(x_1) = \sum_{i=1}^{20} (\alpha_i^* - \alpha_i) (K(x_i,x) + \lambda^2)$$

= (0,1292-0)(1,0575) + (0,0584-0)(1,0558) + (0,1929-0)(1,0577) +
(0,1634-0)(1,585) + (0,1189-0)(1,0580) + ... + (0,4111-0)(1,0580)
= 4,2659

Kemudian melakukan denormalisasi nilai target menggunakan persamaan 2.40, sehingga diperoleh hasil sebagai berikut.

$x_1 = f(x_1)*(x_{max} - x_{min}) + x_{min}$ = 4,2659* (3553– 2507) + 2507 = 6969,1106

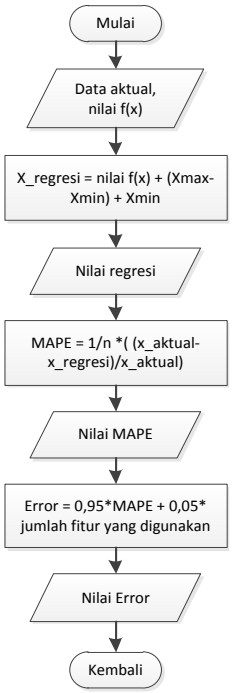
Tabel 3. 28 Nilai f(x) Data Uji Fold K=1 Partikel 1 Iterasi 0

Uji	F(x)	Hasil denormalisasi
1	4,2659	6969,1106
2	4,2569	6959,7666
3	4,2597	6962,6153
4	4,2642	6967,3919
5	4,2647	6967,8748

Proses denormalisasi data dilakukan agar nilai f(x) dapat digunakan untuk menghitung tingkat akurasi dengan data aktual.

D. Penghitungan Error

Proses menghitung *error* yaitu proses menghitung tingkat akurasi hasil regresi. Proses ini dimulai dengan menghitung nilai *mean absolute percentage error* (MAPE). Langkah selanjutnya adalah menghitung *error*. Berikut adalah alur dari proses penghitungan error.



Gambar 3. 8 Alur Penghitungan Error

Pada penghitungan manual ini, dilakukan proses menghitung *cost* dan berikut ini contoh penghitungan dengan *dataset* uji *fold K*=1. Tabel 3.29 berisi nilai aktual dan nilai hasil regresi.

Tabel 3. 29 Nilai Aktual dan Nilai Hasil Regresi

Data	Aktual (Ai)	Regresi (fi)
1	3006	6969,1106
2	2800	6959,7666
3	2507	6962,6153
4	2905	6967,3919
5	2973	6967,8748

Pertama, menghitung nilai MAPE dengan persamaan 2.18

$$\text{MAPE} = \frac{1}{5}\sum_{i=1}^{5}\left|\frac{\text{Ai-fi}}{\text{Ai}}\right|$$

$$= \frac{1}{5}\text{ X }\left[\left|\frac{3006-6969{,}1106}{3006}\right| + \left|\frac{2800-6959{,}7666}{2800}\right| + \cdots + \left|\frac{2863-6967{,}8748}{2863}\right|\right]$$

$$= \frac{1}{5}\text{ X } 7{,}3239 = 1{,}4648$$

Kedua, menghitung nilai *error*. Nilai error yang digunakan menggunakan cara yang sama pada kasus *software effort estimation* [27], yaitu

$$\text{Error} = (W_A \text{ x MAPE}) + (W_F \text{ x } \frac{\sum_{j=1}^{3} Fj}{3})$$

$$= (0{,}95 \text{ x } 1{,}4648) + (0{,}05 \text{ x } \frac{2}{3}) = 1{,}4249$$

Dimana fj merupakan nilai fitur yang tepilih. Tahap diatas diulang sebanyak K kali dan Nilai cost merupakan nilai rata-rata error dari semua *fold* yang digunakan. Tabel 3.30 berisi hasil nilai *error* dan *cost* untuk partikel ke-1 iterasi ke-0 yang diperoleh dari rata-rata error.

Tabel 3. 30 Nilai Error dan Cost Partikel ke-1 Iterasi ke-0

Fold	Error
1	1,4249
2	1,3721
3	1,1989
4	1,0970
5	0,8875
Cost	**1,1961**

4. Mencari pBest

Tahap mencari *pBest* yaitu tahap mencari posisi individu terbaik. Tahap ini dimulai dengan cara membandingkan nilai *cost pBest* dan partikel iterasi saat ini. Jika nilai *cost pBest* lebih kecil dari nilai *cost* partikel iterasi saat ini, nilai posisi *pBest* tetap. Jika nilai *cost pBest* lebih besar dari nilai *cost* partikel iterasi saat ini, nilai posisi *pBest* digantikan oleh nilai posisi partikel saat ini.

Pada penghitungan manual ini, dilakukan proses mencari *pBest* menggunakan Persamaan 2.3. Berdasarkan Persamaan 2.8, *pBest* pada iterasi ke-0 diinisialisasi dengan posisi partikel pada iterasi ke-0 sehingga seperti yang ada pada Tabel 3.31.

Tabel 3. 31 Posisi Individu Terbaik Iterasi ke-0

pBest	C	ε	σ	cLR	λ	F1	F2	F3	Cost
1	585,3681	0,0379	9,7027	0,4523	0,2421	0	1	1	1,1961
2	801,4480	0,0602	2,6636	1,4660	2,4079	0	1	1	6,5210
3	99,3700	0,0194	9,6105	0,5259	2,3299	1	0	1	2,4342

Nilai parameter di atas merupakan nilai parameter pada proses inisialisasi partikel. Nilai cost yang didapatkan merupakan hasil dari evaluasi fungsi *fitness* dengan menggunakan pelatihan SVR sebelumnya.

5. Mencari gBest

Tahap mencari *gBest* yaitu tahap mencari posisi terbaik *global*. Tahap ini dimulai dengan cara membandingkan nilai *cost gBest* dan *pBest*. Jika nilai *cost gBest* lebih kecil dari nilai *cost pBest*, nilai posisi *gBest* tetap. Jika nilai *cost gBest* lebih besar dari nilai *cost pBest*, nilai posisi *gBest* digantikan oleh nilai posisi *pBest*.

Pada penghitungan manual ini, dilakukan proses mencari *gBest* menggunakan Persamaan 2.4 dan Tabel 3.32 berisi *gBest* pada iterasi ke-0.

Tabel 3. 32 Posisi Terbaik Global Iterasi ke-0

gBest	C	ε	σ	cLR	λ	F1	F2	F3	Cost
1	585,3681	0,0379	9,7027	0,4523	0,2421	0	1	1	1,1961

Setelah mendapatkan *gBest*, melakukan langkah selanjutnya yaitu menghitung bobot inersia menggunakan nilai bobot yang telah ditentukan pada awal proses. Berikut ini contoh penghitungan dengan bobot inersia pada iterasi ke-0 menggunakan Persamaan 2.14. Perhitungan tersebut menggunakan cara yang sama pada kasus *software effort estimation*.

$$w(0) = (w(0) - w(n_t))\frac{(nt-0)}{nt} + w(n_2) = (0,8 - 0,4)x\frac{(2-0)}{2} + 0,4 = 0,8$$

6. Perbarui Kecepatan Partikel

Tahap memperbarui kecepatan partikel yaitu tahap menghitung kecepatan partikel berdasarkan *pBest*, *gBest*, dan bobot inersia yang diperoleh dari tahap sebelumnya. Tahap ini dimulai dengan cara menghitung kecepatan partikel menggunakan Persamaan 2.11. Setelah itu, untuk partikel pada dimensi bernilai kontinu, mengecek apakah kecepatan partikel tersebut melewati v_{max} yang telah didefinisikan sedangkan untuk partikel pada dimensi bernilai diskrit dilakukan normalisasi.

Pada penghitungan manual ini, dilakukan tahap memperbarui kecepatan partikel dan berikut ini contoh penghitungan pada iterasi ke-0. Pertama, membangkitkan nilai acak r_1 dan r_2 pada range [0,1] yaitu 0,6482 dan 0,2373.

Kedua, perbarui kecepatan partikel menggunakan Persamaan 2.11 dan berikut ini contoh penghitungan pada partikel ke-1 dimensi ke-1 di iterasi ke-0. Diketahui c_1 dan c_2 bernilai 1.

$v_{11}(1) = wv_{11}(0) + c_1r_1(0)[y_{11}(0) – x_{11}(0)] + c_2r_2(0))[y_{11}(0) – x_{11}(0)]$

$= 0,8x0 + 1x0,6482x[585,3681 - 585,3681] + 1x0,2373x$

$[585,3681 - 585,3681]$

$= 0$

Tabel 3. 33 Kecepatan Baru Partikel di Iterasi ke-0

x	*C*	*ε*	*σ*	*cLR*	*λ*	F1	F2	F3
1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
2	-191,3387	-0,0197	6,2331	-0,8976	-1,9178	0,0000	0,0000	0,0000
3	430,3513	0,0164	0,0817	-0,0651	-1,8488	-0,8855	0,8855	0,0000

Ketiga, melakukan pengecekan kecepatan partikel untuk dimensi partikel bernilai kontinu yaitu lima dimensi pertama, dengan cara membandingkan kecepatan baru partikel dengan v_{max} yang telah didefinisikan sebelumnya pada proses inisialisasi partikel menggunakan persamaaan 2.9. Jika kecepatan baru partikel lebih besar dari v_{max} maka nilai kecepatan baru partikel menjadi v_{max}. Tabel 3.34 berisi hasil pembaruan kecepatan untuk partikel dimensi bernilai kontinu.

Tabel 3. 34 Kecepatan Baru Partikel Dimensi Bernilai Kontinu di Iterasi ke-0

X	*C*	*ε*	*σ*	*cLR*	*λ*
1	0,0000	0,0000	0,0000	0,0000	0,0000
2	-191,3387	-0,0197	6,2331	-0,8976	-1,9178
3	430,3513	0,0164	0,0817	-0,0651	-1,8488

Keempat, melakukan normalisasi kecepatan partikel untuk dimensi partikel bernilai diskrit yaitu pada tiga dimensi terakhir menggunakan Persamaan 2.16 dan berikut ini contoh penghitungan pada partikel ke-1 dimensi ke-6 di iterasi ke-0. Tabel 3.35 berisi kecepatan baru partikel dimensi ke-6 sampai 8 di iterasi ke-0.

$$v_{16}(1) = \frac{1}{1 + e^{-v11(0)}} = \frac{1}{1 + e^{-0}} = 0{,}5$$

Tabel 3. 35 Kecepatan Baru Partikel Dimensi Diskrit Hasil Normalisasi

x	F1	F2	F3
1	0,5000	0,5000	0,5000
2	0,5000	0,5000	0,5000
3	0,7080	0,2920	0,5000

7. Perbarui Posisi Partikel

Tahap memperbarui posisi partikel yaitu tahap menghitung posisi partikel berdasarkan kecepatan partikel yang diperoleh dari tahap sebelumnya. Untuk partikel pada dimensi bernilai kontinu, pembaruan posisi menggunakan Persamaan 2.1. Untuk partikel pada dimensi bernilai diskrit, menggunakan Persamaan 2.17.

Pada penghitungan manual ini, dilakukan proses memperbarui posisi partikel dan berikut ini contoh penghitungan pada iterasi ke-0. Pertama, untuk posisi partikel yang berada pada dimensi bernilai kontinu, pembaruan posisi dilakukan menggunakan Persamaan 2.1 dan berikut ini contoh penghitungan pada partikel ke-1 dimensi ke-1 di iterasi ke-0.

$x_1(1) = x_1(0) + v_1(1) = 585{,}3681 + 0 = 585{,}3681$

Kedua, mengecek apakah posisi baru partikel berada dalam ruang pencarian. Proses ini dilakukan dengan cara membandingkan nilai posisi baru partikel dengan nilai batas bawah dan batas atas dimensi partikel. Jika nilai posisi baru partikel lebih kecil dari nilai batas bawah dimensi partikel maka posisi baru partikel sama dengan nilai batas bawah dimensi partikel. Jika nilai posisi baru partikel lebih besar dari nilai batas atas dimensi partikel maka posisi baru partikel sama dengan nilai batas atas dimensi partikel. Tabel 3.36 berisi posisi baru partikel dimensi ke-1 sampai 5 di iterasi ke-0.

Tabel 3. 36 Posisi Baru Partikel Dimensi ke-1 sampai 5 di Iterasi ke-0

X	C	ε	σ	cLR	λ
1	585,3681	0,0379	9,7027	0,4523	0,2421
2	610,1093	0,0405	8,8967	0,5684	0,4900
3	529,7213	0,0358	9,6921	0,4608	0,4811

Ketiga, membangkitkan nilai acak pada range [0,1] sebanyak dimensi diskrit partikel seperti yang ada pada Tabel 3.37.

Tabel 3. 37 Nilai Random

x	F1	F2	F3
1	0,7188	0,2710	0,0988
2	0,5262	0,4607	0,5287
3	0,4528	0,8205	0,0691

Keempat, untuk posisi partikel yang berada pada dimensi bernilai diskrit, pembaruan posisi dilakukan menggunakan Persamaan 2.7 yaitu membandingkan kecepatan baru partikel dengan nilai *random*. Jika nilai *random* lebih kecil dari nilai kecepatan baru partikel maka posisi baru partikel bernilai 1. Jika nilai *random* lebih besar atau sama dengan nilai kecepatan baru partikel maka posisi baru partikel bernilai 0. Mekanisme pengecekan jumlah fitur yang dihasilkan bertujuan untuk menjaga agar proses SVR tetap mendapatkan data latih dan uji. Jika jumlah fitur yang dipilih berjumlah 1 atau tidak ada yang dipilih, maka proses pembaruan posisi akan diulang kembali. Tabel 3.38 berisi posisi baru partikel dimensi ke-6 sampai 8 di iterasi ke-0.

Tabel 3. 38 Posisi Baru Partikel Dimensi ke-6 sampai 8 di Iterasi ke-0

X	F1	F2	F3
1	0	1	1
2	0	1	0
3	1	0	1

Seluruh tahap di atas diulang sebanyak *maxIterasi* dan Tabel 3.39 berisi nilai *gBest* setiap iterasi dan *gBest* pada *maxIterasi* merupakan solusi dari permasalahan.

Tabel 3. 39 Posisi Terbaik Global Sampai MaxIterasi

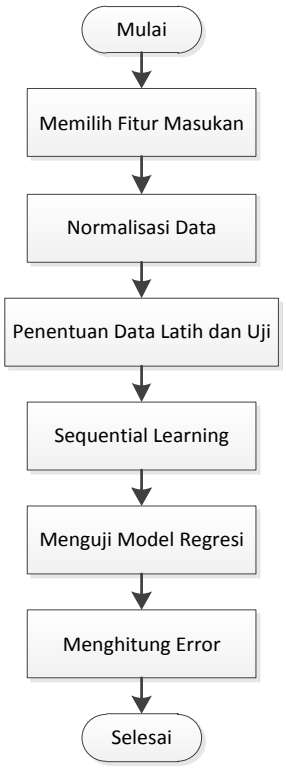
gBest	C	ε	σ	cLR	λ	F1	F2	F3	Cost
0	585,3681	0,0379	9,7027	0,4523	0,2421	0	1	1	1,9712
1	491,4546	0,0283	10	0,0118	0,0100	1	0	1	1,1844
2	0,01	0,01	10	0,01	0,01	1	0	1	0,6511
3	514,2876	0,0185	10	0,01	0,01	1	1	0	0,4915
4	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508

gBest	*C*	ε	σ	*cLR*	λ	F1	F2	F3	Cost
5	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508
6	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508
7	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508
8	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508
9	201,6664	0,01	10	0,01	0,01	0	1	1	0,3508
10	**201,6664**	**0,01**	**10**	**0,01**	**0,01**	**0**	**1**	**1**	**0,3508**

Dari siklus penyelesaian di atas, diketahui bahwa kombinasi parameter SVR yang dihasilkan sedikit mengalami perubahan. Hal ini disebabkan jumlah *maxIterasi* yang sedikit sehingga menyebabkan konvergensi dini. Nilai *error* yang dihasilkan menggunakan optimasi PSO-SVR saat iterasi terakhir adalah 0,3508. Ini menunjukkan bahwa model regresi yang terbentuk sudah cukup akurat dalam memprediksi harga gabah.

3.4.2 Prediksi SVR

Dalam tahap ini fitur masukan dan nilai parameter optimal yang telah didapatkan pada tahap sebelumnya akan digunakan untuk memprediksi harga gabah menggunakan SVR Berikut alur dalam proses prediksi harga gabah. Berikut adalah alur proses prediksi SVR



Gambar 3. 9 Alur Prediksi SVR

1. Memilih Fitur Masukan

Pada tahap sebelumnya telah didapatkan bahwa fitur F2 dan fitur F3 merupakan fitur masukan yang optimal berdasarkan data testing normalisasi pada Tabel 3.6. Berikut adalah data yang digunakan dalam pengujian.

Tabel 3. 40 Data Pengujian

Indeks	F2	F3	Aktual
0	0,5051	0,5940	0,4769
1	0,0000	0,1745	0,2794
2	0,2510	0,0984	0,0000
3	0,3163	0,3727	0,3798
4	0,3727	0,3798	0,4457
5	0,1745	0,2794	0,3403
6	0,0984	0,0000	0,1745
7	0,1250	0,4558	0,4893
8	0,4558	0,4893	0,4204
9	0,3403	0,3059	0,3163
10	0,3059	0,3163	0,3727
11	0,4902	0,5834	0,4911
12	0,5834	0,4911	0,5474
13	0,2794	0,3403	0,3059
14	0,4431	0,4902	0,5834
15	0,9264	1,0000	0,9647
16	1,0000	0,9647	0,6494
17	0,5940	0,4769	0,1250
18	0,4769	0,1250	0,4558
19	0,3798	0,4457	0,5051
20	0,4457	0,5051	0,5940
21	0,4893	0,4204	0,4431
22	0,4204	0,4431	0,4902
23	0,4911	0,5474	0,9264
24	0,5474	0,9264	1,0000

2. Penentuan Data Latih dan Uji

Data gabah yang digunakan dalam proses ini menggunakan perbandingan 80% data latih dan 20% data uji dari 25 data yang digunakan sebagai sampel berdasarkan data acak yang telah ternormalisasi. Berikut adalah hasil penentuanya.

Tabel 3. 41 Data Latih Pengujian

Indeks	F2	F3	Aktual
0	0,5051	0,5940	0,4769
1	0,0000	0,1745	0,2794
2	0,2510	0,0984	0,0000
3	0,3163	0,3727	0,3798
4	0,3727	0,3798	0,4457
5	0,1745	0,2794	0,3403
6	0,0984	0,0000	0,1745
7	0,1250	0,4558	0,4893
8	0,4558	0,4893	0,4204
9	0,3403	0,3059	0,3163
10	0,3059	0,3163	0,3727
11	0,4902	0,5834	0,4911
12	0,5834	0,4911	0,5474
13	0,2794	0,3403	0,3059
14	0,4431	0,4902	0,5834
15	0,9264	1,0000	0,9647
16	1,0000	0,9647	0,6494
17	0,5940	0,4769	0,1250
18	0,4769	0,1250	0,4558
19	0,3798	0,4457	0,5051

Tabel 3. 42 Data Uji Pengujian

Indeks	F2	F3	Aktual
20	0,4457	0,5051	0,5940
21	0,4893	0,4204	0,4431
22	0,4204	0,4431	0,4902
23	0,4911	0,5474	0,9264
24	0,5474	0,9264	1,0000

3. Sequential Learning

Nilai parameter optimal yang telah didapatkan dalam proses optimasi PSO-SVR akan digunakan dalam tahap ini. Berikut adalah nilai parameter optimal yang didapatkan

Tabel 3. 43 Nilai Parameter Optimal

C	ε	σ	cLR	λ
201,6664	0,01	10	0,01	0,01

Kemudian membangun model regresi menggunakan parameter tersebut. Pertama, menghitung jarak antar data. Berikut perhitungan menggunakan data latih ke-1 dan data latih ke-2.

$$||x_1 - x_2||^2 = (0{,}5051 - 0{,}0000)^2 + (0{,}5940 - 0{,}1745)^2$$

$$= 0{,}255126 + 0{,}175980$$

$$= 0{,}4311$$

Tabel 3.48 Jarak Data Latih Pengujian

X	1	2	3	4	5	...	20
1	0,0000	0,4311	0,3102	0,0846	0,0634	...	0,0377
2	0,4311	0,0000	0,0688	0,1393	0,1811	...	0,2178
3	0,3102	0,0688	0,0000	0,0795	0,0940	...	0,1372
4	0,0846	0,1393	0,0795	0,0000	0,0032	...	0,0094
5	0,0634	0,1811	0,0940	0,0032	0,0000	...	0,0044
6	0,2083	0,0415	0,0386	0,0288	0,0494	...	0,0698
7	0,5182	0,0401	0,0330	0,1864	0,2195	...	0,2778
8	0,1636	0,0948	0,1436	0,0435	0,0671	...	0,0650
9	0,0134	0,3069	0,1947	0,0331	0,0189	...	0,0077
10	0,1102	0,1331	0,0510	0,0050	0,0065	...	0,0211
11	0,1168	0,1137	0,0505	0,0033	0,0085	...	0,0222
12	0,0003	0,4075	0,2924	0,0746	0,0553	...	0,0311
13	0,0167	0,4406	0,2647	0,0854	0,0568	...	0,0435
14	0,1153	0,1056	0,0593	0,0024	0,0103	...	0,0212
15	0,0146	0,2960	0,1904	0,0299	0,0171	...	0,0060
16	0,3423	1,5397	1,2690	0,7657	0,6912	...	0,6060
17	0,3823	1,6244	1,3115	0,8179	0,7356	...	0,6540
18	0,0216	0,4443	0,2609	0,0880	0,0584	...	0,0469
19	0,2208	0,2299	0,0517	0,0871	0,0758	...	0,1123
20	0,0377	0,2178	0,1372	0,0094	0,0044	...	0,0000

Langkah kedua, membentuk model regresi berdasarkan jarak data yang telah diperoleh sebelumnya menggunakan persamaan 2.38. berikut ini penghitungan menggunakan data latih ke-1 dan 2.

$$[R]_{1,2} = K(x_1,x_2) + \lambda^2 = \exp\left(-\frac{||x1\text{-}x2||^2}{2\sigma^2}\right) + \lambda^2 = \exp\left(-\frac{0{,}4311}{2*(10)^2}\right) + 0{,}01^2$$

$$= 0{,}9979$$

Tabel 3. 44 Model Regresi Pengujian

X	1	2	3	4	5	...	20
1	1,0001	0,9979	0,9986	0,9997	0,9998	...	0,9999
2	0,9979	1,0001	0,9998	0,9994	0,9992	...	0,9990
3	0,9986	0,9998	1,0001	0,9997	0,9996	...	0,9994
4	0,9997	0,9994	0,9997	1,0001	1,0001	...	1,0001
5	0,9998	0,9992	0,9996	1,0001	1,0001	...	1,0001
6	0,9991	0,9999	0,9999	1,0000	0,9999	...	0,9998
7	0,9975	0,9999	0,9999	0,9992	0,9990	...	0,9987
8	0,9993	0,9996	0,9994	0,9999	0,9998	...	0,9998
9	1,0000	0,9986	0,9991	0,9999	1,0000	...	1,0001
10	0,9995	0,9994	0,9998	1,0001	1,0001	...	1,0000
11	0,9995	0,9995	0,9998	1,0001	1,0001	...	1,0000
12	1,0001	0,9981	0,9986	0,9997	0,9998	...	0,9999
13	1,0000	0,9979	0,9988	0,9997	0,9998	...	0,9999
14	0,9995	0,9996	0,9998	1,0001	1,0000	...	1,0000
15	1,0000	0,9986	0,9991	1,0000	1,0000	...	1,0001
16	0,9984	0,9924	0,9938	0,9963	0,9966	...	0,9971
17	0,9982	0,9920	0,9936	0,9960	0,9964	...	0,9968
18	1,0000	0,9979	0,9988	0,9997	0,9998	...	0,9999
19	0,9990	0,9990	0,9998	0,9997	0,9997	...	0,9995
20	0,9999	0,9990	0,9994	1,0001	1,0001	...	1,0001

Langkah berikutnya menghitung *learning rate*. Berikut penghitungan *learning rate* menggunakan persamaan 2.35.

$$\gamma = \frac{\text{konstanta learning rate}}{\max(\text{diagonal matriks regresi})}$$

$$= \frac{0{,}01}{\max(1{,}0001, 1{,}0001, 1{,}0001, 1{,}0001, 1{,}0001, \ldots, 1{,}0001)}$$

$$= 0{,}01$$

Selanjutnya, menghitung nilai *error* menggunakan Persamaan 2.39. Berikut ini contoh penghitungan menggunakan data ke-1.

$$E_1 = y_1 - \sum_{j=1}^{n} (\alpha_1^* - \alpha_1) R_{1j}$$

$$= 0{,}0000 - (((0\text{-}0)*1{,}0001) + ((0\text{-}0)*0{,}9979) + ((0\text{-}0)*0{,}9986) + ((0\text{-}0)*0{,}9997) + ((0\text{-}0)*0{,}9998) + \ldots + ((0\text{-}0)*0{,}9999))$$

$$= 0{,}0000$$

Tabel 3. 45 Nilai Error Pengujian

Data Ke	1	2	3	4	5	...	20
E_i	0,4769	0,2794	0,0000	0,3798	0,4457	...	0,5051

Lalu langkah berikutnya menghitung *delta_alpha_star* dan *delta_alpha* menggunakan Persamaan 2.40 dan 2.41. Berikut ini contoh penghitungan menggunakan data ke-1.

$\delta\alpha_1^* = \min\{\max[\gamma(E_1 - \varepsilon), -\alpha_1^*], C - \alpha_1^*\}$

$= \min\{\max[0{,}01(0{,}0047 - 0{,}01), -0], 201{,}6664 - 0\}$

$= \min\{0{,}0047, 201{,}6664 - 0\}$

$= 0{,}0047$

$\delta\alpha_1 = \min\{\max[\gamma(-E_1 - \varepsilon), -\alpha_1], C - \alpha_1\}$

$= \min\{\max[0{,}01(-0{,}0000 - 0{,}01), -0], 201{,}6664 - 0\}$

$= \min\{0, 201{,}6664 - 0\}$

$= 0$

Tabel 3. 46 Nilai *delta alpha_star* dan *delta alpha* Pengujian

Data Ke	1	2	3	4	5	...	20
$\delta\alpha_1^*$	0,0047	0,0027	0,0000	0,0037	0,0044	...	0,0050
$\delta\alpha 1$	0,0000	0,0000	0,0000	0,0000	0,0000	...	0,0000

Selanjutnya, memperbarui *alpha_star* dan *alpha* menggunakan Persamaan 2.42 dan 2.43. Berikut ini contoh penghitungan menggunakan data ke-1.

$\alpha_1^* = \alpha_1^* + \delta\alpha_1^* = 0 + 0{,}0047 = 0{,}0047$

$\alpha_1 = \alpha_1 + \delta\alpha_1 = 0 + 0 = 0$

Tabel 3. 47 Nilai *alpha_star* dan *alpha* Pengujian

Data Ke	1	2	3	4	5	...	20
α_1^*	0,0047	0,0027	0,0000	0,0037	0,0044	...	0,0050
α_1	0,0000	0,0000	0,0000	0,0000	0,0000	...	0,0000

Langkah terakhir, mengecek apakah proses telah mengalami konvergensi dengan cara mencari nilai maksimum α^* dan α. Nilai maksimum α^* adalah 0,0054 dan nilai maksimum α adalah 0. Kemudian kedua nilai tersebut dibandingkan dengan nilai *epsilon* yang bernilai 0,000001. Apabila lebih kecil maka proses

berhenti, sedangkan bila lebih besar maka akan dilakukan iterasi kembali ke langkah 3 hingga mendapatkan nilai lebih kecil atau maksimal iterasi. Asumsikan bahwa nilai tersebut sudah memenuhi kondisi terminasi. Jadi nilai α^* dan α yang terdapat pada Tabel 3.47 akan digunakan dalam menguji model regresi.

4. Menguji Model Regresi

Berikut ini contoh perhitungan menggunakan data uji ke-1 dan data latih ke-1. Pertama , menghitung jarak antara data latih dan dan data uji.

$$||x_{1(uji)} - x_{1(latih)}||^2 = (0{,}4457 - 0{,}5051)^2 + (0{,}5051 - 0{,}5940)^2$$
$$= 0{,}003528 + 0{,}007903 = 0{,}0114$$

Tabel 3. 48 Jarak Data Uji dan Latih Pengujian

Uji \ Latih	1	2	3	4	5	...	20
1	0,0114	0,3079	0,2033	0,0343	0,0210	...	0,0079
2	0,0304	0,2999	0,1605	0,0322	0,0152	...	0,0126
3	0,0299	0,2489	0,1475	0,0158	0,0063	...	0,0017
4	0,0024	0,3802	0,2592	0,0611	0,0421	...	0,0227
5	0,1123	0,8650	0,7734	0,3600	0,3293	...	0,2592

Kedua, membentuk matrik kernel berdasarkan jarak data yang telah diperoleh sebelumnya menggunakan persamaan 2.38. berikut ini contoh penghitungan menggunakan data uji ke-1 dan data latih ke-1.

$$[R]_{1,2} = K(x_1,x_1) + \lambda^2 = \exp\left(-\frac{||x1(uji)\text{-}x1(latih)||^2}{2\sigma^2}\right) + \lambda^2$$

$$= \exp\left(-\frac{0{,}0114}{2*(10)^2}\right) + 0{,}01^2$$

$$= 1{,}0000$$

Tabel 3. 49 Matriks Kernel Pengujian

Uji \ Latih	1	2	3	4	5	...	20
1	1,0000	0,9986	0,9991	0,9999	1,0000	...	1,0001
2	0,9999	0,9986	0,9993	0,9999	1,0000	...	1,0000
3	1,0000	0,9989	0,9994	1,0000	1,0001	...	1,0001
4	1,0001	0,9982	0,9988	0,9998	0,9999	...	1,0000
5	0,9995	0,9958	0,9962	0,9983	0,9985	...	0,9988

Ketiga, menghitung nilai regresi menggunakan persamaan 2.36 dan melakukan denormalisasi dengan persamaan 2.20. Nilai *alpha* dan *alpha_star* diambil dari hasil proses *sequential learning*. Berikut ini contoh penghitungan menggunakan data uji ke-1 dan data latih ke-1.

$$f(x_1) = \sum_{i=1}^{20} (\alpha_i^* - \alpha_i) (K(x_i,x) + \lambda^2)$$

= (0,0000-0)(1,0000) + (0,0006-0)(0,9986) + (0,0012-0)(0,9991) +
(0,0023-0)(0,9999) + (0,0021-0)(1,0000) + ...+ (0,0054-0)(1,0001)
= 0,0813

Kemudian melakukan denormalisasi nilai target menggunakan persamaan 2.40, sehingga diperoleh hasil sebagai berikut.

$x_1 = f(x_1)*(x_{max} - x_{min}) + x_{min}$ = 0,0813* (3553– 2507) + 2507 = 2592,0102

Tabel 3. 50 Nilai f(x) Data Pengujian

Uji	F(x)	F(x) denormalisasi
1	0,0813	2592,0102
2	0,0813	2592,0071
3	0,0813	2592,0071
4	0,0813	2592,0103
5	0,0812	2591,9367

Proses denormalisasi data dilakukan agar nilai f(x) dapat digunakan untuk menghitung tingkat akurasi dengan data aktual.

5. Menghitung Error

Berikut adalah perbedaan nilai aktual dengan nilai hasil prediksi harga gabah. Kemudian akan dihitung tingkat errornya. Berikut adalah tahapan-tahapannya.

Tabel 3. 51 Nilai Aktual dan Nilai Hasil Prediksi

Data	Aktual (Ai)	PSO-SVR (fi)
1	3129	2592,0102
2	2971	2592,0071
3	3020	2592,0071
4	3476	2592,0103
5	3553	2591,9367

Pertama, menghitung nilai MAPE dengan persamaan 2.18

$$\text{MAPE} = \frac{1}{5}\sum_{i=1}^{5}\left|\frac{\text{Ai-fi}}{\text{Ai}}\right|$$

$$= \frac{1}{5}\text{ X }\left[\left|\frac{3129-2592{,}0102}{3129}\right| + \left|\frac{2971-2592{,}0071}{2971}\right| + \cdots + \left|\frac{3553-2591{,}9367}{3553}\right|\right]$$

$$= \frac{1}{5}\text{ x } 0{,}9657 = 0{,}1931$$

Kedua, menghitung nilai *error* dengan menggunakan cara yang sama seperti pada kasus *software effort estimation* [27], yaitu

$$\text{Error} = (W_A \text{ x MAPE}) + (W_F \text{ x } \frac{\sum_{j=1}^{3} Fj}{3})$$

$$= (0{,}95 \text{ x } 0{,}1931) + (0{,}05 \text{ x } \frac{2}{3}) = 0{,}2168 \text{ x } 100\% = 21{,}68\%$$

Jadi, dapat disimpulkan bahwa nilai harga gabah hasil prediksi dapat dilihat pada Tabel 3.51 dengan error yang dihasilkan sebesar 21,68%.

3.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional yang dilakukan meliputi analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

3.5.1 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras pada penelitian ini merupakan kebutuhan perangkat keras yang digunakan dalam pembangunan perangkat lunak ini. Adapun perangkat keras yang digunakan sebagai berikut:

a. Laptop dengan Processor 2.70 GHz
b. RAM 2 GB
c. HDD 500GB
d. Keyboard
e. Mouse

3.5.2 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak pada penelitian ini merupakan *tools* yang digunakan untuk pembangunan aplikasi ini. *Tools* yang digunakan adalah sebagai berikut:

a. OS Windows 8.1
b. Microsoft Visual Studio 2010
c. DBMS MySQL

3.5.3 Analisis Pengguna

Karakteristik pengguna yang dapat menjalankan sistem yang akan dibangun hanya terdapat satu jenis user yaitu seorang penguji. Penguji bekerja untuk menjalankan serta mengetahui hasil dari aplikasi yang dijalankan. Adapun spesifikasi pengguna yang dibutuhkan adalah sebagai berikut:

a. Menguasai dalam penggunaan komputer
b. Mengerti secara teknis *tools* dan *software* pendukung dalam menjalankan aplikasi.
c. Mengerti tahap-tahapan dalam menjalankan aplikasi.
d. Memahami proses dan kebutuhan dalam menggunakan aplikasi.

3.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional digunakan untuk menggambarkan secara umum dari sistem yang akan dirancang. Secara garis besar, kebutuhan fungsional yang akan dilakukan pada pembangunan sistem ini meliputi Diagram Konteks, *Data Flow Diagram* (DFD), Spesifikasi Proses, dan Kamus Data.

3.6.1 Diagram Konteks

Sistem yang akan dibangun digambarkan secara garis besar dengan menggunakan diagram konteks. Diagram konteks merupakan tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses yang menunjukkan sistem secara keseluruhan. Proses tersebut diberi nilai nol, dari diagram ini dapat diihat secara umum mengenai alur proses yang ditangani oleh sistem. Berikut adalah gambaran diagram konteks dari sistem prediksi tingkat potensi luas lahan pertanian yang akan dibangun dapat dilihat pada lampran halaman A-1.

3.6.2 Data Flow Diagram (DFD)

DFD merupakan sebuah teknik yang menggambarkan aliran data dan informasi yang digunakan sebagai perjalanan data dari masukan hingga menuju keluaran. DFD dapat menggambarkan arus data didalam system dengan terstruktur dan jelas.

3.6.2.1 DFD Level 1

Dalam DFD Level 1 berisi tentang proses-proses yang terjadi dalam aplikasi yang dibangun. Proses tersebut antara lain kelola data, proses PSO-SVR, dan lihat posisi partikel. Berikut adalah gambaran dari DFD Level 1 dapat dilihat pada lampiran halaman A-1.

3.6.2.2 DFD Level 2 Proses 1

Dalan DFD Level 2 Proses 1 ini merupakan penjabaran dari proses kelola data. Proses yang terjadi antara lain menampilkan keseluruhan dataset, menambahkan data, mengubah data, dan menghapus data. Beirikut gambaran dari DFD Level 2 Proses 1 dapat dilihat pada lampiran halaman A-2.

3.6.2.3 DFD Level 2 Proses 2

Dalam DFD Level 2 Proses 2 ini merupakan penjabaran dari proses PSO-SVR. Proses yang terjadi antara lain normalisasi data, k-fold cross validation, pelatihan SVR dan penghitungan error, pencarian pBest dan gBest, serta perbaruan posisi dan kecepatan partikel. Berikut adalah gambaran dari DFD Level 2 Proses 2 dapat dilihat pada lampiran halaman A-3.

3.6.2.4 DFD Level 2 Proses 4

Dalam DFD Level 2 Proses 4 ini merupakan penjabaran dari proses prediksi harga. Proses yang terjadi antara lain ambil parameter, menentukan perbandingan data latih dan uji, kemudian prediksi SVR. Berikut gambaran dari DFD Level 2 Proses 4 dapat dilihata pada lampiran A-4.

3.6.3 Spesifikasi Proses

Spesifikasi proses merupakan alat bantu sistem yang akan menjelaskan perilaku-perilaku proses yang ada dalam *Data Flow Diagram* (DFD) dan dibangun sesuai dengan kebutuhan sistem. Spesifikasi proses sistem ini dapat dilihat pada lapiran halaman A-4.

3.6.4 Kamus Data

Kamus data berisi tentang data-data yang terlibat dalam sistem yang dibangun. Berikut adalah data yang terlibat dalam sistem dapat dilihat pada lampiran halaman A-9.

3.7 Perancangan Sistem

Perancangan merupakan penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam suatu kesatuan yang utuh dan berfungsi. Perancangan sistem ini terdiri dari perancangan struktur data, struktur menu, dan antarmuka.

3.7.1 Struktur Tabel

Struktur Data merupakan spesifikasi detail mengenai tabel data yang digunakan dalam pembangunan sistem ini. Berikut adalah struktur data yang digunakan dapat dilihat pada lampiran halaman A-10.

3.7.2 Struktur Menu

Perancangan struktur menu merupakan gambaran secara garis besar dari setiap menu yang terdapat pada sistem. Perancangan struktur menu pada sistem ini dapat dilihat pada lampiran halaman A-10.

3.7.3 Perancangan Antarmuka

Perancangan antar muka berisi tentang gambaran dari setiap menu dan sub menu yang terdapat pada sistem dan berfungsi untuk lebih memudahkan user dalam memahami fungsi yang ada serta cara menggunakannya. Perancangan antar muka pada sistem ini dapat dilihat pada lampiran halaman A-11.

3.7.4 Perancangan Pesan

Perancangan pesan merupakan *pop-up windows* yang berasal dari sistem yang menunjukkan notifikasi atau peringatan. Perancangan pesan dalam sisitem ini dapat dilihat pada halaman lampiran A-15.

3.7.4 Jaringan Semantik

Jaringan semantik digunakan untuk menunjukan alur dari setiap menu dalam sistem yang dibangun, berikut jaringan semantik yang terdapat dalam pada sistem prediksi harga gabah ini dapat dilihat pada lampiran halaman A-18.

Berdasarkan gambar jaringan semantik diatas, berikut adalah tabel yang akan menjelaskan setiap simbolnya dapat dilihat pada lampiran halaman A-18.