Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 12, Desember 2018, hlm. 6418-6428 http://j-ptiik.ub.ac.id

Penerapan Algoritme *Particle Swarm Optimization-Learning Vector Quantization* (PSO-LVQ) Pada Klasifikasi Data Iris

Ilham Romadhona[1], Imam Cholissodin[2], Marji[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]romadona.ilham2@gmail.com, [2]imamcs@ub.ac.id, [3]marji@ub.ac.id

Abstrak

Bunga Iris saat ini telah mudah dijumpai diberbagai penjuru dunia dengan spesies yang bermacam-macam. Dalam bahasa Yunani Iris merupakan sosok dari dewi pelangi karena spesies Iris sendiri telah mencapai 260 hingga 300 macam spesies dengan warna bunga yang berwarna-warni dan mencolok. Karena banyaknya spesies Iris tersebut, maka dibutuhkan klasifikasi dalam membedakan spesies pada bunga Iris. Untuk menyelesaikan permasalahan tersebut, digunakan *algoritme Learning Vector Quantization* (LVQ) yang nantinya akan dioptimasi menggunakan algoritme *Particle Swarm Optimization* (PSO) dengan melakukan klasifikasi pada spesies Iris Sentosa, Iris Virginica dan VersiColor yang mana spesies tersebut telah didata sebelumnya pada dataset Iris. Hasil dari penelitian ini selanjutnya dibandingkan dengan klasifikasi menggunakan algoritme LVQ. Didapatkan rata-rata akurasi menggunakan algoritme PSO-LVQ sebesar 93,334%, sedangkan rata-rata akurasi menggunakan algoritme LVQ sebesar 84,268%. Selisih rata-rata akurasi yang didapatkan sebesar 9,066% yang menandakan algoritme PSO-LVQ memberikan peningkatan hasil yang cukup baik dibandingkan algoritme LVQ.

Kata kunci: *Dataset Iris, Optimasi, Particle Swarm Optimization, Learning Vector Quantization*

Abstract

Currently Iris flowers are easily found in around the world with various species. In Greek Iris mean the goddess of the rainbow because Iris species has reached 260 to 300 various species with colorful and light flowers. Because of the large number of Iris species, it is necessary to classify the Iris species. To solve the problem, used the Learning Vector Quantization (LVQ) algorithm which will be optimization using the Particle Swarm Optimization (PSO) algorithm was used to classify species into Sentosa Iris, Virginica Iris and Versicolor Iris category where the species previously recorded on Iris dataset. Then the result of this study was compared with the classification using LVQ algorithm. The average accuracy obtained with PSO-LVQ algorithm is 93.334%, whereas the average accuracy with LVQ algorithm is 84.268%. The differece in accuracy is 9.066% it is mean PSO-LVQ algorithm give more a good provides result than LVQ algorithm.

Keywords: *Iris Dataset, Optimization, Particle Swarm Optimization, Learning Vector Quantization*

1. PENDAHULUAN

Iris merupakan spesies tumbuhan di zaman ini telah banyak kita jumpai di berbagai penjuru dunia. Pada awalnya Iris hanya tumbuh di Eropa, Asia Barat, Afrika di bagian utara, dan daerah medeterania. Iris berasal dari salah satu bahasa Yunani untuk dewi pelangi, hal tersebut dapat dibuktikan dengan banyaknya spesies yang mencapai 260 hingga 300 macam spesies dengan warna bunga yang mencolok (Kamenetsky dan Hiroshi, 2013). Dari banyaknya spesies tersebut terdapat beberapa spesies yang memiliki kemiripan seperti Iris Sentosa, Iris VersiColor, dan Iris Virginica. Dari beberapa spesies yang memiliki kemiripan tersebut perlu dilakukan klasifikasi untuk membedakannya.

Klasifikasi merupakan proses dalam mencari sebuah informasi dari kumpulan data yang bertujuan memprediksi label atau kelas dari objek dalam suatu objek (Han dan Kamber, 2006). Metode-metode pengklasifikasian banyak sekali, salah satunya *Neural Network*. *Neural Network* (*NN*) atau umumnya Jaringan

Syaraf Tiruan (JST) merupakan model komputasi yang memiliki struktur dan fungsional seperti jaringan syaraf manusia. *Neural Network* memiliki proses distribusi secara paralel yang berkecenderungan untuk menyimpan sebuah pengetahuan dari proses pembelajaran (Haykin, 1994). Secara umum *Neural Network* terdiri dari beberapa *input layer, beberapa hidden layer,* dan *output layer*. *Output layer* merupakan hasil permrosesan dari *Neural Network* yang didapat dari proses pembelajaran dari bobot awal yang telah diberikan pada *input layer*.

Banyak sekali metode-metode *Neural Network* yang dapat kita gunakan seperti *Hebbian Learning, Backpropagation*, dan *Learning Vector Quatization*. Salah satu algoritme *Neural Network adalah Learning Vector Quantization* (LVQ). *Learning Vector Quantization* merupakan metode pengklasifikasian pola yang mana *output* setiap unit mewakili kategori tertentu (Kohonen, 1990). *Learnig Vector Quantization* merupakan salah satu dari metode *Neural Network* yang ditemukan oleh T. Kohonen. Kelebihan LVQ salah satunya ialah kemampuannya dalam melakukan pelatihan pada lapisan yang kompetitif.

Penelitian yang terkait dengan penggunaan *Learning Vector Quantization* telah banyak dilakukan. Salah satu penelitian yang telah dilakukan oleh Meri Azmi (2014) didapatkan bahwa kedua metode tersebut dapat menunjukkan hasil perhitungan yang baik dimana akurasi yang didapatkan dengan metode LVQ adalah 86,66%. Meskipun metode LVQ mampu menyelesaikan permasalahan tersebut dengan hasil yang baik, namun hasil akurasi LVQ masih belum maksimal. Kendala dalam metode LVQ ialah penentuan bobot yang digunakan pada awal perhitungan yang dapat mempengaruhi hasil akurasinya. Sehingga diperlukan metode lain yang dapat digunakan untuk meningkatkan akurasi, salah satunya dalam penelitian yang dilakukan oleh Raissa Arniantya (2017). Dengan menggabungkan algoritme evolusi dan LVQ tersebut diperoleh hasil yang lebih baik dimana akurasi yang diperoleh mencapai 92%. Terdapat beberapa metode dalam algoritme evolusi, diantaranya *Genetic Algorithm* dan *Particle Swarm Optimization.* Selain itu dalam penelitian yang dilakukan Haldar dan Mishra (2016), dibahas perihal penggunaan *Particle Swarm Optimization* (PSO) sebagai metode optimasi pada LVQ. Pada penelitian tersebut didapatkan hasil yang lebih baik pada LVQ yang dioptimasi dengan PSO dengan akurasi 92% dibandingkan dengan LVQ tanpa optimasi PSO dengan akurasi 90%.

Dari uraian diatas, maka penulis membuat penelitian dengan mengoptimasi algoritme *Learning Vector Quantization (LVQ)* dengan algoritme *Particle Swarm Optimization (PSO)* dimana algoritme *Particle Swarm Optimization (PSO)* memiliki kelebihan dari segi efisiensi dibanding dengan teknik optimasi lain. Sehingga penulis mengangkat judul "*Penerapan Algoritme Particle Swarm Optimization-Learning Vector Quantization (PSO-LVQ) Pada Klasifikasi Data Iris*" dengan menggunakan dataset iris yang diambil dari UCI Machine Learning yang dijadikan sebagai data latih dalam penelitian ini.

2. LANDASAN KEPUSTAKAAN

2.1. Dataset Iris

Data penelitian ini berasal dari dataset yang didapatkan pada *UCI Machine Learning.* Dataset ini berisikan data sebanyak 150 dengan jumlah kelas sebanyak 3 yaitu *Iris VersiColor, Iris Sentosa* dan *Iris Virginica.* Selain itu terdapat 4 atribut pada setiap data yaitu *Sepal Length, Petal Length, Sepal Width* dan *Petal Width.* Sepal adalah daun kelopak dari bunga iris, sedangkan petal adalah daun mahkota dari bunga iris.

2.2 Klasifikasi

Klasifikasi dibagi dalam 2 jenis yaitu *Supervised Learning* dan *Unsupervised Learning. Supervised Learning*, yaitu belajar di mana satu set data latih akan diidentifikasi sesuai dengan kelompoknya. *Unsupervised Learning* atau yang dikenal sebagai pengelompokan, melibatkan pengelompokkan data ke dalam kategori berdasarkan beberapa ukuran kemiripan atau jarak yang melekat (Alpaydin, 2010).

2.3 *Learning Vector Quantization* (LVQ)

Learning Vector Quantization (LVQ) merupakan metode pelatihan untuk melakukan proses pembelajaran pada lapisan kompetitif yang terawasi (*supervised learning*) yang arsitekturnya berlayer tunggal (*single layer*). Kelas-kelas yang diperoleh sebagai sebuah hasil

dari lapisan kompetitif ini bergantung pada jarak antar vektor-vektor yang menjadi input. Jika dua vektor input tersebut memiliki jarak yang mendekati sama, maka lapisan kompetitif akan berada pada kelas yang sama. LVQ merupakan metode klasifikasi pola yang mewakili kategori atau kelas tertentu (beberapa unit dari keluaran seharusnya digunakan untuk setiap kelas). Keunggulan metode ini adalah kemampuannya dalam memberikan pelatihan pada lapisan-lapisan kompetitif sehingga dapat mengklasifikasikan vektor input yang diberikan (Nugroho, 2011). LVQ memiliki beberapa variasi yang dapat digunakan yaitu LVQ1, LVQ2, LVQ2.1 dan LVQ3 (Fausett, 1994). Pada penelitian ini digunakan algoritme LVQ2 dalam proses *learning* nantinya.

Langkah-langkah dari algoritme LVQ terdiri atas (Wuryandari, 2012):

1. Inisialisasi bobot awal

   Inisialisasi untuk bobot awal pada algoritme *Learning Vector Quantization* diawali dengan memilih data secara acak dari data yang ada.

2. Proses pelatihan

   a. Lakukan perulangan sebanyak *epoch* yang telah ditentukan dan lakukan *update* pada *epoch* apabila telah dilakukan pelatihan terhadap data latih yang disediakan. Proses *update epoch* dilakukan terus menerus hingga mencapai batas maksimal *epoch* yang telah ditentukan.

   b. Lakukan proses pelatihan terhadap seluruh data latih yang ada. Proses ini dilakukan di dalam proses perulangan *epoch.* Proses pelatihan bobot menggunakan Persamaan 1 dengan data latih yang disediakan.

   $$\boldsymbol{D_i} = \sqrt{\sum_{j=1}^{n}\left(\boldsymbol{X_j} - \boldsymbol{W_{ij}}\right)^2} \qquad (1)$$

   Keterangan:
   D_i = Jarak antara bobot dan data pada kelas-*i*.
   n = Jumlah seluruh parameter yang ada.
   X_j = Parameter data latih ke-*j*.
   W_{ij} = Bobot pada kelas ke-*i* dan parameter ke-*j*.

   c. Cari jarak terkecil dari proses perhitungan jarak yang telah dilakukan sehingga didapatkan jarak terkecil pertama dan kedua dengan Persamaan 2 dan Persamaan 3.

   $\boldsymbol{D_c}$ = jarak terkecil pertama, dimana:

   $$\boldsymbol{c} = \min_{i}(\boldsymbol{D_i}) \qquad (2)$$

   $\boldsymbol{D_r} = \min_{i}(\boldsymbol{D_i})$, dimana

   $$\boldsymbol{i \neq c} \qquad (3)$$

   Keterangan:
   D_c = Jarak terkecil pertama.
   D_r = Jarak terkecil kedua.
   $\min_i(D_i)$ =Nilai terkecil pada kelas ke-*i*.

   d. Lakukan *update* terhadap bobot yang sesuai jarak terkecil pertama dan kedua dengan ketentuan:

   - Kelas dari data latih ke-*i* sama dengan kelas bobot dari perhitungan jarak terkecil kedua (D_r).
   - Jarak terkecil pertama dan kedua memenuhi persyaratan *window* pada Persamaan 4.

   $$\frac{\boldsymbol{D_c}}{\boldsymbol{D_r}} > \mathbf{1} - \boldsymbol{\varepsilon} \quad \text{dan} \quad \frac{\boldsymbol{D_r}}{\boldsymbol{D_c}} < \mathbf{1} + \boldsymbol{\varepsilon} \qquad (4)$$

   Keterangan:
   D_c = Jarak terkecil pertama.
   D_r = Jarak terkecil kedua.
   ε = nilai *epsilon*.

   Apabila telah memenuhi persyaratan dari *window* maka bobot akan dilakukan proses *update* secara simultan dengan Persamaan 5 dan Persamaan 6.

   $$\mathbf{W_c(t+1)} = \mathbf{W_c(t)} - \boldsymbol{\alpha}\,\mathbf{(t)}[\mathbf{X_i(t)} - \mathbf{W_c(t)} \qquad (5)$$

   $$\mathbf{W_r(t+1)} = \mathbf{W_r(t)} + \boldsymbol{\alpha}\,\mathbf{(t)}[\mathbf{X_i(t)} - \mathbf{W_r(t)}] \qquad (6)$$

   Keterangan:
   $W_c(t+1)$ = Bobot dari jarak terkecil pertama baru.
   $W_r(t+1)$ = Bobot dari jarak terkecil kedua baru.
   $W_c(t)$ = Bobot dari jarak terkecil pertama saat ini.
   $W_r(t)$ = Bobot dari jarak terkecil kedua saat ini.
   $\alpha(t)$ = Nilai alpha saat ini.
   $X_i(t)$ = Bobot data latih pada kelas ke-*i* saat ini.

   e. Apabila tidak memenuhi persyaratan yang ada maka dilakuan *update* pada bobot dengan jarak terkecil pertama dengan persamaan berikut:

   - Bila jarak terkecil pertama (D_c) memiliki kelas yang sama

dengan data latih maka *update* dengan Persamaan 7.

$$\mathbf{W_c(t+1) = W_c(t) + \alpha\,(t)[X_i(t) - W_c(t)]} \quad (7)$$

Keterangan:

$W_c(t+1)$ = Bobot dari jarak terkecil pertama baru.

$W_c(t)$ = Bobot dari jarak terkecil pertama saat ini.

$\alpha(t)$ = Nilai alpha saat ini.

$X_i(t)$ = Bobot data latih pada kelas ke-*i* saat ini.

- Apabila jarak terkecil pertama (D_c) tidak memiliki kelas yang sama dengan data latih maka bobot akan dilakukan proses *update* dengan Persamaan 5.

f. Apabila telah melakukan pelatihan terhadap seluruh data latih, maka selanjutnya dilakukan proses *update* pada nilai *alpha* dengan persamaan 8.

$$\boldsymbol{\alpha(t+1) = \alpha(init) * [1 - (i/m)]} \quad (8)$$

Keterangan:

$\alpha(t+1)$ = nilai *alpha* baru.

$\alpha(init)$ = nilai *alpha* awal.

i = iterasi saat ini.

m = iterasi maksimum.

2.4 *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) merupakan metode optimasi yang diperkenalkan Kennedy dan Eberhart tahun 1995 berdasarkan penelitian dengan melihat perilaku kawanan burung. Setiap partikel pada *Particle Swarm Optimization* memiliki kecepatan partikel yang dinamis sesuai dengan perilaku historis mereka. Maka dari itu, partikel berkecenderungan untuk bergerak menuju daerah pencarian yang lebih baik selama proses pencarian. Dalam algoritme PSO terdapat beberapa proses sebagai berikut:

1. Inisialisasi

a. Tentukan batas maksimum dan minimum dari partikel dengan melihat nilai maksimum dan minimum pada data latih.

b. Selanjutnya tentukan batas maksimum dan minimun dari kecepatan.

c. Inisialisasi kecepatan awal

Pada iterasi ke-0, dapat dipastikan bahwa nilai kecepatan awal semua partikel adalah 0.

$$\boldsymbol{V_{ij} = 0} \quad (9)$$

Keterangan:

V_{ij} = Kecepatan pada data ke-*i* dan parameter ke-*j*.

d. Inisialisasi posisi awal partikel

Pada iterasi ke-0, posisi awal partikel dibangkitkan dengan persamaan:

$$\boldsymbol{x_{ij}(0) = T_{j,min} + rand\,[0,1]\,.\,(T_{j,max} - T_{j,min})} \quad (10)$$

Keterangan:

x_{ij} = Partikel ke-*i* dan parameter ke-*j*.

$T_{j,max}$ = Batas maksimum partikel pada parameter ke-*j*.

$T_{j,min}$ = Batas minimum partikel pada parameter ke-*j*.

e. Inisialisasi *Pbest* dan *Gbest*

Ketika iterasi ke-0, nilai *Pbest* akan disamakan dengan nilai posisi awal dari partikel. Sedangkan nilai *Gbest* dipilih salah satu *Pbest* dengan *fitness* tertinggi.

$$\boldsymbol{Pbest_{ij}(0) = x_{ij}(0)} \quad (11)$$

$$\boldsymbol{Gbest(0) = Pbest_k(0)} \quad (12)$$

Keterangan:

$Pbest_{ij}(0)$ = Nilai partikel terbaik di iterasi 0 pada *Pbest* ke-*i* dan parameter ke-*j*.

x_{ij} = Partikel ke-*i* dan parameter ke-*j*.

$Gbest(0)$ = Nilai partikel terbaik keseluruhan di iterasi 0.

k = Posisi partikel dengan nilai *fitness* terbaik.

2. *Update* kecepatan

Untuk melakukan *update* kecepatan, digunakan rumus berikut:

$$v_{i,j}^{(t+1)} = w.v_{i,j}^{t} + c_1.r_1(Pbest_{i,j}^{t} - x_{i,j}^{t}) + c_2.r_2(Gbest_j^{t} - x_{i,j}^{t})$$

$$v_{ij}^{t+1} = \begin{cases} v_{max}\,, & v_{ij}^{t+1} \geq v_{max} \\ v_{min}\,, & v_{ij}^{t+1} \leq v_{min} \\ v_{ij}^{t+1}\,, & v_{min} \geq v_{ij}^{t+1} \geq v_{max} \end{cases} \quad (13)$$

Keterangan:

$Pbest_{ij}(t)$ = Nilai *Pbest* di iterasi ke-*t* pada *Pbest* ke-*i* dan parameter ke-*j*.

x_{ij}^{t} = Partikel ke-*i* dan parameter ke-*j* pada iterasi ke-*t*.

$Gbest_j^{t}(0)$ = Nilai parameter Gbest ke-*j* di iterasi ke-*t*.

w = Bobot *inersi*.

v_{ij}^{t} = Kecepatan ke-*i* dan parameter ke-*j* pada iterasi ke-*t*.

c_1 dan c_2 = Nilai konstanta.
r_1 dan r_2 = Nilai acak dari distribusi *uniform* [0,1].
v_{min} = Batas minimum kecepatan.
v_{max} = Batas maksimum kecepatan.

3. *Update* posisi partikel dan menghitung *fitness*
   Untuk proses *update* posisi, akan digunakan rumus berikut:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}$$

$$x_{ij}^{t+1} = \begin{cases} T_{j,max}, & x_{ij}^{t+1} \geq T_{j,max} \\ T_{j,min}, & x_{ij}^{t+1} \leq T_{j,min} \\ x_{ij}^{t+1}, & T_{j,min} \geq x_{ij}^{t+1} \geq T_{j,max} \end{cases} \quad (14)$$

   Keterangan:
   x_{ij}^{t} = Partikel ke-*i* dan parameter ke-*j* pada iterasi ke-*t*.
   x_{ij}^{t+1} = Partikel ke-*i* dan parameter ke-*j* pada iterasi ke-*t+1*.
   v_{ij}^{t+1} = Kecepatan ke-*i* dan parameter ke-*j* pada iterasi ke-*t+1*.
   $T_{j,max}$ = Batas maksimum partikel pada parameter ke-*j*.
   $T_{j,min}$ = Batas minimum partikel pada parameter ke-*j*.

4. *Update* nilai Pbest dan nilai Gbest
   Akan dilakukan perbandingan antara *fitness* pBest pada iterasi sebelumnya dengan hasil fitness dari *update* posisi. Fitness yang lebih tinggi akan menjadi pBest yang baru. pBest terbaru yang memiliki nilai fitness tertinggi akan menjadi gBest yang baru.

$$Pbest_i^{t+1} = \begin{cases} Pbest_i^{t}, fitness(x_i^{t+1}) \leq fitness(Pbest_i^{t}) \\ x_i^{t+1}, fitness(x_i^{t+1}) > fitness(Pbest_i^{t}) \end{cases} \quad (15)$$

$$Gbest^{t+1} = \begin{cases} Gbest^{t}, argmax\{fitness(Pbest_i^{t+1})\} \\ \quad \leq fitness(Gbest^{t}) \\ Pbest_i^{t+1}, argmax\{fitness(Pbest_i^{t+1})\} \\ \quad > fitness(Gbest^{t}) \end{cases} \quad (16)$$

Keterangan:
$Pbest_i^{t}$ = Nilai *fitness Pbest* di iterasi ke-*t* pada *Pbest* ke-*i*.
$Pbest_i^{t+1}$ = Nilai *fitness Pbest* di iterasi ke-*t+1* pada *Pbest* ke-*i*.
x_i^{t+1} = Nilai *fitness* partikel ke-*i* iterasi ke-*t+1*.
$Gbest_i^{t}$ = Nilai *fitness Gbest* di iterasi ke-*t* pada *Gbest* ke-*i*.
$Gbest_i^{t+1}$ = Nilai *fitness Gbest* di iterasi ke-*t+1* pada *Gbest* ke-*i*.
$argmax\{fitness(Pbest_i^{t+1})\}$ = Nilai *fitness Pbest* terbaik ke-*i* di iterasi ke-*t*.

ulangi langkah 2 – 5 hingga mencapai iterasi yang telah ditentukan.

2.5 Perhitungan Evaluasi

Perhitungan evaluasi ini digunakan pada proses perhitungan *fitness* pada algoritme *Particle Swarm Optimization* dan digunakan untuk proses evaluasi dari proses pelatihan pada algoritme *Learning Vector Quantization.* Perhitungan evaluasi sendiri memiliki tahapan beirkut:

1. Hitung jarak antara bobot/partikel dengan data yang digunakan sebagai data latih sehingga akan didapatkan jarak dari setiap bobot / partikel yang ada. Jarak tersebut nantinya akan dicari jarak terpendeknya dan akan dilihat apakah jarak terpendek tersebut berada di kelas yang sama dengan kelas dari data latih yang digunakan dalam perhitungan jarak. Perhitungan jarak sendiri menggunakan persamaan berikut:

$$D_i = \sqrt{\sum_{j=1}^{n}(X_j - W_{ij})^2} \quad (17)$$

   Keterangan:
   D_i = Jarak antara bobot/partikel dan data pada kelas-*i*.
   n = Jumlah seluruh parameter yang ada.
   X_j = Parameter data latih ke-*j*.
   W_{ij} = Bobot pada kelas ke-*i* dan parameter ke-*j*.

2. Selanjutnya hitung jumlah data latih yang memiliki kesamaan kelas antara bobot/partikel dengan kelas data latih berdasarkan jarak terkecil yang telah dihitung dengan Persaaan 17. perhitungan jumlah data ini menggunakan persamaan berikut:

$$Akurasi = \frac{c}{n} \quad (18)$$

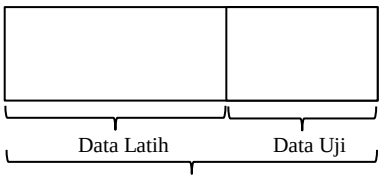
$$Akurasi = Akurasi * 100\% \quad (19)$$

   Keterangan:
   c = Jumlah seluruh data uji yang sesuai.
   n = Jumlah seluruh data uji.

2.6 Holdout Method

Holdout Method merupakan salah satu dari beberapa metode yang ada dalam Cross Validation. *Holdout Validation* merupakan

metode yang sederhana dalam proses validasi data, dimana data yang ada akan dibagi menjadi dua bagian yaitu data latih dan juga data uji. Data latih merupakan data yang nantinya akan digunakan dalam proses pelatihan / *training* data, setelah melakukan proses *training* data barulah data akan diuji menggunakan data uji yang sebelumnya telah ditentukan.

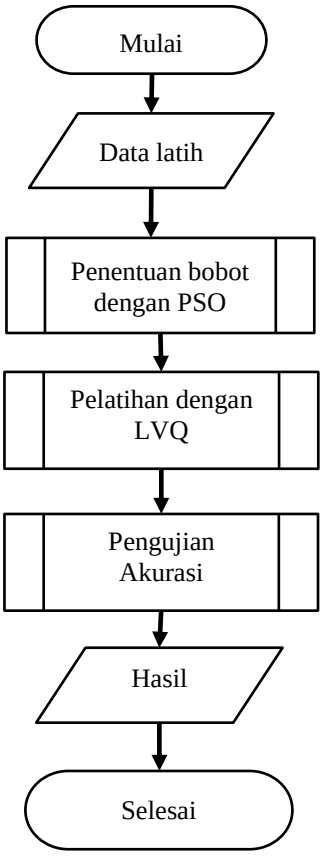


Gambar 1. Pembagian Data Holdout Method

3. PERANCANGAN

3.1. Siklus Algoritme

Pada bagian ini akan menjelaskan siklus algoritme *Particle Swarm Optimization – Learning Vector Quantization* (PSO-LVQ) dalam melakukan proses klasifikasi pada data iris. Gambar 2 akan menjelaskan proses kerja algoritme secara keseluruhan.



Gambar 2 Alur Algoritme dari PSO - LVQ

1. Membaca masukan data latih.
2. Melakukan proses perhitungan untuk mendapatkan bobot terbaik dengan algoritme PSO. Pada proses ini diawali dengan pembentukan partikel awal seperti Tabel 1, dimana setiap partikel terdapat tiga buah bobot yang mewakili setiap kelas yang ada. Selain itu dilakukan perhitungan nilai *fitness* dengan menghitung nilai akurasi dengan data uji menggunakan perhitungan evaluasi.

Tabel 1. Partikel Awal

	Sepal length	Sepal Width	Petal Length	Petal Width	Fitness
P	4,9	2,7	1,4	0,2	-
	7	3,2	4,7	1,4	
	6,3	3,3	6	2,5	

Selanjutnya dilakukan proses inisialisasi pada pbest dan gbest. Setelah proses inisialisasi dilanjutkan dengan proses update terhadap partikel, pbest dan juga gbest.

3. Melakukan proses pengklasifikasian data dengan algoritme LVQ. Pada proses ini dilakukan proses inisialisasi bobot awal dengan menggunakan hasil gbest terakhir pada proses pada algoritme PSO.
4. Melakukan pengujian untuk mengetahui tingkat akurasi dari keseluruhan proses dengan menggunakan bobot hasil pelatihan dengan algoritme LVQ dan juga data uji.
5. Menampilkan hasil dari pengujian yang telah dilakukan.

4. ANALISIS DAN PENGUJIAN

4.1 Pengujian Data

Pengujian data ini menggunakan metode *Holdout Method* dengan membagi data menjadi dua, yaitu data latih dan data uji. Pengujian ini akan dilakukan sebanyak 5 kali dan juga akan dilihat bagaimana rata-rata secara keseluruhan algoritme ini. Selain itu pengujian ini menggunakan nilai parameter berikut:

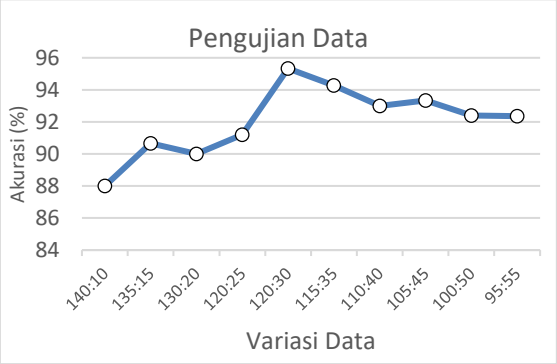
- *Particle Swarm Optimization*
  - Jumlah partikel : 10
  - Iterasi : 50
  - *w* : 0,5

- $c1$ dan $c2$: 0,7
- $r1$ dan $r2$: 0,4

- *Learning Vector Quantization*
  - Iterasi : 50
  - *Alpha* (α) : 0,01
  - *Epsilon* (ε) : 0,35

Tabel 2. Hasil Pengujian Data

No	Perbandingan Data		Percobaan ke-*i* (%)					Rata-Rata Akurasi (%)
	Data latih	Data Uji	1	2	3	4	5	
1	140	10	70	80	100	100	90	88
2	135	15	86,67	93,33	93,33	100	80	90,666
3	130	20	90	90	95	85	90	90
4	125	25	100	92	92	92	80	91,2
5	120	30	83,33	100	96,67	100	96,67	95,334
6	115	35	88,57	100	88,57	97,14	97,14	94,284
7	110	40	95	90	95	92,5	92,5	93
8	105	45	97,78	88,89	95,56	93,33	91,11	93,334
9	100	50	94	94	94	88	92	92,4
10	95	55	85,45	94,54	96,37	89,09	96,37	92,364



Gambar 3. Grafik Akurasi Hasil Pengujian Data

Dari Gambar 3 didapatkan bahwa nilai rata-rata tertinggi ada pada pengujian ke-5 dengan rata-rata 95,334 persen dan rata-rata terendah pada pengujian ke-1 dengan rata-rata 88. Pengujian ke-1 memiliki nilai terendah dikarenakan data uji yang digunakan sedikit, hal tersebut yang membuat nilai rata-rata akurasinya rendah begitu juga apabila data latihnya yang sedikit, akan menyebabkan rendahnya nilai akurasi dikarenakan kurangnya proses pelatihan terhadap bobot data. Hal tersebut dapat dilihat pada Gambar 3 bahwa semakin berkurangnya data uji menyebabkan turunnya akurasi.

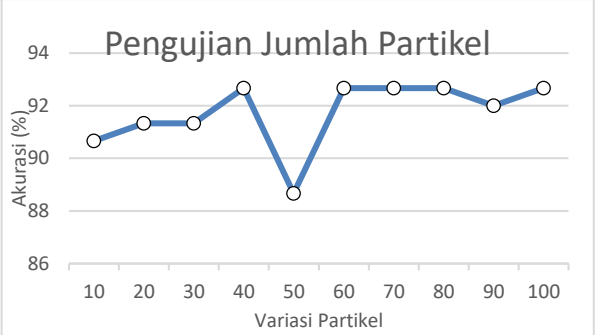
4.2 Pengujian Jumlah Partikel

Pengujian partikel ini dilakukan untuk melihat skenario partikel mana yang menghasilkan hasil akurasi terbaik. Pada pengujian ini akan digunakan nilai parameter berikut:

- *Particle Swarm Optimization*
  - Iterasi : 50
  - w : 0,5
  - $c1$ dan $c2$: 0,7
  - $r1$ dan $r2$: 0,4
- *Learning Vector Quantization*
  - Iterasi : 50
  - *Alpha* (α) : 0,01
  - *Epsilon* (ε) : 0,35

Tabel 3. Hasil Pengujian Partikel

No	Jumlah Partikel	Percobaan ke-*i* (%)					Rata-Rata Akurasi (%)
		1	2	3	4	5	
1	10	90	93,33	83,33	90	96,67	90,666
2	20	96,67	90	86,67	90	93,33	91,334
3	30	96,67	90	93,33	90	86,67	91,334
4	40	86,67	86,67	100	93,33	96,67	92,668
5	50	70	96,67	86,67	93,33	96,67	88,668
6	60	83,33	100	96,67	90	93,33	92,666
7	70	96,67	100	86,67	90	90	92,668
8	80	90	93,33	86,67	100	93,33	92,666
9	90	93,33	100	86,67	90	90	92
10	100	86,67	96,67	90	93,33	96,67	92,668



Gambar 4. Grafik Hasil Pengujian Jumlah Partikel

Dari pengujian jumlah partikel yang telah dilakukan, didapatkan nilai akurasi terendah yang bernilai 70 persen pada percobaan variasi partikel 50, hal tersebut terjadi dikarenakan nilai random yang dihasilkan dari setiap partikel memiliki nilai yang kurang bagus sehingga menghasilkan akurasi yang kurang bagus juga. Jumlah partikel sendiri berpengaruh terhadap variasi nilai yang dihasilkan dalam proses PSO, semakin banyak partikel yang digunakan semakin besar kemungkinan hasil yang dihasilkan semakin baik, namun dengan jumlah partikel yang banyak dapat menyebapkan proses perhitungan melambat dikarenakan banyaknya partikel yang harus diproses.

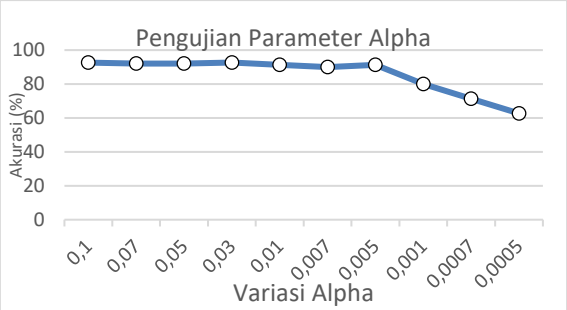
4.3 Pengujian Parameter Alpha pada PSO

Pengujian nilai *alpha* ini dilakukan untuk melihat skenario nilai *alpha* mana yang menghasilkan hasil akurasi terbaik. Pada pengujian ini akan digunakan nilai parameter berikut:

- *Particle Swarm Optimization*
  - Iterasi : 50
  - Partikel : 40
  - w : 0,5
  - $c1$ dan $c2$: 0,7
  - $r1$ dan $r2$: 0,4
- *Learning Vector Quantization*
  - Iterasi : 50
  - *Epsilon* (ε) : 0,35

Tabel 4. Hasil Pengujian Nilai Alpha

No	Nilai *Alpha*	Percobaan ke-*i* (%)					Rata-Rata Akurasi (%)
		1	2	3	4	5	
1	0,1	100	86,67	93,33	93,33	90	92,666
2	0,07	96,67	93,33	90	93,33	86,67	92
3	0,05	100	90	93,33	86,67	90	92
4	0,03	90	96,67	86,67	90	100	92,668
5	0,01	96,67	93,33	93,33	90	83,33	91,332
6	0,007	96,67	80	90	86,67	96,67	90,002
7	0,005	80	96,67	90	96,67	93,33	91,334
8	0,001	70	96,67	90	50	93,33	80
9	0,0007	50	73,33	60	80	93,33	71,332
10	0,0005	30	66,67	83,33	63,33	70	62,666



Gambar 5. Grafik Hasil Pengujian Parameter Alpha

Berdasarkan Tabel 4 diketahui bahwa rata-rata akurasi tertinggi sebesar 92,668 persen dengan nilai akurasi tertinggi sebesar 100 persen pada pengujian dengan nilai *alpha* 0,03. Hasil pengujian tersebut menunjukkan bahwa pengaruh nilai *alpha* cukup signifikan dalam mempengaruhi hasil akurasi, dimana nilai *alpha* yang kecil dapat mengurangi proses pelatihan pada LVQ yang mengakibatkan nilai akurasi berkurang. Hal tersebut ditunjukkan penurunan akurasi pada percobaan dengan nilai *alpha* 0,001 hingga 0,0005.

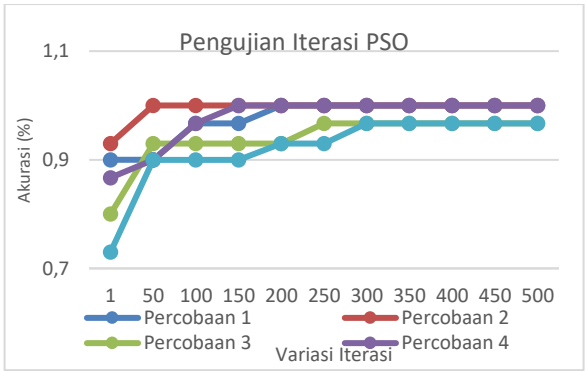
4.4 Pengujian Jumlah Iterasi PSO

Pengujian jumlah iterasi ini dilakukan untuk melihat skenario mana yang menghasilkan hasil akurasi terbaik. Pada pengujian ini akan digunakan nilai parameter berikut:

- *Particle Swarm Optimization*
  - Partikel : 40
  - w : 0,5
  - $c1$ dan $c2$: 07
  - $r1$ dan $r2$: 0,4
- *Learning Vector Quantization*
  - Iterasi : 50
  - *Alpha* (α) : 0,03
  - *Epsilon* (ε) : 0,35

Tabel 5. Hasil Pengujian Iterasi PSO

No		1	2	3	4	5	6	7	8	9	10	11
Iterasi		1	50	100	150	200	250	300	350	400	450	500
Percobaan	1	0,9	0,9	,0967	,0967	1	1	1	1	1	1	1
	2	0,933	1	1	1	1	1	1	1	1	1	1
	3	0,8	0,93	0,933	0,933	0,933	0,967	0,967	0,967	0,967	0,967	0,967
	4	0,867	0,9	0,967	1	1	1	1	1	1	1	1
	5	0,73	0,9	0,9	0,9	0,933	0,933	0,967	0,967	0,967	0,967	0,967



Gambar 6. Grafik Hasil Pengujian Iterasi PSO

Dari Gambar 6 dapat terlihat, perubahan nilai akurasi pada gbest mulai menghasilkan nilai yang stabil pada iterasi ke 300 pada setiap percobaan. Dari percobaan yang telah dilakukan dapat dilihat bahwa nilai yang dihasilkan oleh gbest yang nantinya akan dijadikan sebagai bobot awal memiliki nilai akurasi 1 dan juga 0,967.

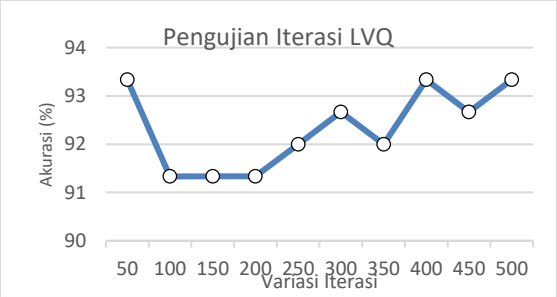
4.5 Pengujian Jumlah Iterasi LVQ

Pengujian jumlah iterasi ini dilakukan untuk melihat skenario mana yang menghasilkan hasil akurasi terbaik. Pada pengujian ini akan digunakan nilai parameter berikut:

- *Particle Swarm Optimization*
  - Iterasi : 300
  - Partikel : 40
  - w : 0,5
  - $c1$ dan $c2$: 07
  - $r1$ dan $r2$: 0,4
- *Learning Vector Quantization*
  - *Alpha* (α) : 0,03
  - *Epsilon* (ε) : 0,35

Tabel 6. Hasil Pengujian Iterasi LVQ

No	Iterasi	Percobaan ke-*i* (%)					Rata-Rata Akurasi (%)
		1	2	3	4	5	
1	50	100	93,33	96,67	90	86,67	93,334
2	100	83,33	93,33	96,67	93,33	90	91,332
3	150	90	90	96,67	86,67	93,33	91,334
4	200	86,67	96,67	93,33	86,67	93,33	91,334
5	250	86,67	93,33	96,67	93,33	90	92
6	300	93,33	93,33	96,67	90	90	92,666
7	350	93,33	96,67	90	93,33	86,67	92
8	400	90	90	96,67	100	96,67	93,334
9	450	86,67	96,67	96,67	93,33	90	92,668
10	500	96,67	100	90	86,67	93,33	93,334



Gambar 7. Grafik Hasil Pengujian Iterasi LVQ

Hasil pengujian didapatkan bahwa perubahan iterasi tidak begitu berpengaruh pada hasil percobaan iterasi ini. Hal tersebut terjadi dikarenakan nilai bobot yang merupakan hasil perhitungan gbest dapat membantu meningkatkan akurasi dari proses ini. Namun jumlah iterasi berpengaruh terhadap lamanya proses pelatihan, karena semakin banyak proses iterasi yang dilakukan membutuhkan waktu untuk memprosesnya.

4.6 Analisis Global

Pada analisis global ini akan dilakukan perbandingan dari algoritme *Particle Swarm Optimization – Learning Vector Quantization* (PSO-LVQ) dengan algoritme *Learning Vector Quantization* (LVQ). Untuk jumlah data latih yang digukanan sebanyak 75 data latih, begitu juga untuk data uji yang digunakan sebanyak 75. Selain itu nilai parameter yang digunakan sama dengan menggunakan nilai parameter terbaik dari pengujian sebelumnya yaitu:

- *Particle Swarm Optimization*
  - Iterasi : 300
  - Partikel : 40
  - w : 0,5
  - $c1$ dan $c2$: 0,7
  - $r1$ dan $r2$: 0,4
- *Learning Vector Quantization*
  - Iterasi : 50
  - *Alpha* (α) : 0,03
  - *Epsilon* (ε) : 0,35

Tabel 7. Hasil Perbandingan Pengujian

Algoritme	Percobaan ke-*i* (%)					Rata-Rata Akurasi (%)
	1	2	3	4	5	
PSO-LVQ	89,33	94,67	94,67	90,67	97,33	93,334
LVQ	96	90,67	88	60	86,67	84,268
Selisih						9,066

Tabel 8. Hasil Time Excecution

Algoritme	Percobaan ke-*i* (milisecond)					Rata-Rata
	1	2	3	4	5	
PSO-LVQ	286	285	306	322	301	300
LVQ	158	153	151	156	153	154,2

Dari hasil Tabel 7 dan Tabel 8 didapatkan bahwa nilai akurasi pada algoritme LVQ mengalami peningkatan dimana akurasi terbaik pada algoritme PSO-LVQ adalah 97,33% dan pada algoritme LVQ sendiri adalah 96%. Pada algoritme LVQ sendiri terdapat nilai akurasi yang rendah yaitu 60 %, hal tersebut disebabkan nilai bobot awal yang diperoleh merupakan nilai bobot awal yang kurang bagus sehingga nilai akurasi yang diperoleh juga tidak sesuai harapan. Namun walaupun dapat mengoptimalkan nilai akurasi dari algortime LVQ, algoritme PSO-LVQ memiliki waktu eksekusi yang cukup lama dibandingkan dengan algoritme LVQ pada umumnya yaitu 300 *milisecond* atau sekitar 0,3

detik, berbeda dengan LVQ yang memiliki waktu eksekusi 154,2 *milisecond* atau sekitar 0,15 detik.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang berjudul "Penerapan Algoritme *Particle Swarm Optimization – Learning Vector Quantization* (PSO-LVQ) pada Klasifikasi Data Iris", maka dapat disimpulkan bahwa:

1. Algoritme *Particle Swarm Optimization – Learning Vector Quantization* (PSO-LVQ) untuk klasifikasi data *Iris* dapat diimplementasikan dengan baik. Langkah dalam proses penerapan algoritme sendiri diawali dengan melakukan pembagian data yang sebelumnya diambil dari website UCI *Machine Learning*, data tersebut dibagi menjadi dua bagian yaitu data latih dan data uji dengan menggunakan metode *Holdout Method*. Selanjutnya dilakukan proses penentuan bobot awal sebagai kelas prediksi dengan algoritme PSO untuk mendapatkan bobot terbaik yang nantinya digunakan pada algoritme LVQ. Setelah mendapatkan bobot terbaik, dilakukan proses pelatihan pada algoritme LVQ sehingga didapatkan bobot prediksi dari setiap kelas. Hasil bobot tersebut kemudian diuji dengan menggunakan data uji yang telah dibuat sebelumnya berupa nilai akurasi yang diperoleh dengan melakukan perhitungan kedekatan data uji dengan bobot hasil perhitungan.
2. Hasil akurasi yang diperoleh dari algoritme *Particle Swarm Optimization – Learning Vector Quantization* (PSO-LVQ) berdasarkan perbandingan data uji dan data latih sebesar 120:30 didapatkan rata-rata akurasi 95,334%. Selain itu nilai akurasi berdasarkan jumlah partikel = 40, w = 0,5, $c1$ = 0,7, $c2$ = 0,7, $r1$ = 0,4, $r2$ = 0,4, iterasi PSO = 300, nilai *alpha* = 0,05, iterasi LVQ = 50, dan *epsilon* = 0,35 diperoleh rata-rata 93,334% dengan akurasi tertinggi 100%. Selain itu hasil perbandingan dengan algoritme LVQ didapatkan bahwa algoritme PSO-LVQ mampu mengoptimalkan akurasi dari LVQ dimana akurasi rata-rata PSO-LVQ sebesar 93,334% dan LVQ sebesar 84,268% dengan selisih 9,066%. Namun dari segi waktu ekseskusi waktu yang dibutuhkan PSO-LVQ lebih lama dengan rata-rata waktu eksekusi 0,3 detik berbeda dengan LVQ yang membutuhkan waktu eksekusi 0,15 detik.

5.2 Saran

Proses klasifikasi ini masih memiliki kekurangan, saran yang mampu diberikan untuk pengembangan klasifikasi dengan algoritme ini kedepannya yaitu:

1. Pada penelitian selanjutnya diharapkan dapat melakukan optimasi bobot dengan algoritme lain sehingga akan didapat hasil akurasi yang lebih baik. Karena hasil akurasi yang didapatkan dari gbest pada proses PSO dipengaruhi jumlah partikel dan juga jumlah iterasi pada PSO sehingga perlu dilakukan pencarian terlebih dahulu pada kedua nilai tersebut.
2. Pada penelitian selanjutnya dapat dilakukan pengujian terhadap nilai *epsilon* sehingga dapat diketahui pengaruh epsilon terhadap tingkat akurasi dari LVQ yang digunakan.
3. Pada penelitian selanjutnya dapat dikembangkan dengan variasi LVQ yang lain seperti LVQ3. Karena pada LVQ3 proses pelatihan juga dipengaruhi oleh nilai epsilon sehingga akan didapatkan hasil yang lebih baik.

6. DAFTAR PUSTAKA

Arniantya, R., Setiawan, B. D. & Adikara, P. P., 2018. Optimasi Vektor Bobot Pada Learning Vector Quantization Menggunakan Algoritme Genetika Untuk Identifikasi Jenis Attention Deficit Hyperactivity Disorder Pada Anak. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Volume 2, pp. 679-687.

Azmi, M., 2014. Komparasi Metode Jaringan Syaraf Tiruan Som Dan Lvq Untuk Mengidentifikasi Data Bunga Iris. *Jurnal TEKNOIF*, Volume 2, pp. 64-70.

Fausett, L. V., 1994. *Fudamentals of Neural Networks (Architectures, Algorithms, and Applications.* New Jersey: Prentice-Hall.

Fisher, R. A., 2018. *Iris Data Set.* [Online] Available at: https://archive.ics.uci.edu/ml/datasets/iris [Accessed 20 January 2018].

Haldar, R. & Mishra, P. K., 2016. Learning Vector Quantization (LVQ) Neural Network Approach for Multilingual Speech Recognition. *International Research Journal of Engineering and Technology (IRJET)*, 03(05), pp. 2863-2869.

Han, J. & Kamber, M., 2006. *Data Mining: Concepts and Techniques.* 2nd ed. San Francisco: Elsevier Inc.

Hardinata, J. T. et al., 2017. *Modification Of Learning Rate With Lvq Model Improvement In Learning Backpropagation.* Medan, IOPSCIENCE.

Haykin, S., 2001. *Neural Networks: A Comprehensive Foundation.* 2nd ed. Ontario: Pearson Education Inc.

Jatmiko, W. et al., 2009. Fuzzy Learning Vector Quantization Based on Particle Swarm Optimization For Artificial Odor Dicrimination System. *WSEAS TRANSACTIONS on SYSTEMS*, 8(12), pp. 1239-1252.

Kennedy, J. & Eberhart, E., 1995. *Particle swarm optimization.* Perth, WA, Australia, IEEE Xplore.

Rahardian, B. A., Dewi, C. & Rahayudi, B., 2018. Implementasi Genetic Algorithm Dan Artificial Neural Network Untuk Deteksi Dini Jenis Attention Deficit Hyperactivity Disorder. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer* , Volume 2, pp. 688-694.

Swain, M., Dash, S. K., Dash, S. & Mohapatra, A., 2012. An Approach For Iris Plant Classification Using Neural Network. *International Journal on Soft Computing*, Volume 3, pp. 79-89.

Tan, H. T. W. & Xingli, G., 2008. *Plant Magic: Auspicious and Inauspicious Plants from Around the World.* 1st ed. Singapura: Marshall Cavendish International (Asia).

Wuryandari, M. D. & Afrianto, I., 2012. Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation dan Learning Vector Quantization pada Pengenalan Wajah. *Jurnal Komputer dan Informatika (KOMPUTA)*, Volume 1, pp. 45-51.