ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – KI141502

SELEKSI FITUR MENGGUNAKAN METODE *HYBRID PARTICLE SWARM OPTIMIZATION* DENGAN OPERASI *LOCAL SEARCH* (*HPSO-LS*) UNTUK KLASIFIKASI DATA

DIMAS YOAN SHAILENDRA
NRP 5112100162

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

SELEKSI FITUR MENGGUNAKAN METODE *HYBRID PARTICLE SWARM OPTIMIZATION* DENGAN OPERASI *LOCAL SEARCH* (HPSO-LS) UNTUK KLASIFIKASI DATA

DIMAS YOAN SHAILENDRA
NRP 5112100162

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

FEATURE SELECTION USING HYBRID PARTICLE SWARM OPTIMIZATION WITH LOCAL SEARCH OPERATION (HPSO-LS) FOR DATA CLASSIFICATION

DIMAS YOAN SHAILENDRA
NRP 5112100162

Supervisor I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Supervisor II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

SELEKSI FITUR MENGGUNAKAN METODE *HYBRID PARTICLE SWARM OPTIMIZATION* DENGAN OPERASI *LOCAL SEARCH* (*HPSO-LS*) UNTUK KLASIFIKASI DATA

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
DIMAS YOAN SHAILENDRA
NRP : 5112 100 162

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP *197512202001122002*) (Pembimbing 1)
2. Dr.Eng. Nanik Suciati, S.Kom., M.Kom. .....
   (NIP *197104281994122001*) (Pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

Seleksi Fitur Menggunakan Metode *Hybrid Particle Swarm Optimization* dengan Operasi *Local Search* (*HPSO-LS*) untuk Klasifikasi Data

Nama Mahasiswa	**: DIMAS YOAN SHAILENDRA**
NRP	**: 5112100162**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**

Abstrak

Seleksi fitur adalah suatu proses untuk memilih fitur penting dalam suatu dataset dengan mengurangi fitur-fitur yang redundant. Tujuan dari tugas akhir ini adalah untuk menerapkan algoritma hybrid particle swarm optimization dengan operasi local search (HPSO-LS) dalam melakukan seleksi fitur.

Metode HPSO-LS menggunakan operasi local search yang ditanamkan pada algoritma particle swarm optimization (PSO) untuk memilih fitur yang tidak berkorelasi satu sama lain. Operasi local search didasarkan pada nilai korelasi antar fitur. Semakin kecil nilai korelasi fitur, semakin besar kemungkinan fitur dipilih.

Dataset yang digunakan untuk melakukan uji coba sebanyak 3 macam yaitu data glass, wine dan heart. Uji coba dilakukan dengan menerapkan algoritma HPSO-LS dan juga algoritma PSO untuk melakukan seleksi fitur. Hal tersebut dilakukan untuk menganalisa perbandingan hasil seleksi fitur menggunakan algortima HPSO-LS dengan algoritma PSO.

Hasil uji coba menunjukkan bahwa, akurasi klasifikasi data dipengaruhi dengan jumlah fitur yang digunakan dan kombinasi fitur-fitur tersebut. Klasifikasi data k-NN dengan

menggunakan kombinasi fitur-fitur yang memiliki korelasi yang rendah, lebih besar akurasinya jika dibandingkan dengan fitur-fitur dengan korelasi tinggi. Hasil uji coba juga menunjukkan bahwa menggunakan fitur terseleksi hasil seleksi fitur HPSO-LS dapat meningkatkan akurasi klasifikasi data k-NN jika dibandingkan dengan menggunakan fitur keseluruhan. Akurasi meningkat sebesar 2,33% pada dataset glass, 4,49% pada dataset wine dan 6,29% pada dataset heart.

Kata Kunci: Feature Selection, Particle Swarm Optimization, Correlation Information, Local search, k-Nearest Neighbor.

Feature Selection Using Hybrid Particle Swarm Optimization with Local Search Operation (*HPSO-LS*) for Data Clasification

Nama Mahasiswa	**: DIMAS YOAN SHAILENDRA**
NRP	**: 5112100162**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**

Abstract

Feature selection is a process for selecting important features in a dataset by reducing redundant features. The purpose of this final project is to apply hybrid particle swarm optimization algorithm with local search (HPSO-LS) operation for feature selection.

The HPSO-LS method uses a local search process embedded in the particle swarm optimization (PSO) algorithm to select features that do not correlate with each other. The local search process is based on correlation values between features. The smaller the value of the correlation the more likely the feature is selected.

There are 3 datasets that being used on this experiment which is glass, wine and heart. Experiment were performed by applying the HPSO-LS algorithm and the PSO algorithm to perform feature selection. The purpose of that two algorithm are performed, is to analyze the comparison of feature selection result using HPSO-LS and PSO.

The experimental results show that the accuracy of data classification is influenced by the number of used features and the combination of the features. K-NN classification by using a combination of features that have low correlation, will

have greater accuracy compared with feature that have high correlation. The result also show that using selected feature of HPSO-LS feature selection can improve the accuracy of k-NN classification compared to using overall feature. An accuracy of 2.33% increase in glass dataset, 4.49% in wine dataset and 6.29% in heart dataset.

Kata Kunci: Feature Selection, Particle Swarm Optimization, Local search, Correlation Coefficien, k-Nearest Neighbor

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur penulis kehadirat Allah SWT karena berkat rahmat dan karunia-NYA penulis dapat menyelesaikan Tugas Akhir yang berjudul

SELEKSI FITUR MENGGUNAKAN METODE *HYBRID PARTICLE SWARM OPTIMIZATION* DENGAN OPERASI *LOCAL SEARCH (HPSO-LS)* UNTUK KLASIFIKASI DATA

Tugas Akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis ingin menyampaikan terima kasih yang sebesar-besarnya atas dukungan dan semangat yang diberikan dan membantu penulis baik secara langsung ataupun tidak dalam menyelesaikan Tugas Akhir ini. Penulis ingin mengucapkan terima kasih kepada

1. Allah SWT karena berkat rahmat dan karunianya penulis berhasil menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, dan keluarga penulis, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
5. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku Dosen Pembimbing I Tugas Akhir yang telah

memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.

6. Ibu Nanik Suciati, S.Kom., M.Kom., Dr.Eng. selaku pembimbing II Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
7. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
8. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.

Penulis mohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Juni 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini, gambaran tugas akhir secara umum dapat dipahami.

1.1 Latar Belakang

Ruang fitur pada *dataset* merupakan faktor kunci dalam mempengaruhi performa proses klasifikasi data. Jika suatu objek memiliki ruang fitur yang besar, maka dapat membuat kinerja algoritma menjadi lebih lama. Tidak semua fitur pada suatu objek atau *dataset* mampu memberikan hasil klasifikasi yang baik. Karena masih terdapat beberapa fitur yang *redundant*.

Kombinasi subset fitur yang optimal dicari dengan melakukan perhitungan dan evaluasi terhadap semua kumpulan fitur yang mungkin. Seluruh pencarian mencakup keseluruhan fitur. Oleh karena itu, permasalahan menemukan subset fitur yang optimal tersebut merupakan *NP-hard problem* [5,6]. Sehingga proses tersebut memerlukan waktu komputasi yang lama dan tidak praktis.

Salah satu solusi untuk menangani masalah tersebut adalah dengan menggunakan seleksi fitur. Seleksi fitur adalah proses mengurangi fitur-fitur yang *redundant* pada *dataset*. Seleksi fitur ini dapat meningkatkan kualitas dari ruang fitur sehingga dapat meningkatkan akurasi klasifikasi data.

Beberapa jenis metode seleksi fitur diantaranya *filter*, *wrapper* dan *hybrid.* Metode *wrapper* adalah metode yang menggunakan suatu *learning algorithm* untuk melakukan evaluasi terhadap *subset* fitur yang terpilih secara iteratif. Hasil dari *learning algorithm* tersebut digunakan sebagai petunjuk

untuk iterasi berikutnya. Jika metode *filter* tidak menggunakan *learning algorithm* melainkan menggunakan perhitungan statistik pada data. Proses dari metode *filter* lebih cepat daripada metode *wrapper*. Namun kualitas *subset* fitur yang terpilih, metode *wrapper* lebih bagus daripada metode *filter*, dilihat dari hasil klasifikasi data [1]. Sedangkan metode *hybrid* adalah metode yang mengkombinasikan metode *filter* dan *wrapper*.

Metode *wrapper* terbagi menjadi *sequential search method* dan *global search method* [1]. Sejumlah algoritma telah diterapkan dalam *sequential search method*, dengan proses pencarian fitur yang dipandu oleh *learning algorithm* tertentu [7,8]. Namun metode tersebut masih memiliki kekurangan yaitu hanya mencari fitur pada pencarian lokal saja [1]. *Global search method* menerapkan metode pemilihan fitur secara *random* ke dalam strategi pencariannya untuk mengeksplorasi sebagian besar ruang solusi untuk melakukan seleksi fitur secara lebih baik [1]. Salah satu yang sudah diterapkan adalah menggunakan algoritma *Particle Swarm Optimization* (*PSO*) [7,8]. Algoritma ini lebih mudah diimplementasikan, karena memiliki sedikit parameter yang digunakan dan kecepatan komputasinya tinggi. Namun *PSO* juga memiliki kekurangan, yaitu tidak memperhatikan jumlah fitur pada *search process*, dan hanya menekankan pemilihan fitur yang memiliki akurasi tinggi pada proses klasifikasi data. Hal tersebut masih memungkinan memilih fitur yang *redundant* pada hasil fitur-fitur terpilihnya [1]. Fitur-fitur *redundant* adalah fitur-fitur yang memiliki pola yang sama. Pola tersebut adalah pola dari kesamaan nilai fitur. Oleh karena itu fitur-fitur yang memiliki pola yang sama dapat disebut fitur yang *redundant*. Pemilihan fitur yang *redundant* tersebut memungkinkan terjadinya penurunan kualitas ruang fitur. Karena jika ada beberapa fitur yang memiliki pola yang sama, sebaiknya dipilih atau diambil salah satu fitur saja untuk diproses pada klasifikasi data.

Oleh karena itu, tujuan dari tugas akhir ini adalah menerapkan algoritma *Particle Swarm Optimization* dengan

menambahkan operasi *Local Search* (*HPSO-LS*) untuk melakukan seleksi fitur. Operasi *local search* tersebut tertanam pada *PSO* untuk menyempurnakan proses pencarian fitur. Operasi *local search* yaitu dengan memilih fitur yang relatif tidak berkorelasi dengan fitur lainnya. Atau dengan kata lain, operasi *local search* meminimalisir fitur-fitur yang *redundant*. *HPSO-LS* menggunakan nilai korelasi *(correlation information)* untuk mengetahui hubungan antar fitur-fitur. Jika suatu fitur memiliki nilai korelasi yang tinggi maka fitur tersebut memiliki hubungan dengan fitur lainnya dan memiliki sedikit peluang untuk dipilih pada proses seleksi fitur *HPSO-LS*. Sebaliknya, jika fitur memiliki nilai korelasi yang rendah, maka semakin tinggi kemungkinan fitur tersebut dipilih [1]. Implementasi algoritma tersebut diharapkan dapat lebih mengoptimalkan ruang fitur sehingga dapat meningkatkan akurasi klasifikasi data.

1.2 Rumusan Masalah

Tugas akhir ini memiliki beberapa rumusan masalah yang mendasari diantaranya:

a) Bagaimana meminimalisir pemilihan fitur-fitur yang *redundant* dalam suatu *dataset*?
b) Bagaimana menerapkan proses *local search* pada algoritma *HPSO-LS* untuk melakukan seleksi fitur?
c) Bagaimana pengaruh fitur-fitur terpilih terhadap performa klasifikasi data *k-NN*?
d) Bagaimana perbandingan performa klasifikasi data *k-NN* menggunakan fitur-fitur terpilih hasil algoritma *HPSO-LS* dan *PSO* maupun dengan fitur keseluruhan?

1.3 Batasan Masalah

Tugas akhir ini memiliki beberapa batasan terkait implementasinya, diantaranya:

a) *Dataset* diambil dari *UCI Machine Learning Repository*.
b) *Dataset input*-an menggunakan 3 *dataset* yaitu *glass, wine* dan *heart.*
c) Perangkat lunak dibuat menggunakan bahasa pemrograman Matlab.
d) Data yang dapat digunakan dalam perangkat lunak ini berupa data bilangan.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah melakukan seleksi fitur menggunakan metode *hybrid particle swarm optimization* dengan operasi *Local Search* (*HPSO-LS*) untuk klasifikasi data. Seleksi fitur dilakukan dengan meminimalisir pemilihan fitur yang *redundant.* Sehingga proses peminimalisiran tersebut dapat meningkatkan akurasi klasifikasi data *k-NN*. Serta bertujuan untuk mengetahui pengaruh fitur-fitur terpilih tersebut terhadap performa klasifikasi data *k-NN*.

1.5 Manfaat

Tugas akhir ini diharapkan mampu memberikan alternatif solusi dalam melakukan proses klasifikasi data secara efisien. Solusinya adalah dengan melakukan seleksi fitur pada *dataset*. Sehingga proses seleksi fitur diharapkan dapat meningkatkan kecepatan dan akurasi dari proses klasifikasi data tersebut.

1.6 Metodologi

Metodologi yang dipakai pada pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir.
   Tahap awal yang dilakukan dalam pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Di dalam

proposal diajukan suatu gagasan pembuatan perangkat lunak untuk melakukan seleksi fitur menggunakan *HPSO-LS*.

2. Studi literatur.
   Tahap ini melakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan *particle swarm optimization*, *correlation coefficient*, *local search* dan *k-nearest neighbour*. Literatur yang digunakan meliputi: jurnal dan dokumentasi dari internet. Penyaringan literatur dilakukan dengan memilih teori-teori yang digunakan untuk pengerjaan tugas akhir ini. Pembelajaran literatur dilakukan baik secara mandiri maupun dengan melakukan bimbingan dosen pembimbing. Teori-teori yang dipelajari meliputi seleksi fitur, algoritma *PSO*, operasi *local search*, klasifikasi data menggunakan *k-NN, pearson coefficient coreelation, linear normalization, confussion matrix* dan *k-fold cross validation.* Data uji coba yang digunakan meliputi *dataset glass*, *wine* dan *heart.* Data uji coba tersebut didapatkan dari *website UCI Machine Learning Repository*. *Dataset glass* memiliki 9 jumlah fitur, 214 jumlah *instances* dan 7 jumlah *class. Dataset wine* memiliki 13 jumlah fitur, 175 jumlah *instances* dan 3 jumlah *class. Dataset heart* memiliki 13 jumlah fitur, 270 jumlah *instances* dan 2 jumlah *class.* Studi literatur juga dilakukan pada program pembangun perangkat lunak. Program yang digunakan untuk membuat perangkat lunak ini adalah Matlab R2014a. Studi dilakukan dengan mencari literatur dari internet dan diskusi.
3. Pembuatan perangkat lunak.
   Tahap ini dilakukan pembuatan perangkat lunak sesuai dengan rancangan perangkat lunak yang dibuat. Rancangan desain perangkat lunak ini mengacu pada *paper* yang didapatkan [1]. Langkah-langkah pembuatan perangkat lunak ini diantaranya:

- Perancangan desain perangkat lunak.
  Perancangan desain perangkat lunak mengacu pada *paper* yang didapatkan [1]. Proses-proses dari perangkat lunak diantaranya *input* jumlah subset fitur (*sf*), mengelompokkan fitur, menormalisasi *dataset*, menginisialisasi partikel, mencari *P-best* dan *G-best* partikel inisial, melakukan proses *HPSO-LS* kemudian menghasilkan *output* fitur-fitur terpilih.
- Studi literatur terhadap proses-proses yang ada pada rancangan desain perangkat lunak.
  Setiap proses-proses diperlukan pemahaman yang baik untuk menghindari *error*. Oleh karena itu, pada tahap ini perlu dilakukan studi literatur kembali. Studi dilakukan dengan mempelajari literatur didapatkan dari internet maupun dengan melakukan diskusi.
- Pembuatan perangkat lunak.
  Perangkat lunak dibuat/dibangun sesuai dengan urutan-urutan pada desain rancangan. Hal ini perlu dilakukan karena setiap proses bergantung pada proses sebelumnya.
- Evaluasi tiap-tiap bagian proses dari perangkat lunak yang telah dibangun.
  Evaluasi dilakukan dengan melakukan uji coba pada tiap-tiap proses perangkat lunak. Setiap proses dilihat tingkat keberhasilannya apakah sesuai dengan kriteria *output* proses yang sudah ditentukan.
- D*ebugging* terhadap perangkat lunak yang telah dibangun.
  Tahap ini dilakukan untuk mengetahui atau memeriksa adanya *bug* pada perangkat lunak. Jika ditemukan *bug*, maka segera diperbaiki. Tahap ini

dilakukan ketika perangkat lunak sudah selesai dibuat.

4. Uji coba dan evaluasi.
   Tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul. Parameter yang diuji coba adalah *dataset input (glass, wine* dan *heart),* jumlah partikel (*NP*), nilai *k* pada *k-NN* dan nilai *subset* fitur (*sf*) yang ingin diambil. Skenario uji cobanya adalah sebagai berikut:
   - Seleksi fitur menggunakan algoritma *PSO* pada masing-masing dataset.
     Skenario ini menghasilkan fitur-fitur terpilih serta akurasi klasifikasi data *k-NN* jika menggunakan fitur-fitur terpilih tersebut.
   - Seleksi fitur menggunakan algoritma *HPSO-LS* pada masing-masing dataset.
     Skenario ini juga menghasilkan fitur-fitur terpilih serta akurasi klasifikasi data *k-NN* jika menggunakan fitur-fitur terpilih tersebut.
   - Proses mengambil kombinasi fitur-fitur terpilih algoritma *PSO* dan *HPSO-LS* yang memiliki performa akurasi klasifikasi data *k-NN* terbaik.
   - Skenario terakhir membandingkan hasil akurasi klasifikasi data *k-NN* menggunakan fitur-fitur terpilih hasil seleksi fitur *PSO*, *HPSO-LS* dan mengunakan fitur keseluruhan (tanpa seleksi fitur).

   Evaluasi dilakukan dengan membandingkan hasil akurasi dari klasifikasi data menggunakan fitur-fitur terpilih hasil algortima *PSO* dan *HPSO-LS* maupun menggunakan fitur keseluruhan. Evaluasi tingkat keberhasilan dilihat dari pemilihan fitur-fiturnya. Pemilihan fitur-fitur tersebut mengacu pada nilai korelasi fitur yang sudah dihitung. Jika kombinasi fitur-fitur yang terpilih memiliki nilai korelasi yang rendah, maka nilai akurasi klasifikasi datanya akan

lebih tinggi dibandingkan fitur-fitur terpilih dengan nilai korelasi tinggi.

5. Penyusunan laporan tugas akhir.
   Tahap ini dilakukan penyusunan laporan pengerjaan tugas akhir yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil yang diperoleh selama pengerjaan tugas akhir. Penyusunan ini sangat terbantu dengan melakukan bimbingan terhadap dosen pembimbing maupun diskusi dengan teman. Penyusunan juga terbantu dengan melakukan pelaporan kepada dosen penguji. Masukkan-masukkan sangat dibutuhkan oleh penulis dalam menyusun laporan. Panduan penulisan buku tugas akhir juga sangat dibutuhkan untuk penyusunan buku ini. Penyusunan laporan dilakukan selama kurang lebih 3 bulan.
   Secara garis besar , buku tugas akhir ini terdiri atas beberapa bagian seperti berikut:
   BAB I Pendahuluan
   Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan penelitian, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan penelitian.
   BAB II Tinjauan Pustaka
   Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan penelitian ini.
   BAB III Desain Perangkat Lunak
   Bab ini berisi desain perangkat lunak secara umum. Dan juga pembahasan masing-masing proses dari perangkat lunak.
   BAB IV Implementasi
   Bab ini berisi implementasi dari perancangan perangkat lunak.
   BAB V Uji Coba dan Evaluasi
   Bab ini membahas pengujian seleksi fitur menggunakan algortima *PSO* dan *HPSO-LS.* Bab ini berisi hasil uji coba

kedua algortima dan evaluasi terhadap kedua algortima tersebut.

BAB VI Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan dari hasil pengujian yang dilakukan. Bab ini juga berisi saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Bagian ini berisi daftar referensi yang digunakan untuk membangun perangkat lunak ini.

Biodata Penulis

Bagian ini berisi biodata dari penulis buku.

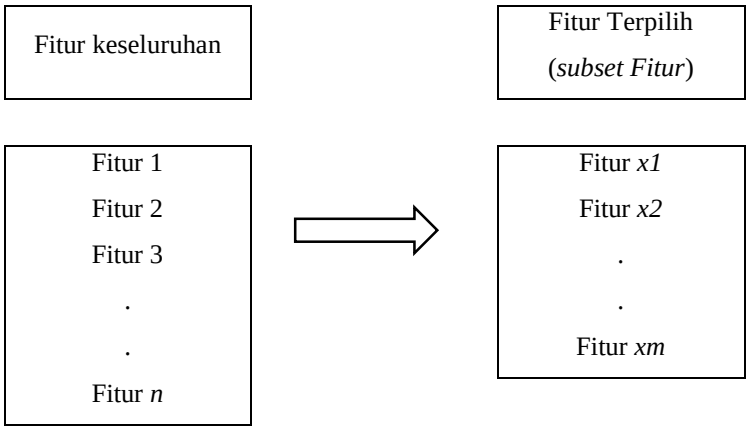
[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini membahas tentang teori-teori yang menjadi dasar dari pembuatan tugas akhir.

2.1 Seleksi Fitur

Seleksi fitur adalah proses mengurangi jumlah fitur dengan menghilangkan fitur yang tidak relevan atau dengan memilih fitur yang paling relevan/ informatif untuk meningkatkan proses klasifikasi. Manfaat dari seleksi fitur adalah mengurangi jumlah data yang dibutuhkan untuk *learning process*, meningkatkan waktu komputasi dan meningkatkan kecepatan dan akurasi klasifikasi.



Gambar 2.1 Ilustrasi Seleksi Fitur.

Seleksi fitur memilih *subset* fitur dari fitur keseluruhan untuk meningkatkan performa klasifikasi. Seleksi fitur dapat mengurangi kompleksitas dari model klasifikasi tanpa mengurangi akurasi dari *classifier* [11]. Seleksi fitur adalah

dengan memilih *m* fitur dari *n* fitur keseluruhan atau fitur asli dari *dataset (m < n)* [11]. Gambar 2.1 menunjukkan ilustrasi proses seleksi fitur. Dimana dari fitur keselurhan pada sisi kiri menjadi *subset* fitur disebelah kanan.

Beberapa metode yang digunakan untuk melakukan seleksi fitur diantaranya yaitu *filter*, *wrapper*, dan *hybrid* [1]. Metode *filter* menerapkan beberapa analisis statistik pada satu set fitur untuk menyelesaikan permasalahan seleksi fitur tanpa menggunakan model pembelajaran (*learning algorithm*). Diantaranya *chi-square test, information gain* dan *correlation coefficient* [1]. Oleh karena itu, metode ini biasanya tidak membutuhkan waktu komputasi yang banyak.

Metode *wrapper* menggunakan algoritma pembelajaran untuk melakukan evaluasi terhadap subset fitur melalui proses pencarian. Dengan kata lain, metode *wrapper* adalah proses pencarian berulang dimana hasil dari algortima pembelajaran pada setiap literasi digunakan untuk memandu proses pencarian [1].

Sedangkan metode *hybrid* adalah kombinasi dari metode *filter* dan *wrapper* [1]. Metode *hybrid* berfokus kepada penggabungan *filter* dan *wrapper* untuk mencapai performa terbaik yaitu dengan menggunakan algoritma pembelajaran tertentu dengan waktu kompleksitas mirip dengan metode *filter* [1].

2.2 *Particle Swarm Optimization*

Particle Swarm Optimization (*PSO*) adalah algoritma berbasis kecerdasan buatan yang digunakan untuk menyelesaikan persoalan optimasi dengan mencari solusi terbaik dari suatu permasalahan. Metode ini terinspirasi dari perilaku gerakan kawanan hewan seperti ikan, hewan *herbivore* dan burung yang kemudian tiap objek hewan disederhanakan menjadi sebuah partikel [1]. Suatu partikel dalam ruang memiliki posisi yang dikodekan sebagai vektor koordinat.

Vektor posisi ini dianggap sebagai keadaan yang sedang ditempati oleh suatu partikel di ruang pencarian. Dan setiap partikel tersebut bergerak dengan kecepatan v.

Setiap posisi dalam ruang pencarian merupakan alternatif solusi. Jika suatu partikel memiliki kecepatan yang konstan maka jika jejak posisi suatu partikel divisualisasikan akan membentuk garis lurus [1]. Namun posisi partikel tersebut dapat diarahkan (di *update*) ke titik optimal (solusi terbaik) dengan beberapa faktor. Faktor tersebut adalah posisi terbaik yang pernah dikunjungi partikel (*local best position*) dan posisi terbaik seluruh partikel (*global best position*). Hal tersebut dapat dievalusi menggunakan persamaan sebagai berikut:

$$v_{id}(t+1) = v_{id}(t) + c_1.r_1.\big(Pbest - x_{id}(t)\big)v_{id}(t) + c_2.r_2.\big(Gbest - x_{id}(t)\big), \quad (2.1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \quad (2.2)$$

Dimana:

i	= Partikel ke-*i*.
d	= Dimensi.
$v_i(t)$	= Kecepatan partikel.
$v_{id}(t)$	= Kecepatan partikel di-*d*.
c_1	= *Learning factor*.
c_2	= *Learning factor*.
r_1	= *Random number*.
r_2	= *Random number*.
Pbest	= *Local best position*.
Gbest	= *Global best position*.
$x_{id}(t)$	= Posisi dari *i-th* partikel di-*d*.

Pencarian titik optimum (*global best position*) dilakukan dengan cara melakukan *update* posisi sebanyak

literasi yang ditentukan. Dalam konteks data multidimensi, sistem *PSO* diinisialisai oleh sebuah populasi solusi secara acak yang tiap partikelnya direpresentasikan dengan vektor biner. Panjang dari vektor biner tersebut sama dengan jumlah fitur.

Kecepatan partikel dihitung sesuai dengan persamaan 1. Jika kecepatan melebihi v_{max} dan kurang dari v_{min} maka kecepatan partikel (v_{id}) dibatasi dengan menggunakan persamaan sebagai berikut [1] :

$$if\ V_{id}(t+1)\ \epsilon\ (V_{min}, V_{max})\ else\ V_{id}(t+1) = max\big(min\big(V_{max}, V_{id}(+1)\big), V_{min}\big), \quad (2.3)$$

v_{max} = 4,

v_{min} = -4.

Kemudian posisi/fitur di *update* dengan persamaan sebagai berikut [1]:

$$S\big(v_{id}(t+1)\big) = \frac{1}{1+e^{-v_{id}}},$$

$$if\ rand < S\big(v_{id}(t+1)\big)\ then\ x_{id}(t+1) = 1,$$

$$else\ x_{id}(t+1) = 0. \quad (2.4)$$

Posisi partikel setelah di-*update*, dihitung dengan fungsi $S\big(v_{id}(t+1)\big)$ (Persamaan (2.4)). Jika $V_{id}(t+1)$ lebih besar dari nilai acak, maka nilai posisinya direpresentasikan dengan nilai 1 (artinya fitur yang sesuai dipilih untuk pembaruan/*update* berikutnya). Di sisi lain, jika $V_{id}(t+1)$ lebih kecil dari nilai acak, maka nilai posisinya direpresentasikan dengan nilai 0 yang berarti bahwa fitur yang sesuai tidak dipilih untuk pembaruan/*update* berikutnya [1].

2.3 *Pearson Coefficient Correlation*

Koefisien korelasi *Pearson* digunakan untuk menghitung nilai korelasi antara fitur-fitur yang dimiliki oleh

partikel [1]. Perhitungan koefisien korelasi menggunakan persamaan sebagai berikut :

$$c_{ij} = \frac{\sum_{k=1}^{m}(x_i(k) - \bar{x}_i)(x_j(k) - \bar{x}_j)}{\sqrt{\sum_{k=1}^{m}(x_i(k) - \bar{x}_i)^2}\sqrt{\sum_{k=1}^{m}(x_i(k) - \bar{x}_i)^2}} \quad . \qquad (2.5)$$

Dimana:

c_{ij} = Koefisien korelasi antara dua fitur *i* dan *j*.
m = Jumlah data.
$x_i(k)$ = Nilai fitur vektor *i* pada data ke *k*.
$x_j(k)$ = Nilai fitur vektor *j* pada data ke *k*.
$\bar{x}_i$ = Nilai rata-rata dari keseluruhan x_i data.
$\bar{x}_j$ = Nilai rata-rata dari keseluruhan x_j data.

Setelah menghitung koefisien korelasi untuk semua kombinasi fitur, nilai korelasi untuk fitur *i* dihitung dengan persamaan sebagai berikut:

$$cor_i = \frac{\sum_{j=1}^{f}|c_{ij}|}{f-1} \; if \;\; i \neq j. \qquad (2.6)$$

Dimana:

f = Jumlah fitur.
c_{ij} = Koefisien korelasi Pearson antara fitur j dan j.

Jika nilai korelasi antara fitur tinggi, maka fitur tersebut memiliki hubungan yang tinggi dengan fitur lain. Sebaliknya, jika nilai korelasinya rendah, maka hubungannya rendah [1].

2.4 *Local Search Operation*

Local Search Operation adalah suatu proses untuk memilih fitur yang memiliki hubungan (korelasi) yang rendah dengan fitur lain [1]. Operasi ini diawali dengan pembagian fitur dari *dataset* menjadi 2 grup, yaitu *dissimilar (D)* dan *similar (S)*. *Disimilar* fitur berisi fitur-fitur yang memiliki nilai korelasi yang rendah. Sedangkan *similar* fitur memiliki nilai korelasi yang tinggi. Kemudian fitur diurutkan dari nilai korelasi terendah ke yang tertinggi.

Operasi *local search* dilanjutkan dengan 2 proses berikut [1], diantaranya:

1. *Feature segmentation.*
   Langkah-langkah *feature segmentation*:
   a. Partikel direpresentasikan dengan vektor biner dengan panjang sesuai dengan jumlah fitur.
   b. Memilih nilai 1 pada partikel.
   c. Membangkitkan partikel baru (*X*) berisi fitur yang terpilih dari proses sebelumnya.
   d. Membagi partikel *X* menjadi *Xd* dan *Xs*:
      - *Xd* berisi fitur yang ada di *D.*
      - *Xs* berisi fitur yang ada di *S.*
   e. Mengurutkan fitur di masing-masing *Xd* dan *Xs* secara *ascending* sesuai dengan nilai korelasi.
2. *Particle movement.*
   Setelah melakukan proses *feature selection*, dilakukan proses *particle movement*, yaitu proses untuk mengontrol nilai *bit* 1 atau fitur terpilih pada posisi terbaru. Proses ini menerapkan operasi *add* dan *delete*.
   Langkah-langkahnya sebagai berikut:
   a. Menghitung nilai n_d dan n_s dengan persamaan sebagai berikut:

$$n_d = (1 - \alpha).sf, \tag{2.7}$$

$$n_s = \alpha.sf. \tag{2.8}$$

Dimana:

n_d = Nilai untuk mengontrol 1-*bits* di *Xd.*
n_s = Nilai untuk mengontrol 1-*bits* di *Xs.*
α = *User specific parameter*.
sf = Jumlah subset fitur.

Nilai α ditetapkan sebesar 0.65. dan *sf* adalah jumlah subset fitur yang diinginkan atau dengan kata lain jumlah fitur yang ingin dipilih.

b. Mengecek fitur di *Xd*, langkah-langkahnya sebagai berikut:
   a. Jika jumlah $Xd < n_d$ maka:
      $(n_d - Xd)$ fitur di $(D - Xd)$ ditambahkan (*add*) di *Xd*.
   b. Sebaliknya jika $Xd > n_d$ maka:
      $(Xd - n_d)$ fitur di *Xd* dihapus (*delete*).
c. Mengecek fitur di *Xs*, langkah-langkahnya sebagai berikut:
   a. Jika jumlah $Xs > n_s$ maka:
      $(Xs - n_s)$ fitur di *Xs* dihapus (*delete*).
   b. Sebaliknya jika $Xs < n_s$ maka:
      $(n_s - Xs)$ fitur di $(S - Xs)$ ditambahkan (*add*) di *Xs.*
d. Setelah melakukan proses b dan c, kemudian digabungkan kembali menjadi partikel baru $X = Xd + Xs$ [1].

2.5 *Linear Normalization*

Linear normalization digunakan untuk membantu meningkatkan akurasi klasifikasi data. Setiap fitur di normalisasi dengan skala antara -1 dan 1 [1]. Persamaan yang digunakan sebagai berikut:

$$x^{new} = l + \left[(u - l) * \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right)\right]. \qquad (2.9)$$

Dimana:

l = *Lower bound.*
u = *Upper bound.*
x_{max} = Nilai maksimum *x*.
x_{max} = Nilai minimum *x*.

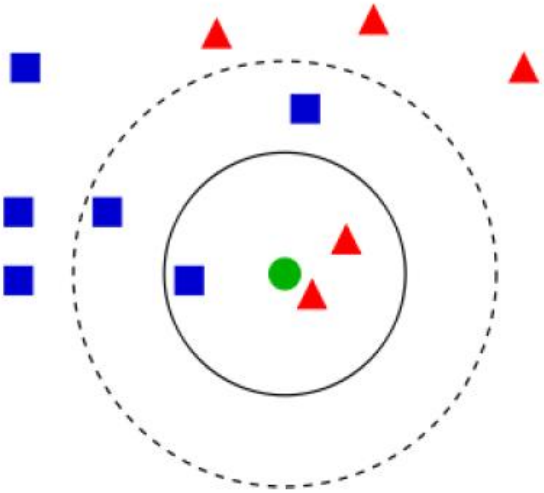
2.6 *k-Nearest Neighbor*

Algoritma *k-nearest neighbor* (*k-NN*) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan dari data pembelajaran yang jaraknya paling dekat dengan objek tersebut [3]. *K-NN* merupakan algoritma *supervised learning* dimana hasil dari *instance* yang baru (*data test*), diklasifikan berdasarkan mayoritas dari kategori pada algoritma *k-NN*. Kelas yang paling banyak muncul yang nantinya akan menjadi kelas hasil dari klasifikasi [3].

Algoritma ini bertujuan mengklasifikasikan objek baru berdasarkan atribut dan *training sample.* Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur/atribut dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas *c* jika kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat titk tersebut, dimana *k* adalah bilangan *integer* positif.

Algoritma *K-Nearest Neighbor* merupakan sebuah algoritma yang sering digunakan untuk klasifikasi teks dan data. Penggunaan *K-Nearest Neighbor* mempunyai sifat *self-learning* dimana jika semakin banyak dokumen, maka semakin banyak pula sumber yang dapat digunakan untuk dibandingkan. *K-Nearest Neighbor* berarti mencari tetangga yang paling dekat dengan sets yang akan di klasifikasi [12].

Contoh:



Gambar 2.2 Ilustrasi *k-NN* dengan 2 (dua) *Neighbor*.

- Dari Gambar 2.2 di atas dilihat bahwa data uji *d* (berbentuk lingkaran) diharapkan memiliki kelas yang sama dengan data *training* yang ada di sekitarnya.
- Probabilitas dari Gambar 2.2 di atas adalah sebagai berikut :
  - P(segitiga dalam lingkaran|segitiga) = 2/3.
  - P(biru dalam lingkaran|kotak) = 1/3.
- Penggunaan nilai *k* sebagai nilai jumlah kluster biasanya menggunakan *k* dengan nilai ganjil, misal *k*=1 *k*=3 *k*=5.
- Penghitungan *distance* yang dicari akan menggunakan *Euclidean Distance* [12].
- Bobot *vote K-Nearest Neighbor* diukur menggunakan *majority vote* atau *cosine* [12].

Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak *euclidean* [3]. Persamaannya sebagai berikut:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}\,. \qquad (2.10)$$

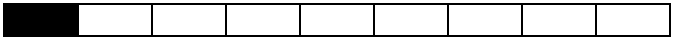
Dimana:

d = Jarak *Euclidean.*
x = Nilai atribut/fitur data *training.*
y = Nilai atribut/fitur data *testing.*

2.7 *k-Fold Cross Validation*

Cross Validation merupakan salah satu teknik untuk menilai/memvalidasi keakuratan sebuah model yang dibangun berdasarkan *dataset* tertentu. Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang boleh jadi belum pernah muncul di dalam *dataset*. Data yang digunakan dalam proses pembangunan model disebut data latih/*training*, sedangkan data yang akan digunakan untuk memvalidasi model disebut sebagai data *test* [2].

K-Fold Cross Validation membagi *dataset* menjadi sejumlah *k*-buah partisi secara acak. Kemudian dilakukan sejumlah *k*-kali eksperimen, dimana masing-masing eksperimen menggunakan data partisi ke-k sebagai data *testing* dan memanfaatkan sisa partisi lainnya sebagai data *training* [2].



Gambar 2.3 Ilustrasi *k-fold Cross Validation* pada Data.

Gambar 2.3 merupakan ilustrasi *k-fold Cross Validation* pada data. Data tersebut dibagi menjadi 9 bagian dimana salah satu bagian akan dijadikan sebagai data *testing*, dan sisanya sebagai data *training*.

2.8 *Confussion Matrix*

Confussion matrix (Kohavi and Provost, 1998) berisi informasi tentang klasifikasi aktual dan prediksi yang dilakukan oleh sistem klasifikasi [4]. *Confusion matrix* adalah sebuah tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan.. Tabel 2.1 berikut menunjukkan c*onfussion matrix* untuk pengklasifikasi dua kelas [4]. Berikut keterangan dari Tabel 2.1:

- A adalah jumlah hasil prediksi benar dengan kelas asli negatif.
- B adalah jumlah hasil prediksi salah dengan kelas asli negatif.
- C adalah jumlah hasil prediksi salah dengan kelas asli positif.
- D adalah jumlah hasil prediksi benar dengan kelas asli positif.

Tabel 2.1 *Confussion Matrix.*

		Prediksi	
		Negatif	Positif
Aktual	Negatif	A	B
	Positif	C	D

Evaluasi kinerja suatu model klasifikasi dapat dilakukan dengan menghitung nilai akurasi dan *error rate* dengan persamaan sebagai berikut.

$$akurasi = \frac{A + D}{A + B + C + D} \tag{2.11}$$

$$error\ rate = \frac{B + C}{A + B + C + D} \tag{2.12}$$

[Halaman ini sengaja dikosongkan]

BAB III
DESAIN PERANGKAT LUNAK

Bab ini menjelaskan tentang rancangan sistem perangkat lunak yang akan dibangun. Perancangan yang dijelaskan meliputi *dataset* yang digunakan sebagai uji coba dan proses-proses yang ada di perangkat lunak.

3.1 Deskripsi *Dataset*

Pada sub bab ini akan dijelaskan mengenai *dataset* yang digunakan sebagai masukan perangkat lunak untuk diuji. *Dataset* yang digunakan diantaranya *glass, wine* dan *heart. Dataset* didapat dari *website UCI Machine Learning Repository.* Setiap *dataset* memiliki nilai yang bermacam - macam dan range yang berbeda. Penjelasan tentang fitur atau atribut dan class setiap *dataset* dapat dilihat pada Tabel 3.2 - Tabel 3.7. Tabel 3.1 menunjukkan jumlah fitur, jumlah data (*instances*) dan jumlah *class. Dataset glass* memiliki 9 jumlah fitur, 214 jumlah *instances* dan 7 jumlah *class. Dataset wine* memiliki 13 jumlah fitur, 175 jumlah *instances* dan 3 jumlah *class. Dataset heart* memiliki 13 jumlah fitur, 270 jumlah *instances* dan 2 jumlah *class.*

Tabel 3.1 *Dataset.*

No	**Nama *Dataset***	**Jumlah Fitur**	**Jumlah *Instances***	**Jumlah *class***
1	*Glass*	9	214	7
2	*Wine*	13	175	3
3.	*Heart*	13	270	2

Tabel 3.2 *Dataset Glass.*

No	**Nama Fitur**	**Keterangan**	**Tipe Data**	**Range Nilai**
1	RI	*refractive index*	*Real*	1.5112 - 1.5339
2	Na	*Sodium*	*Real*	10.73 - 17.38
3	Mg	*Magnesium*	*Real*	0 - 4.49
4	Al	*Aluminum*	*Real*	0.29 - 3.5
5	Si	*Silicon*	*Real*	69.81 - 75.41
6	K	*Potassium*	*Real*	0 - 6.21
7	Ca	*Calcium*	*Real*	5.43 - 16.19
8	Ba	*Barium*	*Real*	0 - 3.15
9	Fe	*Iron*	*Real*	0 - 0.51

Tabel 3.3 Data *Class Glass.*

No	**Representasi pada *Dataset***	***Class***
1	1	*building windows float processed*
2	2	*building windows non float processed*
3	3	*vehicle windows float processed*
4	4	*vehicle windows non float processed*
5	5	*containers*
6	6	*tableware*
7	7	*headlamps*

Tabel 3.4 *Dataset Wine*

No	**Keterangan Fitur**	**Tipe Data**	**Range Nilai**
1	Alcohol	Real	11.030 - 14.8300
2	Malic acid	Real	0.7400 - 5.80000
3	Ash	Real	1.3600 - 3.23000
4	Alcalinity of ash	Real	10.600 – 30
5	Magnesium	Real	70 – 162
6	Total phenols	Real	0.9800 - 3.8800
7	Flavanoids	Real	0.3400 - 5.0800
8	Nonflavanoid phenols	Real	0.1300 - 0.6600
9	Proanthocyanins	Real	0.4100 - 3.5800
10	Color intensity	Real	1.2800 – 13
11	Hue	Real	0.4800 - 1.7100
12	OD280/OD315 of diluted wines	Real	1.2700 – 4
13	Proline	Real	11.030 – 1680

Tabel 3.5 Data *Class Wine*

No	**Representasi pada *Dataset***	***Class***
1	1	*Class 1*
2	2	*Class 2*
3	3	*Class 3*

Dataset yang digunakan merupakan *dataset* yang berisi nilai-nilai atau berisi bilangan. Sistem ini hanya melakukan seleksi fitur dengan data-data yang bertipe bilangan. Bilangan-bilangan pada *dataset* ini terdiri dari bilangan *real.*

Tabel 3.6 *Dataset Heart*

No	Keterangan Fitur	Tipe Data	Range Nilai
1	*Age*	*Real*	29 – 77
2	*Sex*	*Binary*	0 – 1
3	*chest pain type*	*Nominal*	1 – 4
4	*Resting blood pressure*	*Real*	94 – 200
5	*Serum cholestoral (mg/dl)*	*Real*	126 – 564
6	*Fasting blood sugar > 120 mg/dl*	*Binary*	0 – 1
7	*Resting electrocardiographic results*	*Nominal*	0 - 2
8	*Maximum heart rate achieved*		71 – 202
9	*Exercise induced angina*	*Binary*	0 – 1
10	*Oldpeak*	*Real*	0 – 6.2
11	*The slope of the peak exercise ST segment*	*Ordered*	1 – 3
12	*Number of major vessels (0-3) colored by flourosopy*	*Real*	0 – 3
13	*Thal:* *3 = normal;* *6 = fixed defect;* *7 = reversable defect*	*Nominal*	3 - 7

Tabel 3.2 menunjukkan deskripsi dari *dataset glass.* Data tersebut diambil dari website *UCI Machine Learning Repository* dengan nama *Glass Identification Database.*

Tabel 3.7 Data *Class Heart.*

No	**Representasi pada *Dataset***	***Class***
1	1	Tidak Ada penyakit
2	2	Ada penyakit

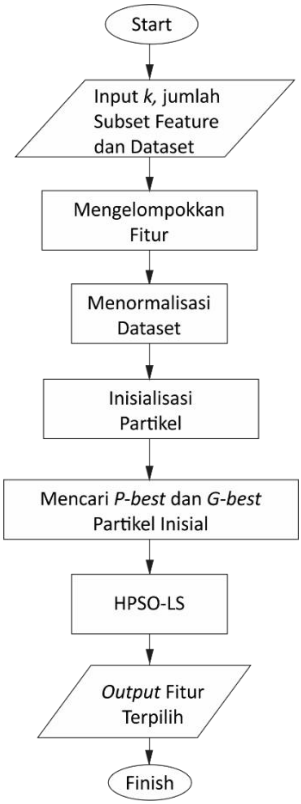
Dataset glass dibuat oleh *B. German*, dari *Central Research Establishment Home Office Forensic Science Service Aldermaston, Reading, England* pada September, 1987. *Vina Spiehler Ph.D.*, dari *Diagnostic Products Corporation* sebagai penyumbang. Tipe data dari *dataset* tersebut adalah bilangan *real* semua. Jumlah *class* pada *dataset glass* tersebut sebanyak 7. Tabel 3.3 menunjukkan *class-class* dari *dataset glass. Class-class* tersebut dikelompokkan berdasarkan fungsi dari *glass* itu sendiri.

Tabel 3.4 menunjukkan deskripsi dari *dataset wine. Data*set tersebut memiliki nama *Wine Recognition Data* pada *UCI Machine Learning Repository. Dataset* tersebut didapatkan dari *Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.* Tipe *dataset* tersebut sama dengan tipe *dataset glass*, yaitu bilangan *real.* Deskripsi kelas pada *dataset wine* (Tabel 3.5) tidak mencantumkan secara rinci pada sumbernya. Deskripsi kelas pada Tabel 3.5 hanya berupa penamaan *class 1, class 2* dan *class 3.*

Tabel 3.6 menunjukkan deskripsi dari *dataset heart.* Informasi yang dicantumkan tidak banyak pada *UCI Machine Learning Repository. Dataset* heart yang asli, memiliki 75 fitur. Namun dari sumber yang didapatkan, sudah di *extract* menjadi 13 fitur. Tipe data pada *dataset heart* bermacam-macam sesuai yang tertera pada Tabel 3.6. *Class-class* dikelompokkan berdasarkan ada tidaknya penyakit jantung. *Class-class* pada

dataset ini terdiri dari 2 *class*, *class* 1 tidak ada penyakit dan *class* 2 ada penyakit.

3.2 Desain Sistem Secara Umum



Gambar 3.1 *Flowchart* Sistem.

Gambar 3.1 menunjukkan proses-proses yang ada pada sistem. Sistem ini diawali dengan memasukkan (*input*) nilai *k* pada *k-NN*, jumlah *subset feature* (*Sf*) dan *dataset*. Kemudian

proses dilanjutkan dengan mengelompokkan fitur. Proses selanjutnya adalah melakukan normalisasi *dataset*, kemudian menginisialisasi partikel. Dilanjutkan dengan proses pencarian nilai *P-best* dan *G-best* partikel inisial tersebut. Setelah itu proses *HPSO-LS* dilakukan untuk melakukan seleksi fitur dan diakhiri dengan menghasilkan fitur-fitur terpilih hasil *HPSO-LS.* Sistem ini berupa *script* berformat .m yang dijalankan pada program matlab. Subbab-subbab dibawah ini akan menjelaskan secara detail dari proses-proses tersebut.

3.3 *Input k (k-NN),* Jumlah Subset Fitur (*Sf*) dan *Dataset*

Input dari program ini adalah nilai *k*, subset fitur dan *dataset*. Nilai *k* adalah nilai tetangga/*neighbor* dari algortima klasifikasi *k-NN. K* bernilai positif *integer*. Subset fitur *(sf)* adalah jumlah fitur yang ingin dipilih. *Sf* bernilai positif *integer*. *Dataset* yang dimasukkan berupa data multidimensional m x n, dimana n-1 menunjukkan jumlah fitur atau atribut dari *dataset* dan sisanya menunjukkan *class.* M dan n adalah bilangan positif *integer. Dataset* yang dimasukkan adalah *dataset* yang sudah disebutkan pada subbab 3.1.

3.4 Mengelompokkan Fitur

Fitur-fitur dari *dataset* dikelompokkan sesuai dengan perhitungan nilai korelasi antar fitur. Perhitungan menggunakan persamaan *pearson correlation coefficient* yaitu persamaan 2.5 dan persamaan 2.6. Persamaan 2.5 digunakan untuk menghitung koefisien korelasi semua fitur pada *dataset*. Persamaan 2.5 menghasilkan nilai koefisien korelasi antara fitur satu dengan fitur lainnya. Sedangkan persamaan 2.6 digunakan untuk menghitung nilai korelasi. Persamaan 2.6 menghasilkan nilai korelasi antar fitur. Masing-masing fitur mempunyai nilai korelasi terhadap fitur lainnya. Setelah proses perhitungan nilai korelasi, kemudian fitur dikelompokkan menjadi 2 bagian

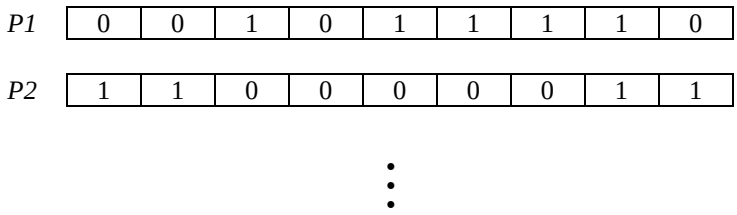
yaitu, *dissimilar (D)* dan *similar (S). Dissimilar* berisi fitur-fitur yang memiliki nilai korelasi yang rendah, sedangkan kelompok *similar* memiliki fitur-fitur dengan korelasi yang tinggi. Pembagian ini dilakukan dengan mengurutkan nilai korelasi dari yang terkecil hingga yang paling besar. Setelah itu, fitur-fitur terurut tersebut dikelompokkan menjadi 2 bagian. Hasil penjumlahan *D* dan *S* merupakan jumlah fitur keseluruhan.

3.5 Menormalisai *Dataset*

Dataset dinormalisai menggunakan *linear normalization* dengan persamaan 2.9. Normalisasi dilakukan dengan membatasi nilai-nilai fitur dengan batas minimum -1 dan batas maksimal 1. Proses ini bertujuan untuk memaksimalkan proses komputasi agar lebih cepat.

3.6 Inisialisasi Partikel

Partikel direpresentasikan dengan vektor biner (*bit of array*) yang dibangkitkan secara *random*. Kemudian untuk kecepatan partikel (*v*) dibangkitkan dengan bilangan *float random.* Panjang vektor keduanya sama dengan panjang vektor partikel. Setiap data (*instance*) pada *dataset* merupakan partikel. Vektor biner pada partikel menunjukkan posisi dari partikel. Setiap bilangan biner pada vektor partikel juga menunjukkan fitur dari *dataset*. Untuk lebih jelasnya, dipaparkan Gambar 3.2.



Pn	0	0	1	0	1	0	1	0	1

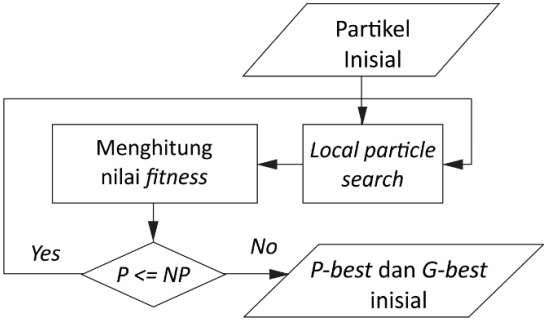
Gambar 3.2 Ilustrasi Partikel.

P1, P2, Pn = Partikel.
n = Jumlah data (*instance*).

Jumlah partikel yang digunakan (*NP*) adalah 20 partikel, *n* = 20. Jumlah partikel tersebut mengacu pada *paper* yang didapatkan.

3.7 Mencari P-best dan G-best Partikel Inisial

Proses pencarian *P-best* dan *G-best* dilakukan dengan 2 proses, yaitu *local particle search* dan menghitung nilai *fitness*. Proses diawali dengan melakukan proses *local particle search* kemudian dilanjutkan dengan menghitung nilai *fitness* masing-masing partikel. Alur prosesnya dijelaskan pada *flowchart* Gambar 3.3.



Gambar 3.3 *Flowchart* Mencari *P-best* dan *G-best* Partikel Inisial.

Proses *local particle search* menghasilkan partikel baru berdasarkan nilai korelasi fitur yang sudah dihitung. Proses menghitung nilai *fitness* ini adalah untuk megetahui sebaik mana fitur-fitur terpilih berdasarkan nilai-nilai bit 1 pada

partikel tersebut. Proses-proses tersebut akan dijelaskan lebih detail pada subbab 3.7.1 dan subbab 3.7.2.

3.7.1 Local Particle Search

Local particle search adalah proses untuk memilih *bits* 1 pada partikel yang memiliki nilai korelasi rendah. Dan menghilangkan nilai *bits* 1 pada partikel yang memiliki nilai korelasi tinggi. Setiap baris pada *dataset* menunjukkan sebuah partikel. Jumlah partikel (*NP*) yang digunakan adalah 20 partikel. Proses ini dilakukan pada setiap partikel. Proses ini terdiri dari *feature segmentation* dan *particle movement. Feature segmentation* adalah proses memilih nilai *bit* 1 pada vektor partikel dan kemudian disusun kembali menjadi partikel baru (*X*) dengan berisikan indeks/nomor fitur. Kemudian dari partikel *X* dibagi menjadi 2 bagian *Xd* dan *Xs. Xd* berisi fitur-fitur *dissimilar (D)* dan *Xs* berisi fitur-fitur *similar* (*S*). Setelah itu dilakukan proses *particle movement* yaitu proses mengontrol nilai 1 pada partikel yang terpilih. Proses diawali dengan menghitung nilai n_s dan n_d. Kemudian mengecek fitur di *Xd*. Jika jumlah fitur di $Xd < n_d$ maka $(n_d - Xd)$ fitur di $(D - Xd)$ ditambahkan (*add*) di *Xd.* Sebaliknya jika jumlah fitur di $Xd > n_d$ maka $(Xd - n_d)$ fitur di *Xd* dihapus (*delete*). Setelah mengecek fitur di *Xd,* selanjutnya mengecek fitur di *Xs*. Jika jumlah fitur di $Xs > n_s$ maka $(Xs - n_s)$ fitur di *Xs* dihapus (*delete*). Sebaliknya jika jumlah fitur di $Xs < n_s$ maka $(n_s - Xs)$ fitur di $(S - Xs)$ ditambahkan (*add*) di *Xs.*

Setelah melakukan proses *particle movement*, kemudian *Xd* dan *Xs* digabungkan kembali menjadi partikel baru $X = Xd + Xs$. Untuk lebih jelasnya dipaparkan pada ilustrasi berikut, Gambar 3.4 merupakan ilustrasi partikel *P*.

P1	0	1	0	0	0	0	0	0	1

Gambar 3.4 Partikel *P*

a. Feature Segmentation

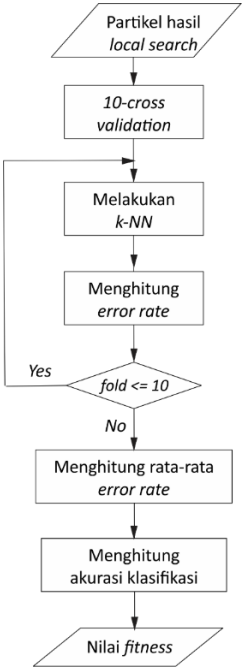
- Misalkan setelah melakukan perhitungan nilai korelasi antar fitur didapatkan:
  D = [3,5,2,6,7]
  S =[1,4,9,8]
- Kemudian memilih nilai *bit* 1 pada *P:*
  X = [2,9]
- Mengelompokkan fitur
  Xd = [2], Xs =[9]
  Jumlah fitur Xd = 1
  Jumlah fitur Xs = 1

b. Particle movement

- Menghitung nilai n_s dan n_d. Misalkan menggunakan Sf = 5, didapatkan n_d.= 2 dan n_s = 3
- Melakukan pengecekan fitur di *Xd.*
  Jumlah fitur $Xd < n_d$
  n_d - Xd = 2 – 1 = 1
  (D - Xd) = [3,5,6,7]
  Maka 1 fitur di *(D - Xd)* ditambahkan di *Xd*
  Xd = [2,3]
- Melakukan pengecekan fitur di *Xs*
  Jumlah fitur $Xs < n_s$
  $n_s - Xs$ = 3 – 1 = 2
  (S - Xs) = [1,4,8]
  Maka 2 fitur di *(S - Xs)* ditambahkan di *Xs*
  Xd = [9,1,4]
- X = Xd + Xs
  = [2,3,1,4,8]

Proses *local particle search* tersebut menghasilkan partikel baru berdasarkan nilai korelasi antar fitur yang telah dihitung. Ilustrasi diatas menunjukkan bahwa fitur dengan nilai korelasi yang rendah akan lebih diprioritaskan untuk dipilih. Hal tersebut menjadi kunci dari proses *local particle search* ini.

3.7.2 Menghitung nilai *fitness*



Gambar 3.5 *Flowchart* Menghitung Nilai *Fitness.*

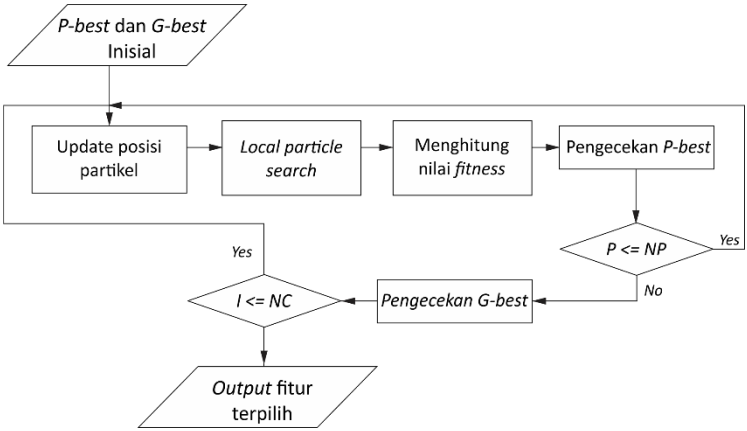
Perhitungan nilai *fitness* dilakukan untuk mengetahui/mengevaluasi sebaik mana kualitas fitur-fitur terpilih. Kualitas yang dimaksudkan disini adalah seberapa baik akurasi klasifikasi data menggunakan *k-NN* jika menggunakan fitur-fitur tersebut. Pemilihan fitur dilakukan dengan melihat nilai bit 1 pada partikel. Sebagai contoh missal, jika bit 1 berada pada fitur ke 2,3,4,5,7,9, maka fitur-fitur 2,3,4,5,7,9 lah yang akan dipilih untuk dihitung nilai *fitness*-nya. Proses pemilihan tersebut sama dengan proses yang ada pada *local particle search.* Namun sebelum melakukan *k-NN*, dilakukan proses *10-fold cross validation. Data training* dibagi menjadi 10 bagian,

dimana tiap-tiap bagian memiliki jumlah porsi yang sama dari masing-masing *class*-nya. 9 bagian pertama digunakan sebagai *training process* untuk membangun *learning model,* sedangkan sisanya untuk data *testing*. Perhitungan nilai *fitness* didapatkan dari nilai rata-rata *error rate* dari *10-fold cross validation* tersebut. Masing-masing data *test* yang di *testing* dihitung *error rate*-nya dengan *confusion matrix. Confusion matrix* menunjukkan jumlah prediksi benar dan prediksi salah terhadap klasifikasi data *k-NN.* Nilai-nilai jumlah prediksi salah dan benar tersebut dapat diketahui *error rate* dari klasifikasi data *k-NN.* Setiap *fold* memiliki *error rate* masing-masing. Kemudian proses perhitungan nilai akurasi dari klasifikasi tersebut dapat dilakukan setelah nilai rata-rata *error rate* dari seluruh *fold* didapatkan,. Nilai akurasi inilah yang dijadikan nilai *fitness* dari fitur-fitur terpilih tersebut.

Setelah proses perhitungan nilai *fitness* tersebut dilakukan, dipilih subset fitur (fitur terpilih) mana yang memiliki *error rate* terendah. Subset fitur dengan *error rate* terendahlah yang kemudian dijadikan *global best position* (*G-best)*. Proses ini menghasilkan partikel-partikel baru yang memiliki posisi terbaik masing-masing (*P-best)* dan posisi terbaik keseluruhan partikel (*G-best).* Setiap *P-best* disebut subset fitur terbaik *instance*/data itu sendiri. Dan *G-best* disebut subset fitur terbaik dari *dataset* keseluruhan. Posisi *P-best* dan *G-best* akan digunakan untuk proses *update* posisi partikel pada proses *HPSO-LS.*

3.8 *HPSO-LS*

Proses *HPSO-LS* diawali dengan meng-*update* posisi partikel menggunakan algortima *PSO.* NP (*Number of particle*) adalah jumlah partikel, yaitu 20. Kemudian proses dilanjutkan dengan melakukan *local particle search* dan menghitung nilai *fitness*-nya. Setelah itu proses dilanjutkan dengan melakukan pengecekan *P-best* partikel.



Gambar 3.6 *Flowchart HPSO-LS.*

Gambar 3.6 merupakan *flowchart*/aliran proses dari *HPSO-LS.* Proses-proses tersebut dilakukan ditiap-tiap partikel. Setelah mendapatkan *P-best* pada setiap partikel, proses pengecekan *G-best* dilakukan. *G-best* inilah yang merupakan posisi terbaik dari partikel-partkel tersebut. Pengecekan *G-best* dilakukan sebanyak iterasi (*NC*) yaitu 50. Jika sudah mencapai iterasi yang ditentukan, maka *G-best* terakhirlah yang merupakan *subset* fitur terpilih. Proses-proses pada *HPSO-LS* tersebut akan dijelaskan lebih detail pada subbab 3.8.1 sampai dengan subbab 3.8.4.

3.8.1 *Update* Posisi Partikel

Partikel mempunyai posisi dan kecepatan. Setiap partikel melakukan pergerakan menuju posisi yang lebih baik dari sebelumnya. Tentunya pada sistem ini bertujuan untuk mencari posisi terbaik dari partikel. Posisi terbaik menunjukkan fitur-fitur terbaik yang dapat dipilih. *Update* posisi partikel dilakukan dengan melakukan perhitungan sesuai dengan

algoritma *PSO* pada persamaan 2.1, 2.2, 2.3 dan 2.4 secara berurutan.

Proses pertama yaitu melakukan *update* kecepatan dari partikel sesuai dengan persamaan 2.1. Jika kecepatan melebihi v_{max} dan kurang dari v_{min} maka kecepatan partikel dibatasi dengan menggunaan persamaan 2.3. Setelah didapatkan posisi hasil dari persamaan 2.1, kemudian melakukan *update* posisi menggunakan persamaan 2.2. *Update* posisi berdasarkan nilai kecepatan yang dihitung pada proses sebelumnya. Jadi posisi partikel bergantung pada kecepatan. Kemudian setelah mendapatkan posisi, dilakukan perubahan (*update*) nilai posisi tersebut menjadi bilangan biner sesuai dengan persamaan 2.4. Jika nilai $S\left(v_{id}(t+1)\right)$ > bilangan acak, maka posisi akan bernilai 1. Sebaliknya, jika $S\left(v_{id}(t+1)\right)$ < bilangan acak, maka posisi akan bernilai 0. *Update* posisi partikel bergantung pada *local best position (P-best)* partikel dan *global best position (G-best).* Karena setiap partikel yang memiliki kecepatan, akan mengarah kepada partikel yang memiliki posisi yang lebih baik dari posisi partikel itu sendiri.

3.8.2 *Local Partcle Search* dan Menghitung Nilai *Fitness*

Proses *local particle search* dan proses perhitungan nilai *fitness* sama dengan proses pada subbab 3.7.1 dan 3.7.2. Bedanya jika pada subbab-subbab tersebut, partikel yang diolah/diproses adalah partikel inisial. Sedangkan pada proses *HPSO-LS*, partikel yang diproses adalah partikel hasil dari *update* posisi (subbab 3.8.1).

3.8.3 Pengecekan *P-best*

Suatu posisi dapat disebut sebagai *P-best* apabila partikel tersebut memiliki nilai *fitness* yang baik. Perhitungan nilai tersebut dilakukan dengan menghitung *error rate* klasifikasi data *k-NN* menggunakan fitur-fitur terpilih dari

partikel tersebut. Fitur-fitur terpilih ini didapatkan dari proses *local particle search*. Perhitungan nilai *fitness* sama dengan proses yang dijelaskan pada bab 3.7.2.

Pengecekan *P-best* dilakukan disetiap partikel. Karena setiap partikel pasti memiliki posisi terbaiknya masing-masing. Pengecekan ini dilakukan sebanyak iterasi yang ditentukan yaitu sebanyak 50 kali.

Pengecekan *P-best* dilakukan dengan membandingkan nilai *fitness (error rate)* antara posisi partikel sekarang dengan posisi partikel di iterasi sebelumnya. Jika pada posisi sekarang nilai *fitness*-nya lebih baik (*error rate* lebih kecil), maka posisi partikel sekarang menjadi *P-best*. Namun jika nilai *fitness*-nya tidak lebih baik, maka posisi *P-best* tetap.

P-best ini digunakan untuk melakukan pembaharuan*/update* kecepatan partikel sesuai dengan persamaan 2.1. Perhitungan kecepatan partikel tersebut kemudian digunakan untuk melakukan pembaharuan/*update* posisi partikel.

3.8.4 Pengecekan *G-best*

Setelah mendapatkan *P-best* dari masing-masing partikel, kemudian melakukan pengecekan terhadap *G-best. G-best* didapatkan dari nilai *fitness* terbaik (*error rate* terkecil) *P-best* seluruh partikel. Pengecekan *G-best* dilakukan dengan membandingkan nilai *fitness P-best* terbaik sekarang dengan *P-best* terbaik sebelumnya. Jika pada *P-best* terbaik sekarang nilai *fitness*-nya lebih baik, maka *P-best* terbaik sekarang menjadi *G-best*. Namun jika tidak lebih baik, maka *G-best* tetap. Pengecekan *G-best* dan *P-best* tersebut dilakukan secara kontinu sebanyak iterasi (*NC*) yang ditentukan yaitu 50 kali. Jika sudah dilakukan pengecekan sebanyak iterasi, maka *G-best* terakhirlah yang menjadi solusi terbaik. Solusi inilah yang merupakan fitur terpilih.

3.9 Output Fitur Terpilih

Setelah proses *HPSO-LS* dilakukan sebanyak iterasi yang ditentukan (*NC* = 50), maka didapatkan *Global best position (G-best)* yang terakhir didapatkan. *G-best* inilah yang menunjukkan subset fitur terpilih hasil seleksi fitur *HPSO-LS.*

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan menggunakan bahasa pemrograman Matlab.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Perancangan Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	*Processor Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz. 4.00 GB RAM.*
Perangkat lunak	Sistem operasi : *Windows 7 Ultimate 64-bit.* Perangkat pegembang : *MATLAB R2014a.*

4.2 Implementasi

Subbab ini menjelaskan tentang implementasi proses yang telah disebutkan pada bab desain perangkat lunak.

4.2.1 Input Jumlah Subset Fitur (*Sf*) dan *Dataset*

Subbab ini menjelaskan tentang proses *input Sf* dan *dataset*. *Sf* merupakan jumlah fitur yang ingin dipilih. Tahap awal sistem ini memerlukan nilai *Sf* untuk mengetahui berapa

jumlah fitur yang ingin dipilih. Implementasi ditunjukkan pada kode sumber dibawah ini.

```
1.  DATA = load('D:\TA\TUGAS
    AKHIR\dataset\glass.data');
2.  [ROW,COL] = size(DATA);
3.  DATASET   = DATA(:, 2:COL);
4.  [ROW,COL] = size(DATASET);
5.  Sf      = 1;
6.  ALPHA   = 0.65;
```

Kode Sumber 4.1 *Input Sf* dan *Dataset*.

Dataset awal yang diproses adalah data *glass*. Data *glass* mulanya memiliki 14 kolom, yang di kolom pertamanya merupakan nomor *instances*. Oleh karena itu, proses pertama dilakukan pengolahan data dengan melakukan pengurangan kolom pada kolom pertama. Kemudian *Sf* awal yang diuji coba bernilai 1. Batas masukkan nilai *Sf* ini adalah kurang dari fitur keseluruhan. Dan parameter α nantinya digunakan untuk menghitung nila n_s dan n_d yang ditetapkan sebesar 0.65.

4.2.2 Mengelompokkan Fitur

Fitur-fitur pada *dataset* dikelompokkan sesuai nilai korelasinya. Implementasi perhitungan korelasinya adalah:

```
1.  DATASET_COEF = DATASET(:,1:COL-1);
2.  CC           = corrcoef(DATASET_COEF);
3.  COR          = sum(CC);
4.  [COR_SORT, IDX_COR]     = sort(COR);
5.  MID_IDX     = round((COL-1)/2);
6.  DISS        = IDX_COR(1:MID_IDX);
7.  SIM         = IDX_COR(MID_IDX+1 :end);
```

Kode Sumber 4.2 Pengelompokan Fitur.

Fitur dikelompokkan menjadi 2 bagian, *dissimilar* dan *similar*. Perhitungan korelasi dilakukan dengan fungsi

`corrcoef` sesuai kode sumber 4.2. Perhitungan ini hanya dilakukan di kolom-kolom fitur *dataset*.

4.2.3 Inisialsisai Partikel

Inisialisasi partikel dilakukan dengan cara membangkitkan vektor bilangan biner *random* sesuai dengan ukuran *dataset*. Implementasinya sebagai berikut:

```
1. NP = 20;
2. X = randi([0 1],NP,COL-1);
3. V = rand(NP,COL-1);
```

Kode Sumber 4.3 Inisalisasi Partikel.

Jumlah partikel yang digunakan adalah 20 partikel. Setiap partikelnya mempunyai posisi dan kecepatan. Vektor posisi dibangkitkan dengan bilangan biner *random* dan vektor kecepatan dibangkitkan dengan bilangan *float random* dengan *range* [01]. Vektor posisi dan kecepatan memiliki ukuran yang sama dengan *dataset*.

4.2.4 *Local Particle Search*

Subbab ini menjelaskan proses *local particle search*. *Local particle search* bergantung pada nilai korelasi antar fitur. Implementasinya sesuai dengan Kode Sumber 4.4. Setiap partikel memiliki posisi terbaik (*P-best*), pada proses ini, setiap partikel inisialisasi menempati posisi terbaiknya. Proses *local particle search* diawali dengan menghitung nilai n_s dan n_d. Keduanya digunakan untuk mengontrol nilai *bit* 1 pada *Xs* dan *Xd*. Setiap partikel dipilih nilai *bit* 1 nya dan dikelompokkan menjadi 2 bagian *Xs* dan *Xd* sesuai dengan nilai korelasinya.

```
 1.  Xpb          = X;
 2.  Nd       = round((1-ALPHA)*Sf);
 3.  Ns       = round(ALPHA*Sf);
 4.  for row_Xpb = 1:NP
 5.   NN_DTEST = [];
 6.    feature = find(Xpb(row_Xpb,1:COL-1));
 7.    Xd = intersect(DISS,feature,'stable');
 8.    Xs = intersect(SIM,feature,'stable');
 9.    [R_Xd, S_Xd]       = size(Xd);
10.    [R_Xs, S_Xs]       = size(Xs);
11.    if S_Xd < Nd
12.        temp = Nd - S_Xd;
13.       fd_Xd = setdiff(DISS,Xd,'stable');
14.          Xd = [Xd fd_Xd(1,1:temp)];
15.    elseif S_Xd > Nd
16.        temp = S_Xd - Nd;
17.          Xd = Xd(1,1:(S_Xd-temp));
18.    else
19.          Xd = Xd;
20.    End
21.    if S_Xs > Ns
22.       temp    = S_Xs - Ns;
23.       Xs = Xs(1,1:(S_Xs-temp));
24.    elseif S_Xs < Ns
25.       temp   = Ns - S_Xs;
26.       fd_Xs  = setdiff(SIM,Xs,'stable');
27.       Xs = [Xs fd_Xs(1,1:temp)];
28.    else
29.       Xs = Xs;
30.    End
31.  End
32.  [LOSS,IDX] = min(Xpb(:,COL));
```

Kode Sumber 4.4 *Local Search Particle*.

Proses pembagian inilah yang disebut *feature segmentation.* Proses tersebut diimplementasikan pada baris kode sumber 6-8. Setelah membagi fitur menjadi 2 bagian, kemudian dilakukan proses *particle movement.* Proses ini dimulai dari baris kode sumber 15 hingga baris 30.

4.2.5 Menghitung nilai *fitness*

Perhitungan nilai *fitness* diimplementasikan sesuai kode sumber di bawah ini.

```
1.  for row_Xpb = 1:NP
2.    NN_DTEST = [];
3.    for n = 1:sum_feature
4.      NN_DTEST = [NN_DTEST  DATASET(1:ROW,
      feature(n))];
5.    end
6.    Mdl =
      fitcknn(NN_DTEST,NN_CLASS,'NumNeighbors',k
      );
7.    cvmdl = crossval(Mdl);
8.    Xpb(row_Xpb,COL) = kfoldLoss(cvmdl);
9.  end
10. [LOSS,IDX] = min(Xpb(:,COL));
```

Kode Sumber 4.5 Menghitung Nilai *Fitness.*

Perhitungan nilai *fitness* dilakukan dengan menghitung akurasi klasifikasi data menggunakan *k-NN*, menggunakan fitur-fitur terpilih. Nilai k didapatkan dari proses *input*-an diawal. Sistem ini menggunakan fungsi `fitcknn` untuk membuat model *k-NN*. Model tersebut menggunakan beberapa parameter diantaranya data training (`NN_DTEST`), *class* (`NN_CLASS`) dari *dataset* dan jumlah *k* (`NumNeighbors`). Kemudian Proses 10-*cross validation* dilakukan menggunakan fungsi `crossval`. Kemudian proses dilanjutkan dengan menghitung nilai rata-rata *loss* atau nilai misklasifikasi (*error rate*) dari model 10-*cross validation* tersebut, menggunakan fungsi `kfoldLoss`. Nilai misklasifikasi menunjukkan nilai *fitness* dari fitur-fitur terpilih. Semakin kecil nilai misklasifikasinya, maka akurasinya semakin baik/besar. Dan nilai akurasi yang terbaik inilah yang dijadikan sebagai *G-best* dari kawanan/keseluruhan partikel.

4.2.6 Update Posisi Partikel

Proses update partikel menggunkan algortima *PSO* sesuai dengan persamaan 1,2 dan 3. Implementasinya sebagai berikut.

```
1.  R1 = rand();
2.  R2 = rand();
3.  Vid(a,b) = V(a,b) + ( C1*R1*( Xpb(a,b) -
    X(a,b) )*V(a,b) ) + ( C2*R2*( Xgb(1,b) -
    X(a,b) ) );
4.  if (Vid(a,b) < Vmin) || (Vid(a,b) > Vmax)
5.    Vid(a,b) = max(min(Vmax,Vid(a,b)),Vmin);
6.  else
7.   S = (1/(1+exp(-Vid(a,b))));
8.   if rand() < S
9.      Xid(a,b) = 1;
10.  else
11.     Xid(a,b) = 0;
12.  end
13. end
```

Kode Sumber 4.6 *Update* Posisi Partikel.

Proses ini merupakan tahapan awal dari proses *HPSO-LS.* Setiap posisi dari partikel di *update* berdasarkan posisi terbaik sebelumnya dan posisi terbaik keseluruhan partikel. Iterasi pertama, *P-best* merupakan posisi terbaik partikel itu sendiri (*local best position*). *P-best* didapatkan dari perhitungan nilai *fitness.* Sedangkan *G-best* pada iterasi pertama, didapatkan dari *P-best* terbaik dari keseluruhan partikel inisial. Setelah melakukan *update* posisi, kemudian proses *local particle search* dilakukan. Proses *local particle search* ini nantinya akan menghasilkan partikel dengan posisi baru. Kemudian setelah proses *local particle search*, proses selanjutnya yaitu menghitung nilai *fitness.*

4.2.7 Pengecekan *P-best* dan *G-best*

P-best dan *G-best* didapatkan dari proses perhitungan nilai *fitness*. Pengecekan ini dilakukan untuk mengupdate posisi terbaik, baik *P-best* ataupun *G-best*, ke posisi yang lebih baik lagi. Implementasi kodenya sebagai berikut.

```
1.  if Xid(a,COL) < Xpb(a,COL)
2.     Xpb(a,:) = Xid(a,:);
3.  end
```

Kode Sumber 4.7 Pengecekan *P-best*

```
1.  [LOSS_NEW,IDX_NEW] = min(Xpb(:,COL));
2.  if Xpb(IDX_NEW,COL) < Xgb(1,COL)
3.     Xgb(1,:) = Xpb(IDX_NEW,:);
4.  End
```

Kode Sumber 4.8 Pengecekan *G-best.*

Proses pengecekan *P-best* dan *G-best* ini sudah diketahui sebelumnya. Pengecekan *P-best* dilakukan dengan membandingkan nilai *fitness* posisi partikel sekarang (iterasi sekarang) dengan posisi partikel sebelumnya (iterasi sebelumnya). Jika posisi sekarang nilai *fitness*-nya lebih kecil (dalam hal ini mengacu pada *loss*) maka posisi sekarang menjadi *P-best* partikel untuk proses *update* posisi (*PSO*) di iterasi berikutnya. Sama halnya dengan pengecekan *G-best.* Setelah mengetahui *G-best* iterasi sebelumnya, kemudian dibandingkan dengan *G-best* pada iterasi sekarang. Jika lebih baik dari yang sebelumnya, maka posisi *G-best* sekarang menjadi *G-best* untuk *update* posisi di iterasi berikutnya.

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Bab ini menjelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasikan.

5.1 Uji Coba

5.1.1 Lingkungan Uji Coba

Lingkungan uji coba yang akan digunakan adalah:

1. Perangkat keras:
   Processor Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz. 4.00 GB RAM.
2. Sistem operasi :
   Windows 7 Ultimate 64-bit.
3. Perangkat pegembang :
   MATLAB R2014a.

5.1.2 Data Uji Coba

Data yang digunakan untuk uji coba seleksi fitur menggunakan moteode *hybrid particle swarm optimization* dengan operasi *local search* (*HPSO-LS*) adalah data UCI. Jumlah data yang diuji coba sebanyak 3, yaitu *dataset glass*, *wine* dan *heart.* Data tersebut memiliki ukuran m x n dengan n-1 merupakan jumlah fitur/atribut dan kolom ke-n merupakan kelas/label dari *instances.* Deskripsi *dataset* secara rinci telah dijelaskan pada subab 3.1.

5.1.3 Alur Uji Coba

Alur uji coba sesuai dengan *flowcart* sistem (Gambar 3.1). Dimulai dari *input* hingga *output.* Tahap *input* cukup sederhana, dengan memasukkan nilai *k (neighbor)* pada *k-NN*, *Sf* dan *dataset*. Pastikan *dataset* yang *input*-kan telah memenuhi

kriteria yang ditetapkan. Proses memasukkan nilai-nilai dan *dataset* tersebut dilakukan di *script* program. Nilai *k (neighbor)* pada *k-NN* merupakan nilai positif *integer* dengan *range* $6 \leq k \leq 10$. Nilai *subset* fitur *sf* merupakan nilai positif *integer* dengan $Sf < F$. F merupakan jumlah fitur keseluruhan dari *dataset.*

Output akan otomatis muncul ketika program dijalankan. *Output* berupa subset fitur yang terpilih dan akurasi dari klasifikasi data *k-NN* menggunakan fitur-fitur terpilih tersebut.

5.1.4 Skenario Uji Coba

Subbab ini memaparkan urutan skenario uji coba yang telah dilakukan. Berikut urut-urutannya:

1. Skenario 1 melakukan seleksi fitur menggunakan algoritma *PSO.*
2. Skenario 2 melakukan seleksi fitur menggunakan algoritma *HPSO-LS.*
3. Skenario 3 mengambil kombinasi fitur-fitur terpilih yang memiliki performa akurasi klasifikasi data *k-NN* terbaik.
4. Skenario 4 membandingkan hasil akurasi klasifikasi data *k-NN* menggunakan fitur-fitur terpilih hasil seleksi fitur *PSO*, *HPSO-LS* dan mengunakan fitur keseluruhan (tanpa seleksi fitur).

Skenario 1 dan 2 menghasilkan fitur-fitur terpilih hasil seleksi fitur masing-masing algoritma dan hasil akurasi klasifikasi data *k-NN* pada masing-masing fitur terpilih.

Skenario 1 hanya mengacu pada nilai *k (neighbor)* pada *k-NN*. Sedangkan pada skenario 2 mengacu pada nilai *Sf* dan nilai *k (neighbor)* pada *k-NN*. Nilai *Sf* yang digunakan yaitu sesuai dengan kriteria yang ditentukan ($1 \leq Sf < F$). F merupakan jumlah fitur keseluruhan dari *dataset*. Sedangkan nilai *k* yang digunakan yaitu $6 \leq k \leq 10$.

5.1.5 Hasil Uji Coba Seleksi Fitur Menggunakan *PSO*

Tabel 5.1 Hasil Seleksi Fitur *PSO Dataset Glass*

No.	Nilai *k*	Jumlah Fitur Terpilih	Kombinasi Fitur Terpilih	Nilai Akurasi (%)
1.	6	**5**	**3,4,6,7,8**	**71,49**
		6	3,4,5,6,7,8	71,96
		7	1,2,3,4,5,6,8	71,02
		8	**1,2,3,4,5,6,7,8**	**71,49**
2.	7	5	1,2,3,4,6	70,09
		6	2,3,4,6,7,8	71,02
		7	**1,2,3,4,5,6,8**	**71,49**
		8	1,2,3,4,5,6,7,8	71,49
3.	8	5	1,2,3,4,6	70,09
		6	2,3,4,6,7,8	71,02
		7	1,2,3,4,5,6,8	71,49
		8	**1,2,3,4,5,6,7,8**	**71,49**
4.	9	5	3,4,5,6,8	71,02
		6	**1,3,4,5,6,8**	**72,42**
		7	1,2,3,4,5,6,8	71,49
		8	1,2,3,4,5,6,7,8	70,56

Tabel 5.2 Hasil Kombinasi Fitur Terpilih Terbaik *PSO Glass*

No.	Jumlah Fitur Terpilih	Kombinasi Fitur Terpilih	Nilai Akurasi (%)
1.	5	3,4,6,7,8	71,49
2.	**6**	**1,3,4,5,6,8**	**72,42**
3.	7	1,2,3,4,5,6,8	71,49
4.	8	1,2,3,4,5,6,7,8	71,49

Tabel 5.3 Hasil Seleksi Fitur *PSO Dataset Wine*

No.	**Nilai *k***	**Jumlah Fitur Terpilih**	**Fitur Terpilih**	**Nilai Akurasi (%)**
1.	6	6	1,4,5,7,10,13	97,19
		7	2,4,6,10,11,12,13	97,75
		8	1,2,3,4,10,11,12,13	97,75
		9	1,2,3,4,7,10,11,12,13	97,19
		10	1,2,3,4,7,8,10,11,12,13	97,19
2.	7	7	1,2,6,10,11,12,13	97,19
		8	1,2,4,9,10,11,12,13	97,75
		9	**1,2,3,4,5,10,11,12,13**	**98,87**
		10	**1,2,3,4,5,7,10,11,12,13**	**98,31**
3.	8	6	2,4,5,7,10,13	96,62
		7	**2,3,4,10,11,12,13**	**98,31**
		8	1,2,3,4,7,10,12,13	97,19
		9	1,2,3,4,5,10,11,12,13	97,75
		10	1,2,3,4,5,6,7,10,12,13	96,62
4.	9	**6**	**2,3,10,11,12,13**	**97,75**
		7	2,4,7,10,11,12,13	98,31
		8	**1,2,3,4,10,11,12,13**	**98,86**
		9	1,2,3,4,7,10,11,12,13	97,75
		10	1,2,4,5,7,9,10,11,12,13	96,62
		11	**1,2,3,4,5,6,7,9,10,11,12,13**	**97,19**
5.	10	6	2,4,10,11,12,13	97,19
		7	1,3,4,10,11,12,13	98,31
		8	1,2,3,4,10,11,12,13	98,31
		9	**1,2,3,4,7,10,11,12,13**	**98,31**
		10	1,2,3,4,5,7,10,11,12,13	97,19
		11	1,2,3,4,5,6,7,8,10,11,12,13	96,62

Tabel 5.4 Hasil Kombinasi Fitur Terpilih Terbaik *PSO Wine*

No.	Jumlah Fitur Terpilih	Kombinasi Fitur Terpilih	Nilai Akurasi (%)
1.	6	2,3,10,11,12,13	97,75
2.	7	2,3,4,10,11,12,13	98,31
3.	8	1,2,3,4,10,11,12,13	98,86
4.	**9**	**1,2,3,4,5,10,11,12,13**	**98,87**
5.	10	1,2,3,4,5,7,10,11,12,13	98,31
6.	11	1,2,3,4,5,6,7,9,10,11,12,13	97,19

Uji coba seleksi fitur menggunakan *PSO*, menghasilkan beberapa kombinasi fitur-fitur yang terpilih. Jumlah dari fitur-fitur terpilih ini juga beragam. Hasil-hasil tersebut akan dipaparkan dalam bentuk tabel. Tabel berisi nilai *k* pada *k-NN* yang digunakan, jumlah fitur terpilih, kombinasi fitur terpilih dan nilai akurasi klasifikasi data *k-NN* menggunakan fitur-fitur terpilih tersebut. Nilai *k* pada *k-NN* ditetapkan berkisar $6 \leq k \leq 10$. Nilai-nilai yang ditebalkan penulisannya pada masing-masing tabel, menunjukkan nilai-nilai dengan akurasi terbaik.

Tabel 5.1 menunjukkan beberapa jumlah fitur terpilih yaitu, 5,6,7 dan 8 jumlah fitur. Tabel 5.1 juga mencantumkan hasil percobaan ketika dimasukkan nilai *k=10*. Hal tersebut dikarenakan nilai akurasi yang dihasilkan tidak lebih baik dari uji coba dengan nilai *k* sebelum-sebelumnya. Tabel 5.2 menunjukkan hasil kombinasi fitur terbaik pada masing-masing jumlah fitur terpilih.

Tabel 5.2 menunjukkan nilai akurasi terbaik dari masing-masing kombinasi fitur terpilih pada uji coba seleksi fitur dengan *dataset glass*. Nilai akurasi terbaik yaitu sebesar 72,42% dengan jumlah fitur 6 dan fitur-fitur terpilihnya adalah 1,3,4,5,6,8.

Tabel 5.3 menunjukkan beberapa jumlah fitur terpilih yaitu, 5,6,7,8,9,10 dan 11 jumlah fitur. Tabel 5.4 menunjukkan hasil kombinasi fitur terbaik pada masing-masing jumlah fitur terpilih.

Tabel 5.4 menunjukkan nilai akurasi terbaik dari masing-masing kombinasi fitur terpilih pada uji coba seleksi fitur dengan *dataset wine* didapatkan nilai akurasi terbaik yaitu sebesar 98,87% dengan jumlah fitur 9 dan fitur-fitur terpilihnya adalah 1,2,3,4,5,10,11,12,13.

Tabel 5.5 menunjukkan beberapa jumlah fitur terpilih yaitu, 4,5,6,7,8,9 dan 10 jumlah fitur. Tabel 5.6 menunjukkan hasil kombinasi fitur terbaik pada masing-masing jumlah fitur terpilih.

Tabel 5.6 menunjukkan nilai akurasi terbaik dari masing-masing kombinasi fitur terpilih pada uji coba seleksi fitur dengan *dataset heart* didapatkan nilai akurasi terbaik yaitu sebesar 86,66% dengan jumlah fitur 5 dan 7. Dan fitur-fitur terpilih pada jumlah fitur 5 adalah 1,3,9,12,13 dan pada jumlah fitur 7 adalah 1,3,4,5,10,12,13.

Hasil uji coba dari masing-masing *dataset* menunjukkan bahwa nilai *k* berpengaruh pada kombinasi fitur-fitur terpilih. Sebagai contoh pada Tabel 5.1, nilai *k* 6 sampai 9 menghasilkan kombinasi fitur-fitur terpilih yang berbeda-beda. Jika dilihat pada jumlah fitur terpilih 5, $k = 6$ menghasilkan kombinasi fitur terpilih 3,4,6,7,8. K=7 menghasilkan kombinasi fitur terpilih 1,2,3,4,6. K=9 menghasilkan kombinasi fitur terpilih 3,4,5,6,8. Namun pada $k = 7$ dan 8, memiliki kesamaan kombinasi fitur-fitur terpilih. Hal tersebut juga terjadi pada *dataset wine* dan *heart.* Pada masing-masing nilai *k*, terdapat perbedaan pemilihan kombinasi fitur-fitur. Namun ada juga yang memiliki kesamaan pemilihan kombinasi fitur-fiturnya, walaupun nilai *k*-nya berbeda.

Tabel 5.5 Hasil Seleksi Fitur *PSO Dataset Heart*

No.	Nilai *k*	Jumlah Fitur Terpilih	Fitur Terpilih	Nilai Akurasi (%)
1.	6	**4**	**3,9,12,13**	**84,44**
		6	1,3,6,9,12,13	84,81
		7	1,3,5,8,9,12,13	85,55
		8	1,3,4,8,9,10,12,13	85,55
		9	1,3,5,8,9,10,11,12,13	85,55
2.	7	**5**	**1,3,9,12,13**	**84,81**
		6	**1,2,3,10,12,13**	**85,55**
		7	1,3,8,9,10,12,13	85,92
		8	1,3,4,9,10,11,12,13	84,81
		9	3,4,5,6,7,9,10,12,13	83,70
		10	1,2,3,5,8,9,10,11,12,13	82,96
3.	8	7	1,3,9,10,11,12,13	86,29
		8	1,3,8,9,10,11,12,13	86,29
		9	1,3,5,8,9,10,11,12,13	85,29
4.	9	**7**	**1,3,4,5,10,12,13**	**86,66**
		8	1,3,4,5,8,10,12,13	86,29
		9	1,2,3,4,5,8,10,12,13	85,55
		10	1,3,4,5,7,8,9,10,12,13	84,44
5.	10	6	3,4,9,10,12,13	84,81
		7	1,3,8,9,10,12,13	85,55
		8	**1,3,8,9,10,11,12,13**	**85,92**
		9	**1,3,5,7,8,9,10,12,13**	**85,92**
		10	**1,3,4,5,8,9,10,11,12,13**	**85,18**

Pemilihan fitur-fitur tersebut berdasarkan pemilihan posisi-posisi terbaik dari partikel. Dengan kata lain, pemilihan fitur menggunakan algoritma *PSO* hanya mengacu pada nilai akurasi terbaik dari fitur. Kemudian fitur-fitur tersebut

digunakan kembali sebagai acuan untuk mencari atau melakukan pemilihan fitur kembali.

Tabel 5.6 Hasil Kombinasi Fitur Terpilih Terbaik *PSO Heart*

No.	Jumlah Fitur Terpilih	Kombinasi Fitur Terpilih	Nilai Akurasi (%)
1.	4	3,9,12,13	84,44
2.	5	1,3,9,12,13	84,81
3.	6	1,2,3,10,12,13	85,55
4.	7	**1,3,4,5,10,12,13**	**86,66**
5.	8	1,3,8,9,10,11,12,13	85,92
6.	9	1,3,5,7,8,9,10,12,13	85,92
7.	10	1,3,4,5,8,9,10,11,12,13	85,18

5.1.6 Hasil Uji Coba Seleksi Fitur Menggunakan *HPSO-LS*

Nilai *Sf* digunakan pada saat operasi *local particle search,* untuk menghitung nilas n_s dan n_d. Kedua nilai tersebut (n_s dan n_d) digunakan pada proses *particle movement.* Proses ini menerapkan operasi *add* dan *delete.* Operasi *add* dijalankan sesuai dengan *pseudo code* 5.1

Algortima 1. Operasi *Add* pada *Particle Movement*

1. Jika jumlah fitur di $Xd < n_d$
2. $(n_d - Xd)$ fitur di $(D - Xd)$ ditambahkan (*add*) di *Xd*.
3. Jika jumlah fitur $Xs < n_s$
4. $(n_s - Xs)$ fitur di $(S - Xs)$ ditambahkan (*add*) di *Xs*.

Pseudocode 5.1 Operasi *Add* pada *Particle Movement*

Operasi *add* ini tidak dapat dilakukan jika fitur yang akan ditambahkan di kedua *Xd* dan *Xs* yaitu $(D - Xd)$ dan $(S -$

Xs) fitur, tidak memenuhi perhitungan jumlah fitur *(n_s – Xs)* dan *(n_d – Xd)*. Jika operasi *add* ini tidak dapat dijalankan, maka *local particle search* tidak dapat dijalankan. Untuk lebih jelasnya, akan diberikan contoh sebagai berikut:

D = [3,5,2,6,7]; S = [1,4,9,8]
F = 9;
Sf = 7;
n_d = 2; n_s = 5
Partikel = [0,1,1,1,0,1,0,0,0]
= [2,3,4,6]

Perhitungan nilai n_s dan n_d menggunkama persamaan 2.7 dan 2.8.

a. *Feature Segmentation :*
Xd = [3,2,6]; *Xs* = [4]
Jumlah fitur di *Xd* = 3
Jumlah fitur di *Xs* = 1
b. *Particle Movement (add):*
Xs < n_s
(n_s – Xs) = 4;
(S - Xs) = [1,9,8];

Jadi 4 fitur di *(S - Xs)* tidak dapat ditambahkan di *Xs*, karena fitur-fitur di *(S - Xs)* hanya terdapat 3 fitur. Contoh tersebut menunjukkan bahwa jika nilai *Sf* yang di-*input*-kan mendekati jumlah fitur keseluruhan *dataset*, maka operasi *local particle search* tidak dapat dijalankan.

Dari hasil uji coba, nilai *Sf* yang dapat di-*input*-kan di masing-masing uji coba *dataset* adalah sebagai berikut:

Glass = 1 sampai 6
Wine = 1 sampai 9
Heart = 1 sampai 9

Perlu diperhatikan bahwa proses *local particle search* memerlukan nilai korelasi antar fitur. Nilai korelasi ini menunjukkan hubungan antara fitur satu dengan yang lain. Semakin tinggi nilai korelasi fitur tersebut, maka kemungkinan fitur tersebut dipilih akan sedikit. Sebaliknya jika fitur memiliki nilai korelasi yang rendah maka semakin besar kemungkinan fitur tersebut dipilih. Hasil perhitungan nilai korelasi antar fitur pada masing-masing ditunjukkan pada Tabel 5.7, 5.8 dan 5.9 berdasarkan *ranking* korelasi yang terendah dibawah ini.

Tabel 5.7 Nilai Korelasi Fitur pada *Dataset Glass*

Fitur	**Nilai Korelasi**
3	-0,89
5	-0,38
2	0,17
6	0,21
7	0,32
1	0,39
4	0,73
9	0,87
8	1,00

Hasil perhitungan nilai korelasi antar fitur pada Tabel 5.7 memiliki nilai rata-rata sebesar 0,27. Hasil perhitungan nilai korelasi antar fitur pada Tabel 5.8 memiliki nilai rata-rata sebesar 1,89. Hasil perhitungan nilai korelasi antar fitur pada Tabel 5.8 memiliki nilai rata-rata sebesar 1,94. Hasil perhitungan nilai korelasi menunjukkan rata-rata nilai korelasi fitur pada *dataset glass* adalah yang paling kecil. Nilai rata-rata *dataset glass* ini sebesar 0,27. Nilai-nilai ini nantinya digunakan sebagai acuan untuk proses *local search* pada *HPSO-LS.*

Tabel 5.8 Nilai Korelasi Fitur pada *Dataset Wine*

Fitur	Nilai Korelasi
8	-0,86
4	-0,14
2	-0,05
11	1,36
10	1,52
12	2,45
5	2,52
3	2,96
1	2,96
9	2,98
7	3,42
13	3,50
6	3,58

Tabel 5.9 Nilai Korelasi Fitur pada *Dataset Heart*

Fitur	Nilai Korelasi
8	-1,54
2	1,49
6	1,51
5	1,71
3	1,98
7	2,00
4	2,24
1	2,26
9	2,51
11	2,56
12	2,63
13	2,91
10	2,90

Tabel 5.10 Hasil Seleksi Fitur *HPSO-LS Dataset Glass*

No.	***Sf***	**Nilai *k***	**Fitur Terpilih**	**Nilai Akurasi (%)**
1.	1	6	4	61,41
		7	4	61,68
		8	4	61,21
		9	**4**	**62,14**
		10	4	61,21
2.	2	6	3,4	65,88
		7	3,4	68,69
		8	**3,4**	**69,15**
		9	3,4	69,15
		10	3,4	67,75
3.	3	6	3,4,8	69,15
		7	3,4,8	69,62
		8	**3,4,8**	**70,09**
		9	3,4,8	70,09
		10	3,4,8	69,15
4.	4	6	1,3,4,8	68,69
		7	1,3,4,8	69,62
		8	**1,3,4,8**	**70,09**
		9	1,3,4,8	70,09
		10	1,3,4,8	69,62
5.	5	6	1,3,4,6,8	71,02
		7	1,3,4,6,8	70,09
		8	1,3,4,6,8	70,56
		9	**1,3,4,6,8**	**71,02**
		10	1,3,4,6,8	71,02
6.	6	6	1,2,3,4,8,9	69,15
		7	1,2,3,4,8,9	69,15
		8	**1,2,3,4,8,9**	**70,09**
		9	1,2,3,4,8,9	68,69

Hasil uji coba seleksi fitur menggunakan *HPSO-LS* dipaparkan dalam bentuk tabel. Tabel terdiri dari nilai *Sf,* nilai *k* pada *k-NN,* fitur terpilih dan nilai akurasi klasifikasi *k-NN* menggunakan fitur-fitur terpilih tersebut.

Tabel 5.10 merupakan hasil dari uji coba seleksi fitur menggunakan algoritma *HPSO-LS* dengan *dataset glass*. Hasil uji coba *HPSO-LS* menggunakan *dataset glass,* didapatkan beberapa fitur terpilih dan hasil akurasi terbaiknya pada masing-masing *Sf*. Hasil akurasi terbaik dari masing-masing *Sf*-nya dijelaskan pada Tabel 5.11.

Tabel 5.11 Hasil Akurasi Terbaik di *Sf dataset glass HPSO-LS*

No.	***Sf***	**Fitur Terpilih**	**Nilai Akurasi (%)**
1.	1	4	62,14
2.	2	3,4	69,15
3.	3	3,4,8	70,09
4.	4	1,3,4,8	70,09
5.	**5**	**1,3,4,6,8**	**72,42**
6.	6	1,2,3,4,8,9	70,09

Tabel 5.11 menunjukkan nilai akurasi terbaik pada uji coba seleksi fitur *HPSO-LS* menggunakan *dataset glass* adalah 72,42 dengan *Sf* sebesar 5 dan fitur-fitur terpilihnya adalah 1,3,4,6,8.

Tabel 5.12 merupakan hasil dari uji coba seleksi fitur dengan *dataset wine*. Nilai *sf* yang dapat dimasukkan pada uji coba tersebut yaitu antara 1 sampai 9. Namun Tabel 5.12 hanya mencantumkan *sf* 4-9. Hasil uji coba *HPSO-LS* menggunakan *dataset wine,* didapatkan beberapa fitur terpilih dan hasil akurasi terbaiknya pada masing-masing *Sf*. Tabel hasil akurasi terbaik dari masing-masing *Sf*-nya dipaparkan pada Tabel 5.13.

Tabel 5.12 Hasil Seleksi Fitur *HPSO-LS Dataset Wine*

No.	***Sf***	**Nilai *k***	**Fitur Terpilih**	**Nilai Akurasi (%)**
4.	4	**6**	**1,7,10,13**	**97,19**
		7	1,7,10,13	96,62
		8	3,7,10,13	96,62
		9	3,7,10,13	96,62
		10	3,7,10,13	96,62
5.	5	**6**	**1,2,7,11,13**	**97,75**
		7	1,2,7,11,13	97,75
		8	1,7,10,12,13	97,19
		9	1,7,10,12,13	97,75
		10	1,3,10,11,13	97,75
6.	**6**	6	1,2,3,7,10,13	97,19
		7	1,2,7,9,10,13	98,31
		8	1,3,4,7,10,13	97,75
		9	**1,3,4,7,10,13**	**98,31**
		10	1,4,6,7,11,13	96,62
7.	7	6	1,3,4,7,9,10,13	97,19
		7	**1,3,4,7,9,10,13**	**97,75**
		8	1,3,4,7,9,10,13	97,75
		9	1,3,4,7,9,10,13	97,75
		10	1,2,3,7,9,10,13	97,19
8.	8	6	1,2,3,4,7,9,10,13	97,19
		7	**1,3,4,5,7,9,10,13**	**97,75**
		8	1,3,4,7,9,10,11,13	96,62
		9	1,3,4,7,9,10,12,13	97,75
		10	1,3,4,7,9,10,11,13	97,19
9.	**9**	6	1,3,4,6,7,9,10,11,13	96,62
		7	**1,3,4,6,7,9,10,12,13**	**98,31**
		8	1,3,4,6,7,9,10,12,13	97,19
		9	1,3,4,6,7,9,10,12,13	97,19

Tabel 5.13 Hasil Akurasi Terbaik di *Sf Dataset Wine HPSO-LS*

No.	***Sf***	**Fitur Terpilih**	**Nilai Akurasi (%)**
4.	4	1,7,10,13	97,19
5.	5	1,2,7,11,13	97,75
6.	**6**	**1,3,4,7,10,13**	**98,31**
7.	7	1,3,4,7,9,10,13	97,75
8.	8	1,3,4,5,7,9,10,13	97,75
9.	**9**	**1,3,4,6,7,9,10,12,13**	**98,31**

Tabel 5.13 menunjukkan nilai akurasi terbaik pada uji coba seleksi fitur *HPSO-LS* menggunakan *dataset wine* adalah 98,31 dengan *Sf* sebesar 6 dan 9. Fitur-fitur terpilih pada *Sf* 6 adalah 1,2,7,9,10,13 dan fitur-fitur terpilih pada *Sf* 9 adalah 1,3,4,6,7,9,10,12,13.

Tabel 5.14 merupakan hasil dari uji coba seleksi fitur dengan *dataset heart*. Nilai *sf* yang dapat dimasukkan pada uji coba tersebut yaitu antara 1 sampai 9. Namun Tabel 5.14 hanya mencantumkan *sf* 4 sampai 9 saja. Karena jumlah kombinasi fitur terpilih *dataset heart* yang dapat dibandingkan dengan hasil algortima *PSO* hanya yang berjumlah 4 sampai 9. Uji coba seleksi fitur menggunakan alogoritma *PSO* dengan *dataset heart* tidak menghasilkan kombinasi fitur terpilih sejumlah 1 sampai 3.

Hasil uji coba *HPSO-LS* menggunakan *dataset heart*, didapatkan beberapa fitur terpilih dan hasil akurasi terbaiknya pada masing-masing *Sf*. Tabel 5.15 menunjukkan hasil akurasi terbaik dari masing-masing *Sf*-nya.

Tabel 5.14 Hasil Seleksi Fitur *HPSO-LS Dataset Heart*

No.	***Sf***	**Nilai *k***	**Fitur Terpilih**	**Nilai Akurasi (%)**
4.	4	**6**	**3,9,12,13**	**84,44**
		7	8,9,12,13	83,70
		8	8,9,12,13	84,07
		9	8,9,12,13	84,44
		10	8,9,12,13	83,70
5.	5	6	3,8,10,12,13	84,44
		7	1,3,5,12,13	85,15
		8	1,3,5,12,13	85,55
		9	**1,3,8,12,13**	**85,92**
		10	1,3,5,12,13	85,92
6.	6	**6**	**1,3,8,9,12,13**	**86,29**
		7	1,3,8,9,12,13	86,29
		8	1,3,8,9,12,13	85,92
		9	1,3,5,9,12,13	85,55
		10	1,3,5,9,12,13	85,18
7.	**7**	6	1,3,8,9,10,12,13	85,92
		7	1,3,8,9,10,12,13	86,29
		8	**1,3,8,9,11,12,13**	**86,66**
		9	1,5,8,9,11,12,13	85,55
		10	1,3,8,9,11,12,13	86,29
8.	8	6	1,3,5,8,9,10,12,13	85,92
		7	**1,3,5,8,9,10,12,13**	**86,29**
		8	1,3,4,8,9,10,12,13	86,29
		9	1,3,6,8,9,11,12,13	85,55
		10	1,3,4,8,9,10,12,13	86,29
9.	9	7	1,3,4,8,9,10,11,12,13	86,29
		8	1,3,5,8,9,10,11,12,13	86,66
		9	1,3,5,8,9,10,11,12,13	85,92
		10	**1,3,5,8,9,10,11,12,13**	**86,66**

Tabel 5.15 Hasil Akurasi Terbaik di *Sf Dataset Heart HPSO-LS*

No.	*Sf*	Fitur Terpilih	Nilai Akurasi (%)
1.	1	13	74,81
2.	2	3,10	78,14
3.	3	3,12,13	86,29
4.	4	3,9,12,13	84,44
5.	5	1,3,8,12,13	85,92
6.	6	1,3,8,9,12,13	86,29
7.	**7**	**1,3,8,9,11,12,13**	**86,66**
8.	8	1,3,5,8,9,10,12,13	86,29
9.	9	1,3,5,8,9,10,11,12,13	86,66

Tabel 5.15 menunjukkan nilai akurasi terbaik pada uji coba seleksi fitur *HPSO-LS* menggunakan *dataset heart* adalah 98,31 dengan *Sf* sebesar 6 dan 9. Fitur-fitur terpilih pada *Sf* 6 adalah 1,3,8,9,11,12,13 dan fitur-fitur terpilih pada *Sf* 9 adalah 1,3,3,5,8,9,10,12,13.

5.1.7 Perbandingan Akurasi *k-NN PSO, HPSO-LS* dan Tanpa Seleksi Fitur (TSF)

Subbab ini membahas tentang perbandingan akurasi klasifikasi data *k-NN* dengan mengambil jumlah fitur terpilih yang memiliki nilai akurasi terbaik di masing-masing *dataset* pada uji coba *PSO* dan *HPSO-LS*. Kemudian dibandingkan hasil akurasi antara *PSO* dan *HPSO-LS.* Jumlah fitur-fitur terpilih yang akan dibandingkan akurasinya, pada hasil seleksi fitur *PSO* yang diambil, disamakan atau disesuaikan dengan jumlah fitur-fitur terpilih pada hasil seleksi fitur *HPSO-LS.*

Perbandingan juga dilakukan dengan melihat hasil akurasi klasifikasi *k-NN* jika menggunakan fitur keseluruhan (TSF).

Tabel 5.16 Perbandingan akurasi *k-NN PSO* dan *HPSO-LS Dataset Glass*

SF	***PSO***		***HPSO-LS***	
	Fitur Terpilih	**Akurasi**	**Fitur Terpilih**	**Akurasi (%)**
5	**3,4,6,7,8**	**71,49**	1,3,4,6,8	71,02
6	**1,3,4,5,6,8**	**72,42**	1,2,3,4,8,9	70,09

Tabel 5.16 menunjukkan kedua algoritma *PSO* dan *HPSO-LS* memiliki jumlah fitur terpilih yang sama sebesar 5 dan 6. Hasil akurasi keduanya menunjukkan bahwa akurasi *PSO* lebih tinggi dari pada *HPSO-LS* pada jumlah fitur 5 sebesar 71,49 dan pada jumlah fitur 6 yaitu sebesar 72,42. Hal ini dikarenakan fitur-fitur yang terpilih pada *PSO* sebagian besar adalah fitur-fitur yang memiliki tingkat korelasi yang rendah.

Tabel 5.17 Perbandingan akurasi *k-NN PSO* dan *HPSO-LS Dataset Wine*

SF	***PSO***		***HPSO-LS***	
	Fitur Terpilih	**Akurasi**	**Fitur Terpilih**	**Akurasi (%)**
6	2,3,10,11,12, 13	97,75	**1,3,4,7,10,13**	**98,31**
7	**2,3,4,10,11,1 2,13**	**98,31**	1,3,4,7,9,10,13	97,75
8	**1,2,3,4,10,11 ,12,13**	**98,86**	1,3,4,5,7,9,10, 13	97,75
9	**1,2,3,4,5,10, 11,12,13**	**98,87**	1,3,4,6,7,9,10, 12,13	98,31

Tabel 5.17 menunjukkan hasil seleksi fitur *PSO* dan *HPSO-LS* memiliki jumlah fitur terpilih yang sama sebesar 6,7,8 dan 9. Nilai akurasi terbesarnya adalah pada algortima *PSO* dengan jumlah fitur terpilih sebanyak 9 dengan besar akurasi 98,87. Namun hasil seleksi fitur *HPSO-LS* memiliki nilai akurasi yang lebih besar dibandingkan *PSO* pada saat jumlah fitur terpilihnya sebesar 6 yaitu 98,31. Jumlah fitur terpilih 7,8 dan 9, hasil fitur-fitur terpilih *PSO* memiliki nilai akurasi yang lebih besar dari pada *HPSO-LS.* Hal ini dikarenakan pada *PSO* sebagian besar kombinasi fitur-fitur terpilihnya memiliki nilai korelasi yang rendah jika dibandingkan dengan fitur-fitur terpilih *HPSO-LS.*

Tabel 5.18 Perbandingan akurasi *k-NN PSO* dan *HPSO-LS Dataset Heart*

SF	***PSO***		***HPSO-LS***	
	Fitur Terpilih	**Akurasi**	**Fitur Terpilih**	**Akurasi (%)**
4	**3,9,12,13**	**84,44**	**3,9,12,13**	**84,44**
5	1,3,9,12,13	84,81	**1,3,8,12,13**	**85,92**
6	1,2,3,10,12,13	85,55	**1,3,8,9,12,13**	**86,29**
7	**1,3,4,5,10,12,13**	**86,66**	**1,3,8,9,11,12,13**	**86,66**
8	1,3,8,9,10,11,12,13	85,92	**1,3,5,8,9,10,12,13**	**86,29**
9	1,3,5,7,8,9,10,12,13	85,92	**1,3,5,8,9,10,11,12,13**	**86,66**

Tabel 5.18 menunjukkan nilai akurasi terbesar dari uji coba kedua algoritma adalah sebesar 86,66%. Nilai akurasi tersebut didapatkan dari 2 jumlah fitur yang berbeda, yaitu jumlah fitur 7 dan 9. Jumlah fitur 7, fitur-fitur didapatkan dari hasil seleksi fitur menggunakan *PSO* dan *HPSO-LS.* Jadi jika

ingin melakukan proses klasifikasi data *k-NN* dengan *dataset heart* agar lebih maksimal menggunakan jumlah fitur 7 dengan fitur-fitur 1,3,8,9,11,12,13. Selain nilai akurasinya tinggi, jumlah fiturnya pun juga lebih sedikit. Sehingga dapat meningkatkan kecepatan klasifikasi data.

Tabel 5.19 Hasil Akurasi *k-NN* Menggunakan Fitur Keseluruhan

Dataset	**Nilai *k***	**Akurasi (%)**
Glass	6	64,95
	7	**68,69**
	8	65,88
	9	68,69
	10	65,42
Wine	**6**	**93,82**
	7	93,25
	8	92,69
	9	93,82
	10	93,82
Heart	6	79,62
	7	79,62
	8	80,00
	9	79,62
	10	**80,37**

Tabel 5.19 menunjukkan hasil uji coba perhitungan akurasi klasifikasi data *k-NN*. Uji coba ini dilakukan dengan mengacu pada nilai *k* pada *k-NN*. Nilai *k* yang di-*input*-kan adalah sama dengan nilai-nilai *k* ketika melakukan uji coba seleksi fitur menggunakan algoritma *PSO* dan algoritma *HPSO-LS.*

5.2 Evaluasi

Dataset yang digunakan merupakan *dataset* yang berisi bilangan. Tidak semua *dataset* yang didapatkan sesuai dengan kriteria data *input*-an sistem. *Dataset* ini harus diolah terlebih dahulu agar sesuai dengan kriteria. Sistem ini memasukkan *dataset* berupa matriks berukuran *m x n*, dimana *m* merupakan jumlah data pada *dataset*, dan *n* merupakan jumlah fitur. *Class-class* pada *dataset* harus terletak pada bagian setelah fitur terakhir. Sistem ini tidak dapat melakukan seleksi fitur dengan *dataset* yang berisi selain bilangan/nilai. Sistem ini hanya mampu melakukan seleksi fitur menggunakan *dataset* yang berisi bilangan/nilai.

Tabel 5.19 menunjukkan nilai akurasi klasifikasi data *k-NN* menggunakan fitur keseluruhan jauh lebih rendah jika dibandingkan dengan fitur terpilih hasil seleksi fitur *PSO* maupun *HPSO-LS.* Hal tersebut dikarenakan fitur-fitur yang digunakan ketika melakukan klasifikasi data, masih terdapat fitur yang *redundant*. Atau dengan kata lain fitur yang digunakan masih terdapat fitur-fitur dengan nilai korelasi yang tinggi. Nilai akurasi terbesar pada klasifikasi *k-NN* menggunakan fitur keseluruhan pada *dataset glass* adalah sebesar 68,69%, sedangkan nilai akurasi terbesar menggunakan fitur terpilih *HPSO-LS* sebesar 71,02% menggunakan 5 fitur.

Akurasi klasifikasi data *k-NN* menggunakan fitur keseluruhan terbesar adalah 93,82% sedangkan untuk fitur terpilih *HPSO-LS* adalah 98,31% menggunakan 6 atau 9 fitur pada *dataset wine*. Untuk data *heart,* akurasi *k-NN* menggunakan fitur keseluruhan terbesar adalah 80,37 sedangkan untuk fitur terpilih *HPSO-LS* adalah 86,66 menggunakan 7 atau 9 fitur. Sehingga dapat disimpulkan bahwa proses seleksi fitur *HPSO-LS* ini dapat meningkatkan akurasi

klasifikasi data. Akurasi meningkat sebesar 2,33% pada *dataset glass*, 4,49% pada *dataset wine* dan 6,29% pada *dataset heart*.

Hasil uji coba menunjukkan bahwa korelasi antar fitur sangat berpengaruh terhadap akurasi klasifikasi *k-NN*. Semakin banyak fitur dengan nilai korelasi rendah yang dipilih untuk proses klasifikasi, akurasinya akan lebih besar jika dibandingkan dengan fitur-fitur dengan nilai korelasi tinggi yang dipilih.

Seleksi fitur menggunakan *PSO*, hanya memilih fitur-fitur yang memiliki nilai akurasi atau *fitness* yang terbaik, tanpa memperhatikan korelasi fitur-fitur tersebut. Jadi bisa saja fitur dengan nilai korelasi rendah dapat terpilih lebih banyak daripada fitur dengan korelasi tinggi (sehingga dapat meningkatkan akurasi), ataupun sebaliknya.

Seleksi fitur menggunakan algoritma *HPSO-LS* mengacu pada nilai korelasinya. Proses *local search* di *HPSO-LS*, masih memungkinkan melakukan pemilihan fitur dengan tingkat korelasi tinggi. Hal tersebut dapat terjadi di operasi *add* pada *local search*, dimana beberapa fitur *similar (S)* dapat ditambahkan pada partikel. Dimana *S* berisi setengah dari jumlah fitur keseluruhan yang memiliki nilai korelasi tinggi. Namun pemilihan fitur di *S*, masih menggunakan acuan yaitu memilih fitur dengan nilai korelasi rendah dengan kata lain memilih fitur dengan korelasi terendah di *S*. Hal inilah yang dapat mengakibatkan penurunan akurasi klasifikasi jika dibandingkan dengan *PSO*. Hal tersebut merupakan kelemahan dari algoritma *HPSO-LS*. Sedangkan dari segi kecepatan komputasi, algoritma *PSO* memiliki kecepatan komputasi yang lebih cepat dibandingkan dengan *HPSO-LS*. Karena *HPSO-LS* memerlukan proses tambahan berupa *local search* untuk memilih fitur-fitur dengan korelasi yang rendah dan menghapus fitur-fitur dengan korelasi yang tinggi.

Dari hasil uji coba, nilai *sf* (*subset* fitur) yang dapat di-*input*-kan di masing-masing uji coba *dataset* adalah sebagai berikut: *glass* = 1 sampai 6, *wine* = 1 sampai 9, h*eart* = 1 sampai 9. Hal tersebut dikarenakan jika nilai *sf* semakin mendekati fitur keseluruhan, maka akan terjadi *error* pada proses *local particle search.* Hal tersebut dikarenakan operasi *add* pada *local particle search* tidak dapat dilakukan jika fitur yang akan ditambahkan di kedua *Xd* dan *Xs* yaitu *(D - Xd)* dan *(S - Xs)* fitur, tidak memenuhi perhitungan jumlah fitur $(n_s - Xs)$ dan $(n_d - Xd)$.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. *HPSO-LS* melakukan pencarian fitur berdasarkan nilai *sf*. Parameter *sf* ini mempengaruhi proses *local particle search.* Jika nilai *sf* yang dimasukkan mendekati jumlah fitur keseluruhan, maka proses *local particle search* tidak dapat dijalankan. Hal tersebut dikarenakan ketika operasi *add* dijalankan, ukuran ruang fitur *Xd* ataupun *Xs* tidak dapat menampung jumlah fitur yang akan ditambahkan. Ukuran ruang fitur *Xd* dan *Xs* bergantung pada nilai *sf* yang dimasukkan.
2. Pemilihan fitur dilakukan dengan cara memperhatikan nilai korelasi antar fitur. Nilai korelasi fitur menunjukkan hubungan antar fitur. Semakin kecil nilai korelasi fitur, maka semakin besar kemungkinan fitur tersebut dipilih. Hasil percobaan menunjukkan, cara ini berpengaruh pada proses klasifikasi data. Akurasi meningkat sebesar 2,33% pada *dataset glass*, 4,49% pada *dataset wine* dan 6,29% pada *dataset heart*.
3. *Local search* diterapkan dengan memperhatikan nilai korelasi antar fitur. Prosesnya adalah dengan melakukan pemilihan fitur yang memiliki nilai korelasi yang rendah, serta menghapus fitur yang memiliki nilai korelasi yang tinggi.
4. Hasil uji coba membuktikan bahwa korelasi antar fitur sangat berpengaruh terhadap akurasi klasifikasi. Semakin banyak fitur dengan nilai korelasi rendah yang dipilih untuk proses klasifikasi, maka akurasinya akan lebih besar jika

dibandingkan dengan fitur-fitur dengan nilai korelasi tinggi yang dipilih.

5. Hasil uji coba menunjukkan, pada *dataset heart*, kedua algoritma menghasilkan fitur-fitur terpilih dengan besar akurasi yang sama. Sedangkan akurasi klasifikasi data menggunakan fitur-fitur terpilih algoritma *PSO* lebih tinggi pada *dataset glass* dan *wine* dari pada *HPSO-LS*. Hal tersebut dikarenakan *PSO* hanya memilih fitur-fitur yang dapat menghasilkan akurasi klasifikasi data yang tinggi. Sedangkan *HPSO-LS*, melakukan pemilihan fitur dilakukan dengan memperhatikan nilai korelasi antar fitur. Semakin tinggi nilai korelasi, semakin kecil peluang fitur terpilih.
6. Proses seleksi fitur dapat meningkatkan akurasi klasifikasi data jika dibandingkan dengan menggunakan fitur keseluruhan.

6.2 Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Sistem agar dikembangkan untuk melakukan seleksi fitur dengan dataset yang beragam.
2. Metode pemilihan fitur signifikan agar memilih metode selain pemilihan fitur berdasarkan nilai korelasi antar fitur.
3. Pencarian fitur signifikan dilakukan dengan menggunakan metode selain *local search*. Karena masih ada kelemahan dari metode tersebut.

DAFTAR PUSTAKA

[1] Moradi, Parham., Gholampour, Mozhgan., Jan. 2016. "A hybrid particle swarm optimization for feature subset selection byintegrating a novel local search strategy"

[2] Wihardi, Yaya., April 2013. **K-Fold Cross Validation,** <URL:http://blog.yaya.web.id/riset/k-folds-cross-validation>.

[3] Avelita, Beatrix., 2013. **Klasifikasi K-Nearest Neighbor,** <URL:http://www.academia.edu/9131959/A. Klasifikasi_K-Nearest_Neighbor>.

[4] Kohavi. Provost. 2013. **Confussion Matrix,** <URL:http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html >.

[5] I.A. Gheyas, L.S. Smith., Feature subset selection in large dimensionality domains, Pattern Recogn. 43 (2010) 5–13.

[6] H. Liu, H. Motoda, Computational Methods of Feature Selection, Chapman &Hall/CRC, 2007.

[7] E. Gasca, J.S. Sánchez, R. Alonso, Eliminating redundancy and irrelevanceusing a new MLP-based feature selection method, Pattern Recogn. 39 (2006)313–315.

[8] H. Chun-Nan, H. Hung-Ju, S. Dietrich, The ANNIGMA-wrapper approach tofast feature selection for neural nets, systems, man, and cybernetics, part B:cybernetics, IEEE Trans. 32 (2002) 207–212.

[9] X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based onrough sets and particle swarm optimization, Pattern Recogn. Lett. 28 (2007)459–471.

[10] L.-Y. Chuang, S.-W. Tsai, C.-H. Yang, Improved binary particle swarm opti-mization using catfish effect for feature selection, Expert Syst. Appl. 38 (2011)12699–12707.

[11] Xue, Bing., 2014. "Particle Swarm Optimization for Feature Selection in Classification."

[12] Samuel, Yoseph., Delima, Rosa., Rachmat, Antonius., Juni 2014. "Implementasi Metode K-Nearest Neighbor dengan Decision Rule untuk Klasifikasi Subtopik Berita."

BIODATA PENULIS

Dimas Yoan Shailendra, lahir di Kediri pada tanggal 13 Mei 1994. Penulis adalah anak pertama dari dua bersaudara dan dbersarkan di Kediri. Penulis menempuh pendidikan formal di SD Negeri 1 Krass (2000-2006), SMP Negeri 1 Ngadiluwih (2006-2009), SMA Negeri 2 Kota Kediri (2009-2012). Tahun 2012 penulis memulai pendidikan strata satu di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur. Di Jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Cerdas Visual dan memiliki ketertarikan di bidang kecerdasan buatan, kecerdasan komputasional dan yang berhubungan dengan analisa data secara umum. Penulis dapat dihubungi melalui alamat email dimasyoan8@gmail.com.