BAB 2

LANDASAN TEORI

2.1 Landasan Teori

Bagian ini akan membahas penelitian yang relevan dari topik permasalahan yang sedang diteliti. Berikut ini, terdapat lima penelitian yang dijadikan sebagai referensi. Penelitian pertama membahas Perbandingan *Extreme Leaning Machine* dengan *Support Vector Regression* untuk Prediksi Permeabilitas Reservoir. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai *Root Mean Square Error* (RMSE) sebesar 0,4178 dengan kecepatan pembelajaran 3436,625 *seconds* sedangkan algorime ELM mampu menghasilkan nilai RMSE sebesar 0,4189 dengan kecepatan pembelajaran 2,2231 *seconds*. Berdasarkan hasil tersebut algorime ELM memiliki keunggulan dalam proses pembelajaran sedangkan algoritme SVR unggul dalam akurasi tingkat kesalahan (Cheng, Cai, & Pan, 2009).

Penelitian kedua membahas Prediksi Harga Cabai Rawit di Kota Malang Menggunakan Algoritme *Extreme Learning Machine*. Komoditas cabai rawit merupakan salah penyumbang inflasi terbesar di Indonesia. Untuk itu, diperlukan prediksi harga cabai rawit agar pemerintah dapat melakukan tindakan untuk mencengah permasalahan yang mungkin timbul akibat inflasi. Data yang digunakan pada penelitian ini merupakan data harga cabai rawit yang dikumpulkan mulai tanggal 1 Januari 2017 hingga 31 Desember 2018. Dari hasil penelitian yang telah dilakukan, algoritme ELM menghasilkan rata-rata nilai *Mean Absolute Percentage Error* (MAPE) sebesar 2,087% (Ariwanda, Cholissodin, & Tibyani, 2019).

Penelitian ketiga membahas Peramalan Harga Cabai Merah Besar Wilayah Jawa Timur Menggunakan Metode *Extreme Learning Machine*. Cabai merah besar memiliki tingkat konsumsi yang tinggi karena digunakan untuk bumbu dapur dan bahan masakan. Harga cabai merah besar yang tidak menentu mengakibatkan kerugian bagi negara dan masyarakat. Maka dari itu, diperlukan prediksi harga cabai merah besar untuk memperkirakan kenaikan harga di masa yang akan datang. Data yang digunakan pada penelitian ini merupakan data harga cabai merah besar yang dikumpulkan mulai tanggal 18 Juli 2016 hingga 28 Desember 2018. Dari hasil penelitian yang telah dilakukan, algoritme ELM menghasilkan rata-rata nilai MAPE sebesar 3% (Adiatmaja, Setiawan, & Wihandika, 2019).

Penelitian keempat membahas Prediksi Harga Cabai Merah Menggunakan Support Vector Regression. Cabai merupakan komoditas sayuran yang sangat penting bagi masyarakat Indonesia karena merupakan bahan utama dalam berbagai pengolahan makanan. Harga cabai yang tidak menentuk memicu timbulnya inflasi perekonomian. Untuk mengantisiapsi harga cabai dipasaran agara tetap dalam batas wajar diperlukan prediksi harga cabai. Data yang digunakan pada penelitian ini merupakan data harga cabai rawit yang dikumpulkan mulai tanggal 1 Januari 2017 hingga 31 Desember 2019. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai MAPE sebesar 4,07% (Sepri, et al., 2020).

Penelitian kelima membahas *Support Vector Regression* Untuk Peramalan Permintaan Darah: Studi Kasus Unit Transfusi Darah Cabang – PMI Kota Malang. PMI bertanggung jawab untuk memenuhi permintaan darah dari rumah sakit. Pengelola pusat penyimpanan darah memiliki tugas untuk mengelola jumlah pasokan darah. Kurangan atau lebihnya jumlah pasokan darah di tempat

penyimpaan seharusnya tidak terjadi, karena berdampak pada tingginya angka pasien yang meninggal. Untuk mengurangi kesalahan yang terjadi, diperlukan penelitian untuk melakukan prediksi permintaan darah. Dari hasil penelitian yang telah dilakukan, algoritme SVR menghasilkan nilai MAPE sebesar 3,899%. Hasil nilai MAPE yang kurang dari 10% dapat dikategorikan baik untuk hasil prediksi. (Rifqi, Setiawan, & Bactiar, 2018). Berikut merupakan daftar referensi penelitian yang ditunjukan pada Tabel 2.1.

Tabel 2.1 Daftar Landasan Pustaka

No.	Penelitian	Objek	Metode	Hasil
1.	(Cheng, Cai, & Pan, 2009).	Prediksi Permeabilitas Reservoir (Waduk minyak bumi).	*Extreme Learning Machine* dan *Support Vector Machine* dengan menggunakan perhitungan nilai *error Root Mean Square Error* (RMSE).	Algoritme *Support Vector Machine* menghasilkan nilai RMSE sebesar 0,4178 dengan kecepatan pembelajaran sebesar 3436,625s. Algoritme *Extreme Learning Machine* menghasilkan nilai RMSE sebesar 0,4189 dengan kecepatan pembelajaran sebesar 2,2231s.
2.	(Ariwanda, Cholissodin, & Tibyani).	Data harga cabai rawit di Kota Malang tanggal 1 Januari 2017 hingga 31 Desember 2018.	*Extreme Learning Machine* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Extreme Learning Machine* menghasilkan nilai MAPE sebesar 2,087%.
3.	(Adiatmaja, Setiawan, & Wihandika).	Data harga cabai rawit di Kota Malang tanggal 18 Juli 2016 hingga	*Extreme Learning Machine* dengan menggunakan perhitungan nilai *error Mean Absolute*	Algoritme *Extreme Learning Machine* menghasilkan nilai MAPE sebesar 3%.

		28 Desember 2018.	*Percentage Error* (MAPE).	
4.	(Sepri, et al., 2020).	Data harga cabai rawit di Kota Malang tanggal 1 Januari 2017 hingga 31 Desember 2019.	*Support Vector Regression* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Support Vector Regression* menghasilkan nilai MAPE sebesar 4,07%.
5.	(Rifqi, Setiawan, & Bactiar).	Data jumlah pasokan darah.	*Support Vector Regression* dengan menggunakan perhitungan nilai *error Mean Absolute Percentage Error* (MAPE).	Algoritme *Support Vector Regression* menghasilkan nilai MAPE sebesar 3,899%.

2.2 Prediksi

Prediksi merupakan proses memperkirakan sesuatu yang mengambarkan kondisi di masa yang akan datang berdasarkan data pada periode tertentu. Prediksi tidak perlu memberikan jawaban pasti, melainkan berusaha untuk mencari jawaban sedekat mungkin dengan apa yang akan terjadi (Herdianto, 2013). Prediksi diperlukan untuk memperkirakan kondisi yang akan datang sehingga hasil prediksi dapat dijadikan sebagai acuan dalam mengambil suatu keputusan serta perencanaan kedepannya.

Berdasarkan sifatnya, prediksi dibagi menjadi dua jenis, yaitu prediksi kualitatif dan prediksi kuantitatif. Prediksi kualitatif merupakan prediksi untuk data non-numerik sedangkan prediksi kuantitatif merupakan prediksi yang berdasarkan pada data numerik. Berdasarkan jangka waktu, prediksi dibagi menjadi tiga bagian, yaitu prediksi jangka pendek, prediksi jangka menengah, dan prediksi jangka panjang. Prediksi jangka pendek merupakan prediksi untuk rentang waktu kurang dari satu tahun. Kemudian prediksi jangka menengah merupakan prediksi untuk

rentang waktu tiga bulan sampai tiga tahun. Prediksi jangka panjang merupakan prediksi untuk rentang waktu lebih dari tiga tahun.

Karena penelitian ini menggunakan data numerik dan prediksinya hanya dilakukan untuk satu hari kedepan maka penelitian ini termasuk dalam jenis prediksi kuantitatif dan termasuk dalam prediksi jangka pendek.

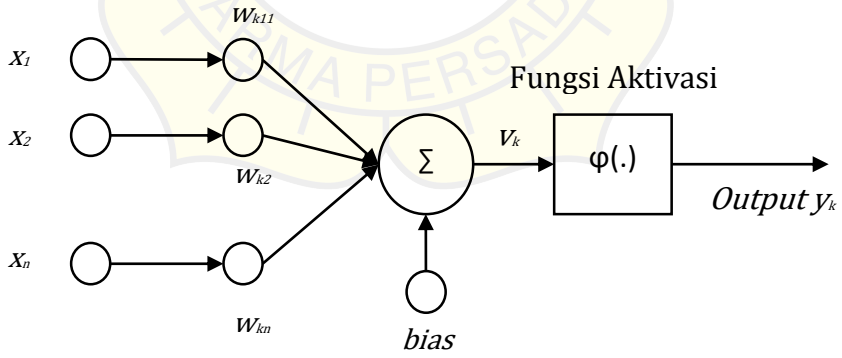
2.3 Cabai Rawit

Tanaman cabai rawit merupakan jenis tanaman dengan berbagai kadungan zat-zat gizi yang cukup lengkap, yaitu lemak, protein, karbohidrat, kalsium, fosfor, besi, vitamin A, B1, B2, C dan senyawa alkaloid seperti *capsaicin*, *flavanoid*, *oleoresin* dan minyak atsiri (Sujitno & Dianawati, 2015). Selain memiliki banyak kandungan zat-zat gizi, cabai rawit juga miliki manfaat untuk memberikan sensasi pedas pada makanan. Selain itu, buah tanaman ini juga berkhasiat untuk menambah nafsu makan (Tjandra, 2011). Cabai rawit merupakan komoditas sayuran yang sangat penting karena menjadi bahan utama yang digunakan masyarakat Indonesia. Rasa pedas dan kaya akan manfaat menjadi alasan utama untuk mengkonsumsi cabai rawit. Mengingat kebutuhan masyarakat Indonesia akan cabai rawit terus meningkat membuat harga cabai rawit mengalami fluktuasi. Kenaikan harga yang fluktuatif menimbulkan kerugian pada beberapa kalangan, yaitu pedangan dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utamanya. Untuk itu, dibutuhkan suatu metode untuk memprediksi harga cabai rawit di masa yang akan datang. Dengan melakukan prediksi harga cabai rawit diharapkan dapat membantu pedagang untuk mengendalikan jumlah permintaan cabai rawit dan pengusaha kuliner yang menggunakan cabai rawit sebagai bahan utama karena hasil

prediksinya dapat digunakan untuk menekan modal pengeluaran sehingga para pengusaha bisa mendapatkan keuntungan yang maksimal.

2.4 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) merupakan suatu teknik yang digunakan untuk membangun sebuah program cerdas dengan menggunakan pendekatan pemodelan yang menstimulasikan cara kerja otak manusia untuk menyelesaikan masalah melalui perhitungan komputer (Fikriya, Irawan, & Soetrisno, 2017). JST bekerja dengan memanfaatkan *neuron* untuk memberikan informasi dari satu *neuron* ke *neuron* lainnya. Bobot (*weight*) digunakan sebagai perantara untuk menyalurkan informasi tersebut. Setiap informasi yang disalurkan kepada *neuron* lainnya memiliki nilai tertentu. Nilai tersebut akan diproses dan disalurkan kembali hingga menghasilkan *output*. Berikut adalah struktur *neuron* pada jaringan saraf tiruan ditunjukan pada Gambar 2.2.



Gambar 2.1 Struktur Jaringan Saraf Tiruan

Sumber: (Fikriya, Irawan, & Soetrisno, 2017)

2.5 Fungsi Aktivasi

Fungsi aktivasi merupakan sebuah perhitungan untuk menentukan *output* berdasarkan *input neuron*. Fungsi aktivasi yang digunakan harus disesuaikan dengan bentuk datanya. Karena data yang digunakan pada penelitian ini merupakan data *time series*, maka fungsi aktivasi yang dapat digunakan adalah fungsi aktivasi *nonlinear*. Fungsi aktivasi ini mampu mengenali pola pengerakan data acak sehingga fungsi aktivasi ini dapat digunakan untuk prediksi data *time series*. Fungsi aktivasi *nonlinear* memiliki beberapa macam pemodelan. Pada penelitian ini fungsi aktivasi yang digunakan adalah fungsi aktivasi *sigmoid biner*. Berikut merupakan perhitungan dari fungsi aktivasi *sigmoid biner* ditunjukan pada Persamaan 2.1 (Srimuang & Intarasothonchun, 2015).

Fungsi *Sigmoid Biner*

$$H = \frac{1}{1+\exp\left(-H_{init}\right)} \tag{2.1}$$

Keterangan:

H = *Output* pada fungsi aktivasi

exp = Basis bilangan logaritma atau biasa disebut dengan bilangan *euler*, bilangan ini memiliki nilai pembulatan sebesar 2,71828183

H_{init} = *Input* pada fungsi aktivasi

2.6 Normalisasi Data

Normalisasi data merupakan suatu teknik untuk mentransformasikan data kedalam satuan yang sama. Normalisasi dilakukan untuk menghilangkan *outlier* pada data yang memiliki nilai yang jauh berbeda dengan nilai lainnya. Normalisasi juga digunakan untuk mempersiapkan data sebelum dihitung menggunakan pendekatan pemodelan. Pada dasarnya, normalisasi memiliki berbagai macam perhitungan. *Min-Max Normalization* digunakan peneliti karena memilki kesesuaian pada data yang digunakan. *Min-Max Normalization* bekerja dengan melakukan transformasi linier terhadap data asli. Untuk lebih lanjut, perhitungan *Min-Max Normalization* ditunjukan pada Persamaan 2.2 (Mustaffa & Yusof, 2011).

$$X_n = \frac{(X_0 - X_{min})}{(X_{max} - X_{min})} \tag{2.2}$$

Keterangan:

X_n = Nilai hasil normalisasi

X_0 = Nilai yang akan dinormalisasi

X_{max} = Nilai maksimum dari keseluruhan data

X_{min} = Nilai minimum dari keseluruhan data

2.7 Denormalisasi Data

Denormalisasi data merupakan proses pengembalian data normalisasi menjadi data yang sebenarnya. Denormalisasi data dilakukan ketika suatu pendekatan pemodelan data telah memberikan hasil akhirnya. Perhitungan denormalisasi disesuaikan dengan perhitungan normalisasi yang digunakan. Berikut adalah

perhitungan denormalisasi data ditunjukan pada Persamaan 2.3 (Mustaffa & Yusof, 2011).

$$X_0 = X_n.(X_{max} - X_{min}) + X_{min} \quad (2.3)$$

Keterangan:

X_0 = Nilai hasil denormalisasi

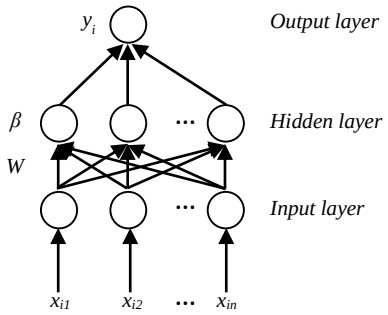
X_n = Nilai yang akan didenormalisasi

X_{max} = Nilai maksimum dari keseluruhan data

X_{min} = Nilai minimum dari keseluruhan data

2.8 *Extreme Learning Machine* (ELM)

Algoritme *Extreme Learning Machine* (ELM) pertama kali diperkenalkan oleh Guang-Bin Huang, dkk pada tahun 2006. Algoritme ELM merupakan bagian dari JST yang dibuat dengan tujuan untuk mengatasi kelemahan dari algoritme JST lainnya terutama pada proses kecepatan pembelajaran (Huang et al, 2004). Algoritme ini sendiri memiliki kemampuan pembelajaran ribuan kali lebih cepat daripada algorime lainnya dan mampu menghasilkan generalisasi yang sangat baik (Wang, et al., 2008). Arsitektur yang digunakan pada algoritme ELM merupakan arsitektur *Single Layer Feedforward Network* (SLFN). Arsitektur SLFN tidak memiliki iterasi didalamnya, karena jaringan *node* yang terhubung pada setiap tidak membentuk *directed cycle/graph*. Arsitektur SLFN sendiri memiliki tiga layer, yaitu input layer, hidden layer dan output layer. Untuk mengetahui lebih lanjut, arsitektur SLFN pada algoritme ELM dapat dilihat pada Gambar 2.3.



Gambar 2.2 Arsitektur Algoritme *Extreme Learning Machine*

Sumber: (Abadi, Musyafa & Soeprijanto, 2014)

Algoritme ELM mempunyai dua proses, yaitu proses pembelajaran atau proses *training* dan proses pengujian atau proses *testing*, kedua proses tersebut saling berkaitan satu sama lain. Untuk lebih lanjut, tahapan yang dilalui pada algoritme ELM dapat dijelaskan sebagai berikut (Fachrony, Cholissodin & Santoso, 2018).

2.8.1 Proses *Training*

Proses *training* pada algoritme ELM merupakan proses memberikan pembelajaran pada sistem untuk mengenali pola data. Berikut adalah proses training pada algoritme ELM dengan menggunakan *bias*.

1. Inisialisasi bobot W_{jk} dan bias *b* secara acak pada rentang tertentu.

   Inisialisasi nilai bobot berkisar antara [0, 1], sedangkan nilai *bias* berkisar antara [-1,1]. Besar ukuran matriks bobot W_{jk} ditentukan pada jumlah fitur (k) dan *hidden neuron* (j). Kemudian, besar ukuran pada matriks bias *b* hanya disesuaikan dengan jumlah *hidden neuron* yang digunakan.

2. Menghitung nilai H_{init}

Proses perhitungan nilai H_{init} digunakan untuk menghitung nilai *hidden layer*. Berikut adalah rumus perhitungan nilai H_{init} ditunjukan pada Persamaan 2.4.

$$H_{init} = X.W^T + bias \quad (2.4)$$

Keterangan:

H_{init} = Matriks H_{init}

x = Matriks data *training*

W^T = Matriks *transpose* bobot

3. Menghitung *hidden layer* (H)

Pada perhitungan nilai *hidden layer*, jenis fungsi aktivasi yang digunakan adalah fungsi *sigmoid biner*. Berikut merupakan proses perhitungan *hidden layer* yang ditunjukan pada Persamaan 2.5.

$$H = \frac{1}{1+\exp(-H_{init})} \quad (2.5)$$

Keterangan:

H = Matriks *Output hidden layer*

exp = Bilangan *euler* dengan nilai pembulatan sebesar 2,71828183

4. Menghitung H *dagger* ($H^\dagger$)

Pada perhitungan H *dagger* atau *Moore-Penrose pseudo inverse* membutuhkan proses perhitungan *inverse* matriks. Berikut merupakan proses perhitungan H *dagger* ditunjukan pada Persamaan 2.6.

$$H^\dagger = (H^T.H)^{-1}.H^T \quad (2.6)$$

Keterangan:

$H^{\dagger}$ = Matriks H *dagger*

H^{T} = Matriks H *transpose*

$(H^{T}.H)^{-1}$ = Matriks *inverse* dari perkalian H *transpose* dengan H

5. Menghitung nilai *output weight* ($\hat{\beta}$)

Proses perhitungan *output weight* merupakan tahap terakhir pada proses *training*. Nilai yang dihasilkan pada perhitungan ini akan diteruskan pada proses *testing* untuk mendapatkan hasil prediksi. Proses perhitungan *output weight* ditunjukan pada Persamaan 2.7.

$$\hat{\beta} = H^{\dagger}.Y \tag{2.7}$$

Keterangan:

$\hat{\beta}$ = Matriks *ouput weight*

$H^{\dagger}$ = Matriks H *dagger*

Y = Matriks data target

2.8.2 Proses *Testing*

Proses *testing* pada algoritme ELM merupakan tahap yang dilakukan untuk menggeneralisasi data baru. Proses *testing* pada algoritme ELM dapat diuraikan sebagai berikut.

1. Menghitung H_{init}

Proses perhitungan H_{init} disesuaikan pada Persamaan 2.4. Data yang digunakan untuk menghitung H_{init} adalah data *testing*.

2. Menghitung *hidden layer* (H)

Matriks H_{init} yang telah dihitung akan digunakan pada proses perhitungan *hidden layer*. Proses perhitungan *hidden layer* pada proses *testing* disesuaikan dengan Persamaan 2.5.

3. Menghitung *Output Layer* ($\hat{Y}$)

Proses perhitungan *output layer* merupakan proses terakhir pada algortime ELM. Proses perhitungan ini membutuhkan matriks *hidden layer* dan matriks *output weight* yang dihasilkan dari proses *training*. Hasil perhitungan *output layer* perlu didenormalisasi untuk mengembalikan bentuk data yang sebenarnya sehingga hasil tersebut dapat dilakukan pengujian untuk mengukur tingkat kesalahannya. Proses untuk menghitung *output layer* ditunjukan pada Persamaan 2.8.

$$\hat{Y} = H.\hat{\beta} \tag{2.8}$$

Keterangan:

$\hat{Y}$ = Matriks *output layer*

H = Matriks *hidden layer*

$\hat{\beta}$ = Matriks *ouput weight*

2.9 *Support Vector Regression* (SVR)

Support Vector Regression (SVR) merupakan model pengembangan dari *Support Vector Machine* (SVM) yang digunakan untuk menyelesaikan kasus prediksi pada data *time series*. Algoritme SVR bekerja dengan mencari garis pemisah atau *hyperplane* terbaik dengan cara mengukur jarak antara garis pemisah

dengan data terdekat. Tujuan algoritme SVR dibuat adalah untuk mengatasi permasalahan *overfitting* yang terjadi pada model pendekatan regresi lainnya. Algoritme ini sendiri memiliki kelebihan dalam memanfaatkan fungsi kernel untuk memetakan vektor input ke ruang fitur berdimensi tinggi sehingga algoritme SVR dapat mengatasi *overfitting* dan mampu menghasilkan performa generalisasi yang lebih baik (Maharesi, 2013). Untuk lebih lanjut, alur tahapan yang dilalui algoritme SVR dapat diuraikan sebagai berikut (Vijayakumar & Wu, 1999).

1. Inisialisasi parameter *sigma* (σ), *lambda* (λ), *epsilon* (ε), *coefisien Learning Rate* (cLR), kompleksitas (C) dan jumlah iterasi maksimum.

2. Inisialisasi nilai *Lagrange Multiplier* α_i = 0 dan α_i^* = 0 kemudian hitung matriks *Hessian* menggunakan fungsi kernel *Gaussian RBF* pada Persamaan 2.9.

$$R_{ij} = (K(x_i, x_j) + \lambda^2) \; untuk \; i, j = 1, \dots, n \tag{2.9}$$

Keterangan:

R_{ij} = Matriks *Hessian*

x_i = Data ke-i

x_j = Data ke-j

$K(x_i, x_j)$ = Fungsi kernel *Gaussian RBF*

λ = Variable skalar

3. Masuk pada iterasi pertama sampai batas iterasi yang ditentukan, untuk setiap data ke-*i* sampai ke-*n* lakukan perhitungan sebagai berikut.

   a. Nilai *error* ke-*i*.

$$E_i = y_i - \sum_{j=1}^{n} (\alpha_i^* - \alpha_i) R_{ij} \tag{2.10}$$

b. Hitung perubahan nilai *Lagrange Multiplier*

$$\delta\alpha_i^* = min\{\max[\Upsilon(E_i - \varepsilon), -\delta\alpha_i^*], C - \delta\alpha_i^*\} \quad (2.11)$$

$$\delta\alpha_i = min\{\max[\Upsilon(E_i - \varepsilon), -\delta\alpha_i], C - \delta\alpha_i\} \quad (2.12)$$

c. Menambahkan nilai *Lagrange Multiplier* dengan hasil perubahannya

$$\delta\alpha_i^* = \alpha_i^* + \delta\alpha_i^* \quad (2.13)$$

$$\delta\alpha_i = \alpha_i + \delta\alpha_i \quad (2.14)$$

Keterangan:

E_i = Nilai *Error* ke-*i*

y_i = Nilai data target

$\alpha i*$ = *Lagrange Multiplier*

αi = *Lagrange Multiplier*

Rij = Matriks *Hessian*

$\delta\alpha_i^*$ = Perubahan nilai α_i^*

$\delta\alpha_i$ = Perubahan nilai α_i

Υ = Nilai *learning rate* atau *gamma*

ε = Nilai *epsilon*

C = Nilai Kompleksitas

4. Kembali mengulang langkah ketiga sampai batas iterasi maksimum yang telah ditentukan atau sampai dalam kondisi max ($|\delta\alpha_i^*|$) < ε dan max ($|\delta\alpha_i|$) < ε.

5. Menghitung fungsi regresi

Proses perhitungan fungsi regresi merupakan tahap terakhir pada algoritme SVR. Perhitungan ini digunakan untuk menghasilkan nilai prediksi. Proses perhitungan fungsi regresi ditunjukan pada Persamaan 2.15.

$$f(x) = \sum_{j=1}^{n}(\alpha_i^* - \alpha_i)(K(x_i, x_j) + \lambda^2) \quad (2.15)$$

Keterangan:

$\delta\alpha_i^*$ = *Lagrange Multiplier*

$\delta\alpha_i$ = *Lagrange Multiplier*

x_i = Data ke-i

x_j = Data ke-j

$K(x_i, x_j)$ = Fungsi kernel *Gaussian RBF*

λ = Variable skalar

2.10 *Mean Absolute Percentage Error* (MAPE)

MAPE merupakan suatu model perhitungan yang bertujuan untuk menentukan tingkat kesalahan berdasarkan selisih antara nilai hasil prediksi dengan data target. Perhitungan MAPE ditunjukan pada Persamaan 2.16 (Andini & Auristandi, 2016).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \; x \; 100\,\% \quad (2.16)$$

Keterangan:

$\hat{y}_i$ = Nilai hasil prediksi

y_i = Nilai data target

n = Jumlah data *testing*

Penilaian kinerja sebuah model pembelajaran pada perhitungan nilai MAPE terbagi atas empat kategori. Berikut adalah kriteria penilaian MAPE ditunjukan pada Tabel 2.2 (Rahmadiani & Anggraeni, 2012).

Tabel 2.2 Kriteria Nilai MAPE

Nilai MAPE	Status
< 10%	Sangat baik
10 – 20%	Baik
20 – 50%	Cukup
> 50%	Buruk