# BAB III
# METODE PENELITIAN

## 3.1 Prosedur Penelitian

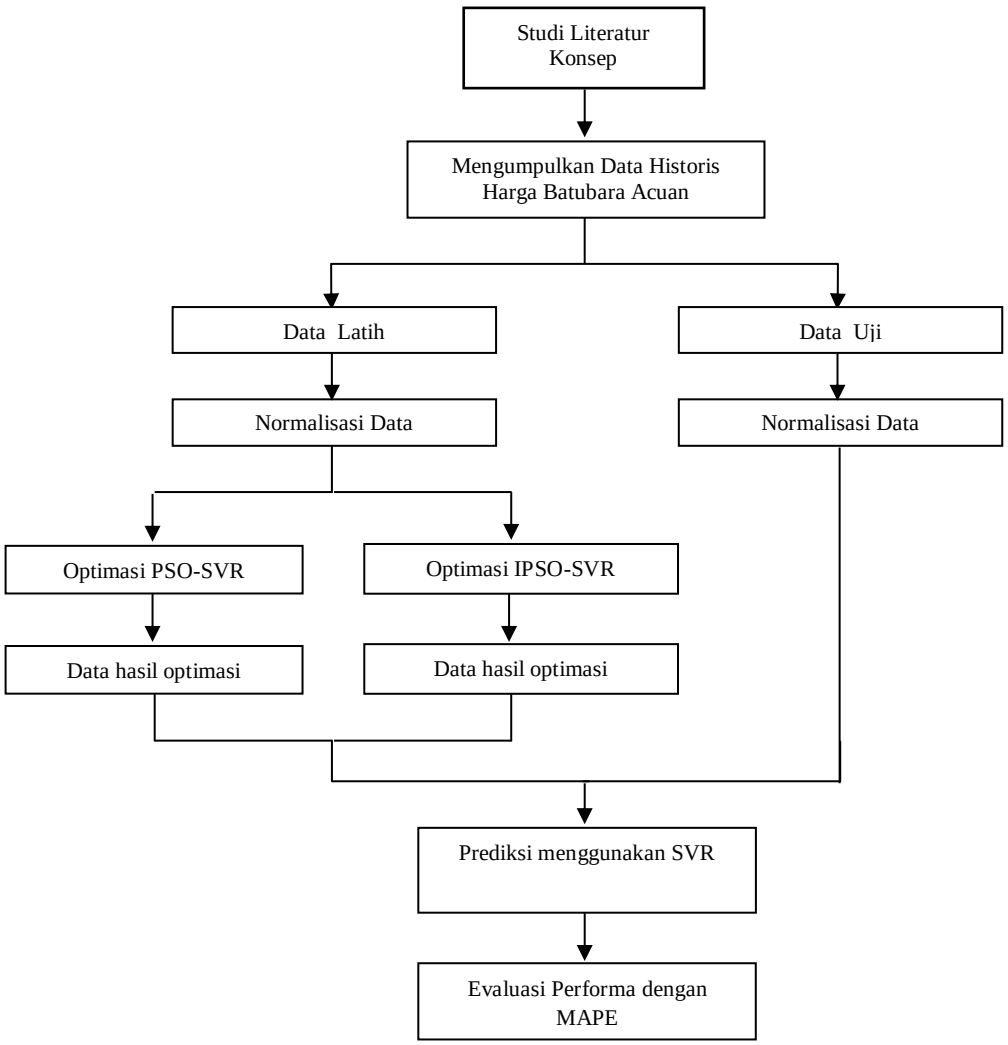
Pada subbab ini akan membahas proses yang dilakukan dalam penelitian ini yaitu tentang metode *Support Vector Regression* (SVR) yang dioptimasi dengan algoritma *Particle Swarm Optimization* (PSO) dan metode *Support Vector Regression* (SVR) yang dioptimasi dengan algoritma *Improved-Particle Swarm Optimization* (IPSO) untuk prediksi harga batubara acuan. Hal pertama yang dilakukan yaitu mengumpulkan data. Data yang digunakan untuk memprediksi harga batubara acuan dalam penelitian ini merupakan data sekunder yang diperoleh dari *website* resmi Kementerian Energi dan Sumber Daya Mineral (minerba.esdm.go.id). Data tersebut merupakan data harga batubara acuan per bulan periode bulan Januari 2009 sampai dengan bulan Oktober 2019 sebanyak 130 data. Data ini akan digunakan untuk memprediksi harga batubara acuan bulan selanjutnya yang akan dibahas dalam penelitian ini. Langkah-langkah yang dilakukan dalam penelitian ini yaitu pertama menentukan parameter awal dan data masukkan yang akan digunakan. Langkah selanjutnya adalah data dinormalisasi menggunakan metode normalisasi min-max. dengan menggunakan rumus sebagai berikut (Kumar & Thenmozhi, 2006):

$$\boldsymbol{x'} = \left(\frac{\boldsymbol{x - x\,min}}{\boldsymbol{x\,max - x\,min}}\right) \quad (3.\ 1)$$

dimana $x'$ merupakan hasil normalisasi data, $x$ merupakan nilai data yang akan dinormalisasi, $x\,max$ merupakan nilai maksimum dari dataset yang digunakan dan $x\,min$ merupakan nilai minimum dari dataset yang digunakan.

Kemudian data hasil normalisasi dioptimasi menggunakan metode PSOSVR dan metode IPSOSVR untuk mendapatkan nilai parameter yang optimal. Langkah terakhir nilai parameter optimal dari hasil proses optimasi akan digunakan untuk prediksi menggunakan SVR. Hasil prediksi tersebut dievaluasi keakuratanya menggunakan *Mean Absolute Percentage Error* (MAPE) sehingga dapat diketahui metode mana yang baik dalam memprediksi harga batubara acuan.

Berikut penjelasan singkat alur prosedur penelitian yang akan dilakukan dalam bentuk diagram:




**Gambar 3. 1 Tampilan Alur Prosedur Penelitian**

## 3.2 *Particle Swarm Optimization* (PSO)

*Particle swarm optimization* (PSO) pertama kali dikenalkan oleh Dr. Kennedy dan Dr. Eberhart pada tahun 1995. PSO merupakan Teknik *metaheuristic* yang meniru perilaku sosial kawanan burung dan sekelompok ikan dengan tiap individunya dimisalkan dengan partikel (Kennedy & Eberhart, 1995). Perilaku sosial yang dimaksud yaitu terdiri dari perilaku tiap individu itu sendiri serta pengaruh perilaku individu lain yang terdapat dalam suatu kelompok (Kennedy & Eberhart, 1995). Misalnya seekor burung dalam mencari jalan tercepat menuju sumber makanan,

mereka menggunakan kecerdasannya (*intelligence*) sendiri dan dipengaruhi pula oleh partikel lain dalam suatu kelompok. Setiap partikel harus memiliki kecerdasan sendiri dan komunikasi antar partikel.

Secara umum PSO dikenal sebagai alat sederhana dengan parameter minimal untuk mencari karakteristik optimum dengan bantuan pencarian lokal dan global di ruang fitur secara iteratif (Seal, Ganguly, Bhattacharjee, Nasipuri, & Martin, 2015). Dalam PSO, setiap partikel memiliki dua properti yaitu kecepatan dan posisi (Kennedy & Eberhart, 1995). Sekelompok partikel acak bergerak di ruang solusi dengan memperbarui kecepatan dan posisi secara iteratif hingga kondisi konvergen dipenuhi (Kennedy & Eberhart, 1995). Untuk melihat kecocokan nilai dengan fungsi yang akan dioptimisasi digunakan sebuah fungsi tertentu disebut dengan *fitness function* (Kennedy & Eberhart, 1995).

Pergerakan tiap partikel di ruang solusi dipengaruhi oleh kecepatan masing – masing partikel, posisi terbaik yang diperoleh oleh partikel itu sendiri (*pbest*) dan posisi terbaik yang diperoleh dari kelompok partikel (*gbest*). Berikut adalah rumus utuk memperbarui kecepatan dan posisi tiap partikel (Qasim & Algamal, 2018):

$$V_i(t+1) = w \times V_i(t) + c_1 \times r_1 \times [\, pbest_i(t) - X_i(t)] + \boldsymbol{c_2 \times r_2 \times [gbest(t) - X_i(t)\,]} \quad (3.\,2)$$

$$\boldsymbol{X_i(t+1) = X_i(t) + V_i(t+1)} \quad (3.\,3)$$

dimana $V_i(t)$ merupakan kecepatan partikel *i* pada iterasi ke *t*, $X_i(t)$ merupakan posisi partikel *i* pada iterasi ke *t*. $pbest_i(t)$ merupakan posisi terbaik yang diperoleh oleh partikel *i* pada iterasi sejauh *t*, $gbest(t)$ merupakan posisi terbaik dari sekelompok partikel pada iterasi sejauh *t*, *w* merupakan bobot inersia yang nilainya terletak di [0,1], $c_1$ dan $c_2$ merupakan koefisien akselerasi yang nilainya terletak di [0,4], $r_1$ dan $r_2$ merupakan bilangan acak yang nilainya terletak di [0,1].

### 3.2.1 Inisialisasi Partikel

Partikel merupakan individu dalam suatu *swarm* atau populasi yang merepresentasikan solusi penyelesaian masalah (Cholissodin & Riyandani, 2016).

Setiap partikel memiliki posisi dan kecepatan yang ditentukan oleh representasi solusi pada saat itu.

1. Inisialisasi Posisi Partikel

   Pada proses inisialisasi posisi partikel diasumsikan bahwa sebuah partikel untuk setiap dimensinya harus berada dalam domain yang didefinisikan oleh dua vektor, yaitu $x_{min}$ dan $x_{max}$. $x_{min}$ mewakili batas bawah setiap dimensi dan $x_{max}$ mewakili batas atas setiap dimensi. Metode inisialisasi untuk posisi partikel adalah (Engelbrecht A. P., 2007).

$$\boldsymbol{x(0) = x_{min,j} + r_j\left(x_{max,j} - x_{min,j}\right)}$$
$$\boldsymbol{\forall j = 1, \dots, n_x \quad \forall i = 1, \dots, n_s} \qquad (3.\ 4)$$

   dimana $x$ merupakan posisi partikel (merepresentasikan nilai *C*, ε , σ, *cLR*, $\lambda$) dan $r_j$merupakan nilai acak dalam range [0,1].

2. Inisialisasi Kecepatan Partikel

   Pada proses kecepatan awal partikel diinisialisasikan menjadi nol $v_i(0) = 0$. Dalam implementasi PSO, terkadang ditemukan bahwa kecepatan partikel bergerak ke nilai yang besar dengan cepat. Akibatnya, partikel tersebut memiliki kecenderungan untuk keluar dai ruang batas pencarian. Oleh karena itu, perlu adanya pembatsan kecepatan minimum dan maksimum untuk mencegah partkel bergerak terlalu jauh melampaui ruang pencarian. Inisialisasi pembatasan kecepatan ditentukan dengan menggunakan persamaan berikut (Marini & Walczak, 2015):

$$\boldsymbol{v_{max,j} = k\ \frac{(x_{max,j} - x_{min,j})}{2}} \qquad (3.\ 5)$$

   Dimana $v_{max,j}$ merupakan kecepatan maksimum dimensi j, $k$ merupakan nilai acak dari 0-1, $x_{max,j}$ mewakili batas atas dimensi j dan $x_{min,j}$ mewakili batas bawah dimensi j.

   Batasan kecepatan yang digunakan adalah sebagai berikut (Marini & Walczak, 2015): Jika $V_i(t) > v_{max,j}$ maka $V_i(t) = v_{max,j}$.

### 3.2.2 Bobot Inersia

Bobot inersia dikenalkan oleh Shi dan Eberhart (1998) dengan tujuan sebagai sebuah mekanisme untuk mengontrol dampak dari perubahan kecepatan yang diberikan oleh partikel (Kennedy & Eberhart, 1995). Nilai $\boldsymbol{w_{max}}$ dan $\boldsymbol{w_{min}}$ yang digunakan dan terbukti dapat meningkatkan solusi optimum pada pencarian banyak masalah menurut (Cholissodin & Riyandani, 2016) adalah 0,9 dan 0,4.

Bobot inersia $\boldsymbol{w}$ diperbaharui untuk mendapatkan nilai $\boldsymbol{w}$ yang adaptif untuk setiap iterasi, sehingga nilainya bias dinamis dan mampu meningkatkan hasil optimasi yang diharapkan, semakin besar nilai iterasinya, maka nilai $\boldsymbol{w}$ akan semakin kecil, dan sebaliknya, jika iterasinya kecil, maka nilai $\boldsymbol{w}$ akan cenderung lebih besar. Artinya, jika nilai $\boldsymbol{w}$ semakin besar, maka partikel lebih difokuskan ke arah eksplorasi, tetapi ketika nilai $\boldsymbol{w}$ semain kecil, maka partikel lebih difokuskan ke arah eksploitasi (Cholissodin & Riyandani, 2016). Menghitung nilai bobot inersia baru ditentukan dengan menggunakan perumusan sebagai berikut (Zhang, Qiu, & Mu, 2016):

$$\boldsymbol{w = w_{max} - \frac{w_{max} - w_{min}}{t_{max}} * t} \qquad (3.\ 6)$$

dimana $w$ merupakan bobot inersia, $w_{max}$ merupakan batas atas bobot inersia, $w_{min}$ merupakan batas bawah bobot inersia, $t_{max}$ merupakan iterasi maksimal dan $t$ merupakan iterasi saat ini.

## 3.3 *Support Vector Regression* (SVR)

*Support Vector Machine* (SVM) pertama kali ditemukan oleh Vlandimir Vapnik dan rekannya pada tahun 1995. SVM merupakan salah satu *Machine learning* yang termasuk dalam metode *Supervised Learning* digunakan untuk menyelesaikan masalah klasifikasi (Vapnik, 1999). SVM dikembangkan untuk menyelesaikan kasus regresi yang disebut *Support Vector Regression* (SVR). Ide utama dari SVR yaitu mencari sebuah fungsi $y(x)$ sebagai *hyperplane* berupa fungsi regresi yang akan memaksimumkan margin (Smola & Scholkopf, 2004).

Misal diberikan data *training* $\{x_i, y_i\}$ , $i = 1,2, \dots, N$ dengan $x_i \in R^d$ merupakan *vektor input* berdimensi d (banyak fitur) dan $y_i \in R$ merupakan nilai target aktual atau nilai pada saat ini. Selanjutnya akan mencari fungsi $y(x)$ yang memiliki *error* ($\varepsilon$) yang sekecil mungkin dan tidak melebihi ($\varepsilon$) dari nilai target aktual ($y_i$) dari semua data *training*. Bentuk umum dari SVR adalah sebagai berikut (Smola & Scholkopf, 2004) :

$$\boldsymbol{y(x) = \langle w, x \rangle + b} \tag{3. 7}$$

dengan, $y(x)$ merupakan fungsi regresi, $x$ merupakan vector *input*, $w$ merupakan parameter bobot dan $b$ merupakan parameter bias berupa skalar.

Data - data yang berada pada *hyperplane* akan memenuhi ketentuan $y(x) = \langle w, x \rangle + b = 0$. Tujuan utama dari SVR adalah mencari maksimum dari margin $\frac{1}{\|w\|}$, dengan demikian dapat diperoleh masalah minimalisasi yang ekuivalen yaitu:

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

dengan kendala:

$$y_i - \langle w, x_i \rangle - b \leq \varepsilon$$

$$\boldsymbol{\langle w, x_i \rangle + b - y_i \leq \varepsilon} \tag{3. 8}$$

dimana dengan, $y_i$ merupakan fungsi regresi, $x_i$ merupakan vector *input*, $w$ merupakan parameter bobot, $b$ merupakan parameter bias berupa scalar.

Pada persamaan (3.8) diasumsikan bahwa fungsi $y(x)$ dapat mendekati semua pasangan $(x_i, y_i)$ dengan presisi $\varepsilon$. Dengan kata lain semua data target $y_i$ yang berada pada $\epsilon$-*tube* akan memenuhi kondisi sebagai berikut:

$$\boldsymbol{y(x_i) - \varepsilon \leq y_i \leq y(x_i) + \varepsilon} \tag{3. 9}$$

Untuk data target yang berada diluar $\varepsilon$-tube akan dilakukan pendekatan *soft margin* yaitu penambahan variabel *slack* $\xi_i$ dan $\hat{\xi}_i$, dimana

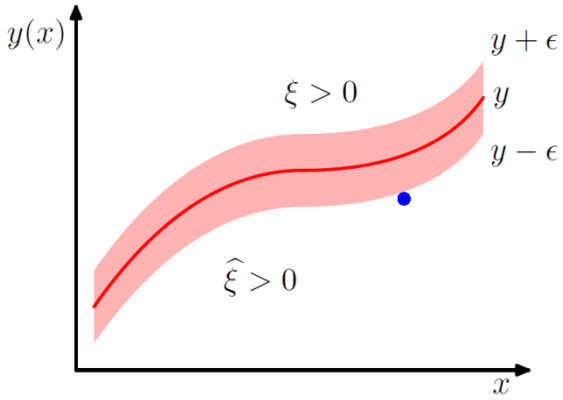
$\xi_i > 0$ untuk $y_i > y(x_i) + \varepsilon$

Dan

$\hat{\xi}_i < 0$ untuk $y_i < y(x_i) + \varepsilon$

dengan demikian persamaan (3.9) dapat dituliskan dalam bentuk sebagai berikut:

$$\boldsymbol{y(x_i) - \varepsilon - \hat{\xi}_i \leq y_i \leq y(x_i) + \varepsilon + \xi_i} \quad (3.10)$$




Gambar 3. 2 Ilustrasi SVR

Gambar (3.2) Ilustrasi dari SVR, menunjukkan kurva regresi dan $\epsilon$-*insenstitif* '*tube'*, juga menunjukkan variabel *slack* $\xi_i$ dan $\hat{\xi}_i$.

Sumber : Bishop (2006), *Pattern Recognation and Machine Learning*

Dengan demikian model matematis dari SVR dapat dituliskan sebagai berikut (Smola & Scholkopf, 2004):

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i)$$

Dengan kendala:

$$y_i - \langle w, x_i\rangle - b \leq \varepsilon + \xi_i$$
$$\langle w, x_i\rangle + b - y_i \leq \varepsilon + \hat{\xi}_i$$
$$\boldsymbol{\xi_i, \hat{\xi}_i \geq 0} \quad (3.11)$$

Masalah minimalisasi di atas dapat diselesaikan sebagai masalah pemograman kuadrat dengan menggunakan metode *lagrange*. Ide utama dari metode Lagrange ini yaitu menambahkan pengali lagrange $\alpha_i \geq 0$, $\hat{\alpha}_i \geq 0$, $\mu_i \geq 0$, dan $\hat{\mu}_i$ pada setiap kendala, setiap kendala memiliki satu pengali lagrange. Fungsi *Lagrangian* dari model matematis SVR dapat dituliskan dalam bentuk sebagai berikut:

$$L(w, b, \alpha, \hat{\alpha}, \mu, \hat{\mu}) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i) - \sum_{i=1}^{n}(\mu_i\xi_i + \hat{\mu}_i\hat{\xi}_i)$$
$$- \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b)$$
$$\boldsymbol{- \sum_{i=1}^{n}\hat{\alpha}_i(\varepsilon + \hat{\xi}_i + y_i - \langle w, x_i\rangle - b)} \quad (3.12)$$

Solusi optimal persamaan (3.12) diperoleh apabila solusi tersebut memenuhi syarat *Karush-Kuhn-Thucker* (KKT). Syarat KKT adalah sebagai berikut:

$$\boldsymbol{\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) = 0} \quad (3.\ 13)$$

$$\boldsymbol{\widehat{\alpha_i}\left(\varepsilon + \widehat{\xi}_i + y_i - \langle w, x_i\rangle - b\right) = 0}$$

(3. 14)

$$\boldsymbol{(C - \alpha_i)\xi_i = 0} \quad (3.\ 15)$$

$$\boldsymbol{(C - \widehat{\alpha_i})\widehat{\xi}_i = 0} \quad (3.\ 16)$$

Selanjutnya persamaan (3.13), (3.14), (3.15), dan (3.16) diturunkan secara parsial terhadap $w, b, \xi_i$, dan $\widehat{\xi}_i$ dan diperoleh sebagai berikut:

$$\boldsymbol{\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}(\alpha_i - \widehat{\alpha_i})x_i = 0 \leftrightarrow w = \sum_{i=1}^{n}(\alpha_i - \widehat{\alpha_i})x_i} \quad (3.\ 17)$$

$$\boldsymbol{\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}\widehat{\alpha_i} = 0 \leftrightarrow \sum_{i=1}^{n}(\alpha_i - \widehat{\alpha_i}) = 0} \quad (3.\ 18)$$

$$\boldsymbol{\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \xi_i = 0 \leftrightarrow C = \alpha_i + \xi_i} \quad (3.\ 19)$$

$$\boldsymbol{\frac{\partial L}{\partial \xi_i} = C - \widehat{\alpha_i} - \widehat{\xi}_i = 0 \leftrightarrow C = \widehat{\alpha_i} + \widehat{\xi}_i} \quad (3.\ 20)$$

Selanjutnya hasil turunan yaitu persamaan (3.17), (3.18), (3.19) dan (3.20) disubstitusikan kedalam persamaan (3.12). Bentuk dual dari SVR sebagai berikut:

$$L(\alpha, \hat{\alpha}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \widehat{\alpha_i})\left(\alpha_j - \widehat{\alpha_j}\right)\langle x_i, x_j\rangle$$

$$-\varepsilon\sum_{i=1}^{n}(\alpha_i + \widehat{\alpha_i}) + \sum_{i=1}^{n}y_i(\alpha_i - \widehat{\alpha_i})$$

dengan kendala:

$$\sum_{i=1}^{n}(\alpha_i - \widehat{\alpha_i}) = 0$$

$$0 \le \alpha_i \le C$$

$$\boldsymbol{0 \le \widehat{\alpha_i} \le C} \quad (3.\ 21)$$

Dari bentuk dual pemograman (3.21) misalkan diperoleh solusi $\alpha_i^*$ dan $\widehat{\alpha_i}^*$ maka:

$$\boldsymbol{w = \sum_{i=1}^{n}(\alpha_i^* - \widehat{\alpha_i}^*)x_i} \quad (3.\ 22)$$

Maka persamaan (3.7) dapat ditulis sebagai berikut:

$$\boldsymbol{y(x) = \langle w, x\rangle + b = \sum_{i=1}^{n}(\alpha_i^* - \widehat{\alpha_i}^*)\langle x_i, x_j\rangle + b} \qquad (3.\ 23)$$

Data yang berperan dalam model SVR hanyalah data dengan nilai $\alpha_i^* \neq 0$ dan $\widehat{\alpha_i}^* \neq 0$ atau disebut dengan *support vector* sehingga dapat ditulis sebagai:

$$\boldsymbol{y(x) = \sum_{m\in S}(\alpha_m^* - \widehat{\alpha_m}^*)\langle x, x_m\rangle + b} \qquad (3.\ 24)$$

Dimana S adalah himpunan indeks dari *support vectors*. *Support vector* yang berperan dalam model adalah data training yang terletak tepat pada *boundary* atas ($\xi_i = 0$), diatas *boundary* atas ($\xi_i > 0$), tepat pada *boundary* bawah $\left(\widehat{\xi_i} = 0\right)$, dan dibawah *boundary* bawah $\left(\widehat{\xi_i} > 0\right)$. Dengan demikian, jumlah *support vector* akan menurun dengan meningkatnya nilai $\varepsilon$. Nilai parameter b dapat ditentukan berdasarkan model yang hanya mengandung *support vector* yang memenuhi kondisi seperti berikut:

1. $(C - a_i)\xi_i = 0$ sehingga $\xi_i = 0$
2. $a_n(\epsilon + \xi_i + y(x_i) - y_i) = 0$ sehingga $\epsilon + y(x_i) - y_i = 0$

Maka dari persamaan (3.7) dapat diperoleh nilai b sebagai berikut:

$$b = y(x_i) - \sum_{i=1}^{n}(\alpha_i^* - \widehat{\alpha_i}^*)\langle x_i, x_j\rangle$$
$$= \boldsymbol{y_i - \epsilon - \sum_{i=1}^{n}(\alpha_i^* - \widehat{\alpha_i}^*)\langle x_i, x_j\rangle}$$

(3. 25)

Untuk beberapa kasus data yang tidak bisa dipisahkan secara linear oleh *hyperplane* meskipun telah ditambahkan fungsi penalti atau variabel *slack*. Untuk itu dibutuhkan sebuah fungsi kernel $k\left(x_i, x_j\right) = \langle\Phi(x_i), \Phi\left(x_j\right)\rangle$, sebagaimana yang telah dijelaskan di sebelumnya fungsi kernel merupakan fungsi yang memetakan vektor data pada ke ruang fitur sedemikian sehingga data dapat dipisahkan secara linear di ruang fitur oleh *hyperplan*e. Pada ruang fitur, hasil kali dalam antara dua vektor input dapat diganti dengan fungsi kernel. Oleh karena itu persamaan dual *Lagrange* (3.21) dapat diubah kedalam bentuk berikut:

$$L(\alpha, \hat{\alpha}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \hat{\alpha_i})\left(\alpha_j - \hat{\alpha_j}\right)k\left(x_i, x_j\right)$$
$$-\varepsilon\sum_{i=1}^{n}(\alpha_i + \hat{\alpha_i}) + \sum_{i=1}^{n}y_i(\alpha_i - \hat{\alpha_i})$$

Dengan kendala:

$$\sum_{i=1}^{n}(\alpha_i - \widehat{\alpha_i}) = 0$$

$$0 \le \alpha_i \le C$$

$$\boldsymbol{0 \le \widehat{\alpha_i} \le C} \quad (3.\ 26)$$

Dari bentuk dual pemograman diatas misalkan diperoleh solusi $\alpha_i{}^*$ dan $\widehat{\alpha_i}^*$ maka:

$$\boldsymbol{w = \sum_{i=1}^{n}(\alpha_i{}^* - \widehat{\alpha_i}^*)\phi(x_i)} \quad (3.\ 27)$$

$$\boldsymbol{y(x_i) = \sum_{i=1}^{n}(\alpha_i{}^* - \widehat{\alpha_i}^*)k(x_i, x_j) + b} \quad (3.\ 28)$$

Data yang berperan dalam model SVR hanyalah data dengan $\alpha_i{}^* \neq 0$ dan $\widehat{\alpha_i}^* \neq 0$ atau disebut dengan *support vector* sehingga dapat ditulis sebagai:

$$\boldsymbol{y(x) = \sum_{m \in S}(\alpha_m{}^* - \widehat{\alpha_m}^*)k(x_i, x_j) + b} \quad (3.\ 29)$$

Dimana S adalah himpunan indeks dari *support vectors*. *Support vector* yang berperan dalam model adalah data training yang terletak tepat pada *boundary* atas $(\xi_i = 0)$, diatas *boundary* atas $(\xi_i > 0)$, tepat pada *boundary* bawah $(\widehat{\xi_i} = 0)$, dan dibawah *boundary* bawah $(\widehat{\xi_i} > 0)$. Dengan demikian, jumlah *support vector* akan menurun dengan meningkatnya nilai $\varepsilon$. Nilai parameter b dapat ditentukan berdasarkan model yang hanya mengandung *support vector* yang memenuhi kondisi seperti berikut:

1. $(C - a_i)\xi_i = 0$ sehingga $\xi_i = 0$
2. $a_n(\epsilon + \xi_i + y(x_i) - y_i) = 0$ sehingga $\epsilon + y(x_i) - y_i = 0$

Dengan persamaan (3.7)

$$b = y(x_i) - \sum_{i=1}^{n}(\alpha_i{}^* - \widehat{\alpha_i}^*)k\langle x_i, x_j\rangle$$

$$\boldsymbol{= y_i - \epsilon - \sum_{i=1}^{n}(\alpha_i{}^* - \widehat{\alpha_i}^*)k\langle x_i, x_j\rangle} \quad (3.\ 30)$$

Namun terkadang sulit untuk menemukan parameter *lagrange* $\alpha_i{}^*$ dan $\widehat{\alpha_i}^*$ pada masalah pemograman dual untuk kasus linear (3.21) maupun nonlinear (3.26), maka untuk memudahkan menyelesaikan persamaan *lagrangian* diatas dimisalkan $\beta = \hat{\alpha} - \alpha$ dengan menggunakan hubungan $\alpha_i\hat{\alpha}_i = 0$ (Taylor & Cristianini, 2013).

### 3.4 Optimasi PSOSVR

Proses optimasi PSOSVR merupakan proses optimasi menggunakan algoritma PSO dengan SVR sebagai evaluasi fungsi *fitness*nya, proses optimasi dilakukan dengan tujuan untuk mengetahui nilai parameter yang paling optimal.

Data yang digunakan dalam proses optimasi PSOSVR ini merupakan data yang telah melalui proses normalisasi. Parameter yang akan dioptimasi adalah parameter *C*, ε , σ, *cLR*, dan $\lambda$. Berikut penjelasan mengenai proses optimasi PSOSVR :

1. Melakukan K-Fold Cross Validation

   K-Fold Cross Validation merupakan proses mempersiapkan data yang akan digunakan untuk melatih SVR (Hastie, Tibshirani, & Friedman, 2009). K-Fold Cross Validation dilakukan dengan tujuan untuk mengestimasi tingkat kesalahan. K-Fold Cross Validation dilakukan dengan cara mengelompokkan antara data latih dan data uji dari keseluruhan data. Proses pengelompokkan data latih dan data uji dilakukan dengan tahapan sebagai berikut: (1) mengacak urutan hasil normalisasi data, (2) mencari indeks awal dan akhir data uji, dan (3) mendefinisikan data latih dan data uji sebanyak K kali.

2. Melakukan Inisialisasi Partikel

   Proses inisialisasi posisi partikel dilakukan dengan menggunakan persamaan (3.4), untuk proses kecepatan awal partikel diinisialisasikan menjadi nol $v_i(0) = 0$ dan untuk inisialisasi pembatasan kecepatan dilakukan menggunakan persamaan (3.5).

3. Mengevaluasi Fungsi *Fitness*

   Evaluasi fungsi *fitness* dilakukan dengan pelatihan SVR. Tahapan yang dilakukan pada evaluasi fungsi *fitness* adalah (1) memilih fitur yang akan digunakan berdasarkan fitur yang dipilih, (2) melakukan *sequensial learning*, (3) menguji model regresi dengan data uji dan (4) menghitungan nilai error dan nilai cost.

4. Mencari Nilai *pBest*

   *pBest* merupakan posisi terbaik yang pernah dicapai partikel. Proses pencarian nilai pBest dilakukan dengan cara membandingkan nilai cost pBest dengan partikel iterasi saat ini. Apabila nilai cost pBest lebih kecil dari nilai cost partikel

iterasi saat ini, maka nilai posisi pBest tetap. Namun, apabila nilai cost pBest lebih besar dari nilai cost partikel iterasi saat ini, maka nilai posisi pBest akan digantikan oleh nilai posisi partikel saat ini.

5. Mencari Nilai *gBest* dan Nilai Bobot Inersia Baru

   *gBest* merupakan posisi terbaik partikel. Proses pencarian nilai gBest dilakukan dengan cara membandingkan nilai cost gBest dan pBest.Apabila nilai cost gBest lebih kecil dari nilai cost pBest, maka nilai posisi gBest tetap. Namun, apabila nilai cost gBest lebih besar dari nilai cost pBest, maka nilai posisi gBest digantikan oleh nilai posisi pBest. Proses mencari nilai bobot inersia baru dilakukan dengan menggunakan persamaan (3.6).
6. Memperbarui Kecepatan Partikel

   Proses memperbarui kecepatan partikel dilakukan dengan menggunakan persamaan (3.2)
7. Memperbarui Posisi Partikel

   Proses memperbarui posisi partikel dilakukan dengan menggunakan persamaan (3.3)
8. Mengulangi langkah 1-7 sampai diperoleh kondisi yang konvergen yaitu kondisi pada saat maksimal iterasi.
9. Menentukan hasil optimal parameter SVR dan jumlah fitur pilihan dari nilai gBest.

## 3.5 Optimasi IPSOSVR

*Improved-Particle Swarm Optimization (*IPSO*)* merupakan pengembangan dari algoritma *Particle swarm optimization* (PSO) untuk mencegah konvergensi dini. Pada algoritma PSO konvensional, konvergensi partikel terjadi sangat cepat, namun pergerakan dari partikel yang ada hanya terjadi pada area lokal optimal dan global optimal (Yan, Wu, Liu, & Huang, 2013).

Data yang digunakan dalam proses optimasi PSOSVR ini merupakan data yang telah melalui proses normalisasi. Parameter yang akan dioptimasi adalah parameter *C*, ε , σ, *cLR*, dan $\lambda$. Berikut penjelasan mengenai proses optimasi IPSOSVR :

1. Melakukan K-Fold Cross Validation

   K-Fold Cross Validation merupakan proses mempersiapkan data yang akan digunakan untuk melatih SVR (Hastie, Tibshirani, & Friedman, 2009). K-Fold Cross Validation dilakukan dengan tujuan untuk mengestimasi tingkat kesalahan. K-Fold Cross Validation dilakukan dengan cara mengelompokkan antara data latih dan data uji dari keseluruhan data. Proses pengelompokkan data latih dan data uji dilakukan dengan tahapan sebagai berikut: (1) mengacak urutan hasil normalisasi data, (2) mencari indeks awal dan akhir data uji, dan (3) mendefinisikan data latih dan data uji sebanyak K kali.

2. Melakukan Inisialisasi Partikel

   Proses inisialisasi posisi partikel dilakukan dengan menggunakan persamaan (3.4), untuk proses kecepatan awal partikel diinisialisasikan menjadi nol $v_i(0) = 0$ dan untuk inisialisasi pembatasan kecepatan dilakukan menggunakan persamaan (3.5).

3. Mengevaluasi Fungsi *Fitness*

   Evaluasi fungsi *fitness* dilakukan dengan pelatihan SVR. Tahapan yang dilakukan pada evaluasi fungsi *fitness* adalah (1) memilih fitur yang akan digunakan berdasarkan fitur yang dipilih, (2) melakukan *sequensial learning*, (3) menguji model regresi dengan data uji dan (4) menghitungan nilai error dan nilai cost.

4. Mencari Nilai pBest

   *pBest* merupakan posisi terbaik yang pernah dicapai partikel. Proses pencarian nilai pBestdilakukan dengan cara membandingkan nilai cost pBest dengan partikel iterasi saat ini. Apabila nilai cost pBest lebih kecil dari nilai cost partikel iterasi saat ini, maka nilai posisi pBest tetap. Namun, apabila nilai cost pBest lebih besar dari nilai cost partikel iterasi saat ini, maka nilai posisi pBest akan digantikan oleh nilai posisi partikel saat ini.

5. Mencari Nilai gBest dan Nilai Bobot Inersia Baru

   *gBest* merupakan posisi terbaik partikel. Proses pencarian nilai gBest dilakukan dengan cara membandingkan nilai cost gBest dan pBest.Apabila nilai cost gBest lebih kecil dari nilai cost pBest, maka nilai posisi gBest tetap. Namun, apabila nilai cost gBest lebih besar dari nilai cost pBest, maka nilai posisi gBest digantikan oleh

nilai posisi pBest. Proses mencari nilai bobot inersia baru dilakukan dengan menggunakan persamaan (3.6).

6. Memperbarui Kecepatan Partikel

   (Zou, Jifeng, Chenlong, & Qiao, 2015) menambahkan $\lambda$ sebagai faktor konvergen yang diletakkan di depan bobot inersia, dimana $\lambda$=sin$^3\alpha$ dan $\alpha$=[0,$\pi$/8]. Proses memperbarui kecepatan partikel dilakukan dengan menggunakan persamaan:

$$V_i(t+1) = \lambda \times w \times V_i(t) + c_1 \times r_1 \times [\, pbest_i(t) - X_i(t)] + \boldsymbol{c_2 \times r_2 \times [gbest(t) - X_i(t)\,]} \quad (3.31)$$

7. Memperbarui Posisi Partikel

   Proses memperbarui posisi partikel dilakukan dengan menggunakan persamaan berikut (Zou, Jifeng, Chenlong, & Qiao, 2015):

$$\boldsymbol{X_i(t+1) = X_i(t) + \lambda \times w \times V_i(t+1)} \quad (3.32)$$

8. Mengulangi langkah 1-7 sampai diperoleh kondisi yang konvergen yaitu kondisi pada saat maksimal iterasi.
9. Menentukan hasil optimal parameter SVR dan jumlah fitur pilihan dari nilai gBest.

### 3.6 Prediksi dengan SVR

Dalam tahap ini nilai parameter optimal yang telah diperoleh pada tahap optimasi sebelumnya akan digunakan untuk memprediksi harga batubara acuan menggunakan metode SVR. Langkah-langkah prediksi sebagai berikut :

1. Memilih fitur masukan

   Fitur masukan disini berdasarkan hasil dari optimasi PSOSVR dan IPSOSVR

2. Penentuan data latih dan data uji
3. *Sequential Learning*

   Adapun langkah-langkahnya sebagai berikut (Vijayakumar, 1999):

a. Inisialisasi parameter SVR yang digunakan dan inisialisasi nilai awal $a_i$ dan $a_i^*$ sebesar 0
b. Menentukan matriks Hessian dengan persamaan (3.34) berikut

$$[\boldsymbol{R_{ij}}] = (\boldsymbol{K(x_i\,,\ x_j) + \lambda^2}) \quad (3.33)$$

Dimana :

$[R_{ij}]$ = matriks Hessian baris ke-I kolom ke-j

$K(x_i, x_j)$ = fungsi kernel

$\lambda$ = variabel scalar

Karena pada penelitian ini menggunakan kernel RBF, jadi persamaanya sebagai berikut :

$$[\boldsymbol{R_{ij}}] = \boldsymbol{exp}\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right) + \boldsymbol{\lambda^2} \quad (3.\ 34)$$

Keluaran dari matriks Hessian kemudian dimasukan pada persamaan dibawah untuk mencari nilai $\gamma$ yang berfungsi mengontrol kecepatan proses *learning* dimana *cLR* adalah kontanta *learning rate*

$$\boldsymbol{\gamma} = \frac{\boldsymbol{cLR}}{\boldsymbol{\max(R_{ij})}} \quad (3.\ 35)$$

c. Untuk setiap data latih, hitung nilai error dengan persamaan (3.36)

$$\boldsymbol{E_i} = \boldsymbol{y_i} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{a_i^*} - \boldsymbol{a_i})\, \boldsymbol{R_{ij}} \quad (3.\ 36)$$

Hitung perubahan nilai *Lagrange multiplier* dengan persamaan (3.37) dan (3.38)

$$\boldsymbol{\delta a_i^*} = \mathbf{min}\{\mathbf{max}(\boldsymbol{\gamma}(\boldsymbol{E_i} - \boldsymbol{\varepsilon}), -\boldsymbol{a_i^*}), \boldsymbol{C} - \boldsymbol{a_i^*}) \quad (3.\ 37)$$

$$\boldsymbol{\delta a_i} = \mathbf{min}\{\mathbf{max}(\boldsymbol{\gamma}(-\boldsymbol{E_i} - \boldsymbol{\varepsilon}), -\boldsymbol{a_i}), \boldsymbol{C} - \boldsymbol{a_i}) \quad (3.\ 38)$$

Hitung nilai *Lagrange multiplier* yang baru dengan persamaan (3.39) dan (3.40)

$$\boldsymbol{a_i^*(baru)} = \boldsymbol{\delta a_i^*} + \boldsymbol{a_i^*} \quad (3.\ 39)$$

$$\boldsymbol{a_i(baru)} = \boldsymbol{\delta a_i} + \boldsymbol{a_i} \quad (3.\ 40)$$

Dimana $y_i$ adalah nilai aktual ke-*i*, ε adalah *epsilon*, dan *C* adalah nilai kompleksitas

d. Ulangi langkah c sampai iterasi maksimum yang ditentukan atau telah mencapai konvergensi dengan syarat $\max(|\delta a_i^*|) < \varepsilon$ dan $\max(|\delta a_i|) < \varepsilon$.

4. Menguji model regresi

Yaitu proses memperoleh hasil regresi dari model regresi yang telah dibangun. Proses ini dimulai dengan menghitung fungsi regresi untuk memberikan

hasil prediksi dengan perkalian anatara nilai *Lagrange multiplier* terakhir yang dihasilkan dan matrik Hessian, dengan persamaan (Vijayakumar, 1999) :

$$\boldsymbol{f(x) = \sum_{i=1}^{n}(a_i^* - a_i)(K(x_i\ ,\ x_j) + \lambda^2)} \quad (3.\ 41)$$

Langkah selanjutnya adalah denormalisasi untuk mengembalikan data sehingga didaptkan *predicted value* menggunakan persamaan berikut (Smola & Scholkopf, 2004) :

$$\boldsymbol{x_i = f(x_i)(x_{max} - x_{min}) + x_{min}} \quad (3.\ 42)$$

Dimana :

$x_i$ = nilai data normal

$f(x_i)$ = hasil output ke-*i*

$x_{max}$ = data dengan nilai maksimum

$x_{min}$ = data dengan nilai minimum.

5. Menentukan nilai error

Yaitu proses menghitung tingkat akurasi hasil regresi. Proses ini dimulai dengan menghitung nilai *mean absolute percentage error* (MAPE) dengan menggunakan persamaan sebagai berikut (Khair, Fahmi, Hakim, Rahim, & Robbi, 2017) :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|PE| * 100\%$$

$$\boldsymbol{|PE| = \left|\frac{A_i - F_i}{A_i}\right|} \quad (3.\ 43)$$

Dimana :

n = banyak data

$A_i$ = nilai aktual

$F_i$ = nilai prediksi pada data ke-i

Hasil MAPE yang didapat selanjutnya dikategorikan ke dalam beberapa kriteria yang mengindikasikan hasil tersebut (Khair, Fahmi, Hakim, Rahim, & Robbi, 2017).

**Tabel 3. 1 Kriteria Nilai MAPE**

| Nilai MAPE | Kriteria |
|---|---|

| Nilai MAPE | Kriteria |
|---|---|
| <10% | Sangat Baik |
| 10% - 20% | Baik |
| 20% - 50% | Cukup |
| >50% | Buruk |

## 3.7 Kontruksi Perancangan Program Aplikasi

Subbab ini membahas mengenai rancangan data masukkan, data keluaran, dan tampilan program menggunakan bahasa pemograman *R* sebagai *beck-end* dan *PyQt* sebagai *front-end*.

### 3.7.1 Data Masukkan

Data masukkan pada program aplikasi PSOSVR dan IPSOSVR tersaji pada Tabel 3.2.

**Tabel 3. 2 Data Masukkan Program Aplikasi PSOSVR Dan IPSOSVR**

| Data Masukkan | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| HBA-F1 | f1 | Float | HBA 3 bulan sebelum target |
| HBA-F2 | f2 | Float | HBA 2 bulan sebelum target |
| HBA-F3 | f3 | Float | HBA 1 bulan sebelum target |
| HBA-Target | Target | Float | HBA yang akan di prediksi |
| C | C | Float | Kompleksitas Parameter |
| Epsilon | Epsilon | Float | ~ |
| Sigma | Sigma | Float | ~ |
| Clr | Clr | Float | Learning rate |
| Lambda | Lambda | Float | ~ |
| Nilai C1 | c1,c2 | Integer | ~ |
| Nilai W | wmax,wmin | Float | ~ |

### 3.7.2 Data Keluaran

Data keluaran yang akan ditampilkan dari hasil program aplikasi PSOSVR dan IPSOSVR adalah berupa nilai dari setiap parameter hasil optimasi, *cost*, nilai prediksi dan nilai *error* seperti yang tersaji pada tabel 3.3.

**Tabel 3. 3 Data Keluaran Program Aplikasi PSOSVR Dan IPSOSVR**

| Data Keluaran | Nama Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| *Cost* | Cost | Float | Hasil *error* dari pencarian parameter |

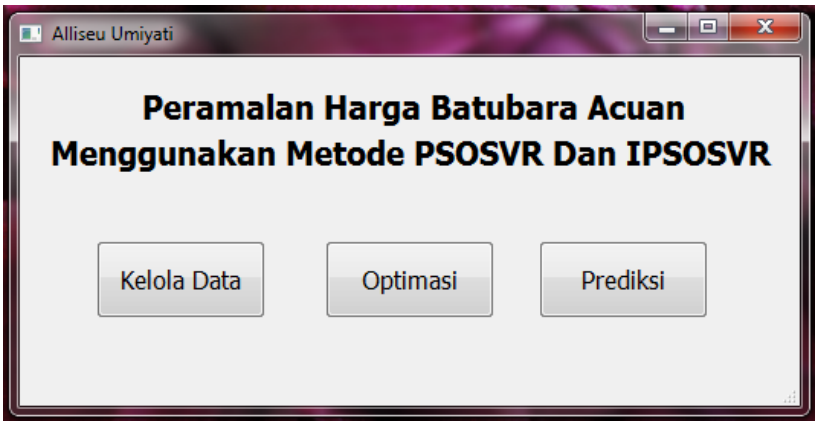
| | | | |
|---|---|---|---|
| | | | dalam proses PSO dan IPSO |
| Hba-prediski | pred_target | Float | Prediksi HBA bulan depan |
| *Error* | mape_error | Float | Nilai error dari hasil prediksi data uji |

### 3.7.3 Perancangan Tampilan Program Aplikasi

Rancangan tampilan program aplikasi PSOSVR dan IPSOSVR meliputi tampilan utama dan menu-menu lainnya adalah sebagai berikut :

1. Menu Utama

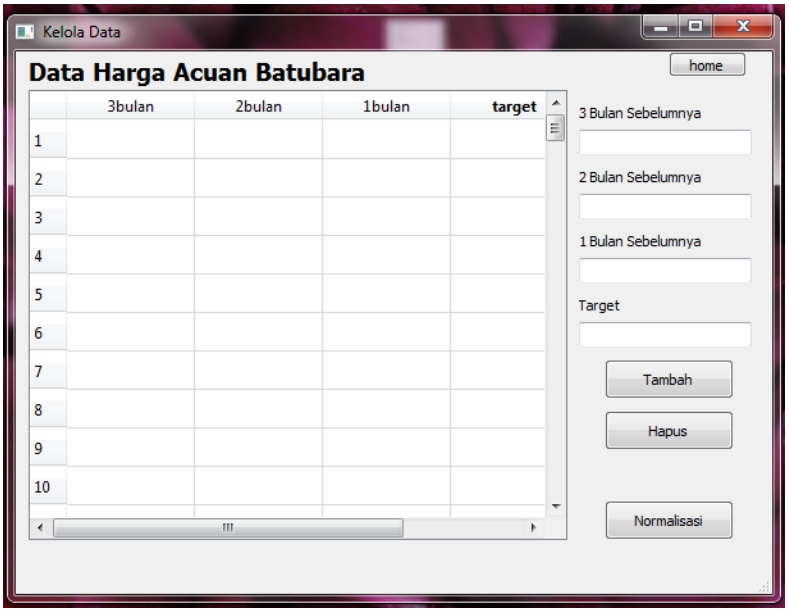
   Berikut tampilan menu utama program aplikasi PSOSVR Dan IPSOSVR seperti tersaji pada gambar 3.3.




**Gambar 3. 3 Menu Utama Program Aplikasi PSOSVR Dan IPSOSVR**

2. Menu Kelola Data

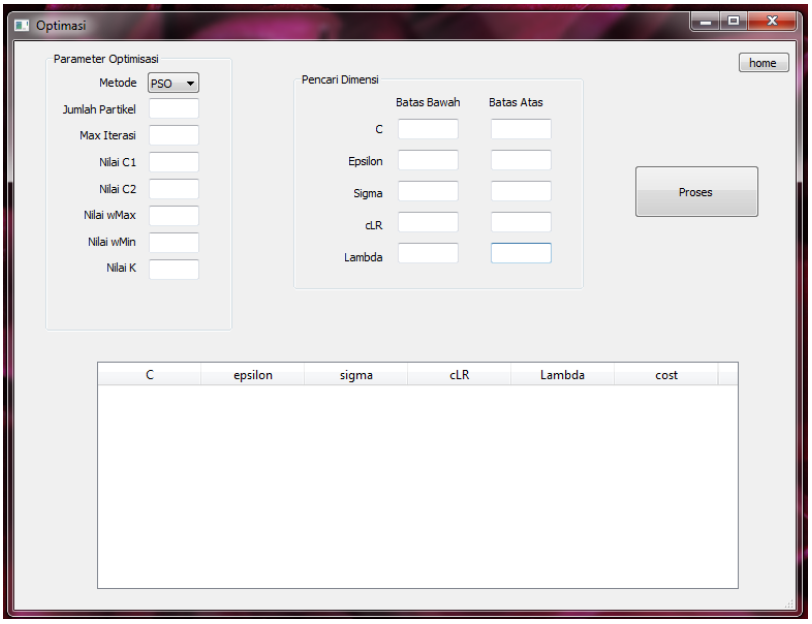
   Berikut tampilan menu kelola data program aplikasi PSOSVR Dan IPSOSVR seperti tersaji pada gambar 3.4.



**Gambar 3. 4 Menu Kelola Data Program Aplikasi PSOSVR Dan IPSOSVR**

3. Menu Optimasi

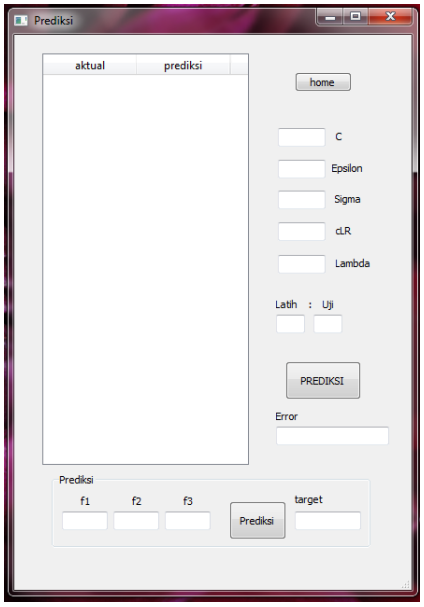
   Berikut tampilan menu optimasi program aplikasi PSOSVR Dan IPSOSVR seperti tersaji pada gambar 3.5.




**Gambar 3. 5 Menu Optimasi Program Aplikasi PSOSVR Dan IPSOSVR**

4. Menu Prediksi

   Berikut tampilan menu prediksi program aplikasi PSOSVR Dan IPSOSVR seperti tersaji pada gambar 3.6.



**Gambar 3. 6 Menu Prediksi Program Aplikasi PSOSVR Dan IPSOSVR**