OPTIMASI ALGORITMA *SUPPORT VECTOR MACHINE* BERBASIS ALGORITMA *K-MEANS* DAN *PARTICLE SWARM OPTIMIZATION* PADA DIAGNOSIS PENYAKIT GINJAL KRONIS

Skripsi

disusun sebagai salah satu syarat

untuk memperoleh gelar Sarjana Komputer

Program Studi Teknik Informatika

oleh

Hafid Alpin Al Gazni

4611415016

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2020

PERNYATAAN

Saya menyatakan bahwa skripsi saya berjudul "Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma K-Means dan *Particle Swarm Optimization* pada Diagnosis Penyakit Ginjal Kronis" disusun berdasarkan penelitian saya dengan arahan dosen pembimbing. Sumber informasi atau kutipan yang berasal dari karya yang diterbitkan telah disebutkan dalam teks dan dicantumkan dalam daftar pustaka di bagian akhir skripsi ini. Saya menyatakan bahwa skripsi ini bebas plagiat, dan apabila dikemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi ketentuan peraturan perundang-undangan.

Semarang, 24 Juni 2020

METERAI TEMPEL
F2AC8AHF511563403
6000
ENAM RIBU RUPIAH

Hafid Alpin Al Gazni
4611415016

PERSETUJUAN PEMBIMBING

Nama : Hafid Alpin Al Gazni

NIM : 4611415016

Program Studi : S-1 Teknik Informatika

Judul Skripsi : Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma *K-Means* dan *Particle Swarm Optimization* pada Diagnosis Penyakit Ginjal Kronis

Skripsi ini telah disetujui oleh pembimbing untuk diajukan ke sidang panitia ujian skripsi Program Studi Teknik Informatika FMIPA UNNES.

Semarang, 24 Juni 2020
Pembimbing

Budi Prasetiyo, S.Si., M.Kom.
NIP. 198805012014041001

PENGESAHAN

Skripsi yang berjudul

Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma *K-Means* dan *Particle Swarm Optimization* Pada Diagnosis Penyakit Ginjal Kronis

disusun oleh

Hafid Alpin Al Gazin

4611415016

telah dipertahankan di hadapan sidang panitia ujian skripsi FMIPA UNNES pada tanggal 6 April 2020.

Pembimbing

Prof. Dr. Wiyanto, M.Si.
NIP. 196102191993031001

Sekretaris

Dr. Alamsyah, S.Si., M.Kom.
NIP 197405172006041001

Penguji 1

Zaenal Abidin S.Si., M.Cs., Ph.D.
NIP. 198205042005011001

Penguji 2

Endang Sugiharti S.Si., M.Kom.
NIP. 197401071999032001

Anggota Penguji

Budi Prasetiyo S.Si., M.Kom.
NIP. 198805012014041001

MOTTO DAN PERSEMBAHAN

MOTTO

- Selalu lihat sisi positif dari setiap kegagalan, kemudian bangun dan belajar.
- Selalu belajar menjadi manusia yang bermanfaat berlandaskan kejujuran dan keikhlasan untuk orang-orang sekitar.
- Terbentur, terbentur, terbentur kemudian terbentuk.

PERSEMBAHAN

- Kedua orang tua, yang selalu memberikan kasih sayang, doa dan dukunganya.
- Kakek dan nenek saya yang selalu memberikan dukungan moril dan materiil serta semangat dan doa.
- Adik saya yang saya cintai.
- Teman-teman ilmu komputer 2015 yang telah memberikan dukungan dan kenangan semasa perkuliahan.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu hingga terselesaikannya penulisan skripsi ini.
- Almamaterku, Universitas Negeri Semarang
- Dan pembaca semoga dapat memberikan inspirasi untuk penelitian yang akan dilakukan.

PRAKATA

Puji dan syukur kami panjatkan ke hadirat Allah SWT, yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul **"Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma *K-Means* dan *Particle Swarm Optimization* Pada Diagnosis Penyakit Ginjal Kronis "**.

Skripsi ini disusun guna melengkapi salah satu syarat untuk menyelesaikan Program Studi Teknik Informatika, Jurusan Ilmu Komputer Universitas Negeri Semarang. Atas tersusunnya skripsi ini, penulis mengucapkan terima kasih yang sebesar besarnya kepada :

1. Bapak Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang.
2. Bapak Dr. Sugianto, M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Bapak Dr. Alamsyah, S.Si., M.Kom., Ketua Jurusan Ilmu Komputer FMIPA Universitas Negeri Semarang sekaligus dosen penguji yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.
4. Bapak Budi Prasetiyo, S.Si., M.Kom., Dosen Pembimbing yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.
5. Bapak Zaenal Abidin S.Si., M.Cs., Ph.D., Dosen Penguji yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.

6. Endang Sugiharti S.Si., M.Kom., Dosen Penguji yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.
7. Bapak dan Ibu Dosen Jurusan Ilmu Komputer Universitas Negeri Semarang, yang telah memberikan bekal ilmu yang bermanfaat kepada penulis.
8. Kedua Orang Tua saya Bapak Suyud Agung Setiawan dan Ibu Suhartini yang telah mencurahkan keringatnya untuk membiayai pendidikan saya, yang selalu memberikan kasih sayang, doa, dan dukungannnya.
9. Kakek dan Nenek saya Bapak Sumadi dan Ibu Warkinasih yang telah merawat serta memberikan kasih sayang, doa dan semangat sejak saya kecil.
10. Adik saya Hafifah Sukma Latynina yang senantiasa memberi semangat dan doanya.
11. Keluarga besar serta sahabat kontrakan *Console House* (Imron, Raka, Khakim, Akhsin, Arief, Warson, Khamim, Iqbal, Farhan, Fachrizal, Salman dan Jefri) yang telah saling mendukung dalam menyelesaikan skripsi.
12. Kedua sahabatku di perantauan Aulia Rosi Kusuma Wardani dan Berly Maryam Sartono yang selalu menghibur, memberi semangat motivasi hingga terselesaikanya skripsi ini.
13. Kelima sahabatku (Khakim, Fadholi, Inta, Rachel dan Ririn) yang selalu memberi semangat motivasi dalam menyelesaikan skripsi.

14. Teman-teman saya di jurusan Ilmu Komputer, Fakultas MIPA, serta teman-teman di Universitas Negeri Semarang yang telah memberikan semangat dan dukungannya.
15. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu persatu, terimakasih atas bantuannya.

Semoga dengan membaca skripsi ini dapat memberi manfaat bagi kita semua di masa yang akan datang, Penulis menyadari sepenuhnya bahwa skripsi ini jauh dari kesempurnaan karena keterbatasan, kemampuan, waktu, pengetahuan kami yang masih terbatas. Oleh karena itu, kritik dan saran yang bersifat membangun sangat penulis harapkan untuk lebih sempurnanya penyusunan skripsi ini.

Atas semua perhatian dari segala pihak yang telah membantu penulis dalam menyusun skripsi ini, penulis ucapkan terima kasih.

Semarang, 24 Juni 2020

Penulis,

Hafid Alpin Al Gazni

ABSTRAK

Gazni H.A.A. 2020. Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma *K-Means* dan *Particle Swarm Optimization* pada Diagnosis Penyakit Ginjal Kronis. Skripsi, Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Budi Prasetiyo, S.Si., M.Kom.

Kata kunci: Optimasi Algoritma SVM, *K-Means*, PSO, Ginjal Kronis.

Data mining adalah metode yang digunakan untuk memberikan solusi dalam penyelesaian masalah dalam menggali informasi dari data yang besar. *Data mining* dalam dunia medis bermanfaat untuk mendiagnosis suatu penyakit seperti penyakit ginjal kronis. Klasifikasi merupakan salah satu teknik menggali data yang tersembunyi yang dimiliki oleh *data mining*, ada beberapa metode klasifikasi *data mining*, salah satunya adalah dengan algoritma *Support Vector Machine.* Algoritma *Support Vector Machine* telah membuktikan hasil yang lebih baik dari algoritma KNN, *Decision Tree* dan *Linear Regresion*. Dalam proses klasifikasi, hasil akurasi dan efisiensi waktu yang diperoleh sangat penting. Maka diperlukan optimasi agar meningkatkankan akurasi dan efisiensi waktu saat proses klasifikasi. Optimasi algoritma *Support Vector Machine* yang dilakukan menggunakan algoritma *K-Means* untuk proses *clustering* data kontinu pada dataset dan proses seleksi fitur menggunakan *Particle Swarm Optimization*. Penelitian ini bertujuan untuk mengetahui cara kerja optimasi akurasi dan efisiensi waktu pada algoritma *Support Vector Machine* dan hasil akurasi serta efisiensi waktu yang diperoleh dalam diagnosis penyakit ginjal kronis. Penelitian ini menggunakan *Chronic Kidney Disease Dataset* dari UCI *Machine Learning Repository*. Dari hasil penelitian, metode *Support Vector Machine* dengan menggunakan *K-Means* dan PSO memberikan akurasi rata-rata sebesar 98,80% lebih tinggi jika dibandingkan dengan algoritma *Support Vector Machine* sebesar 55,00% namun lebih rendah 0,12% banding dengan *Support Vector Machine* dengan metode PSO saja. Metode *Support Vector Machine* dengan menggunakan *K-Means* dan PSO memiliki rata-rata waktu pemrosesan sekitar 20 detik lebih cepat dibanding metode *Support Vector Machine* dengan metode PSO yang membutuhkan rata-rata waktu selama 33,87 detik. Percobaan berdasar metode yang digunakan terbukti dapat meningkatkan akurasi klasifikasi sebesar 43,80% dari akurasi yang dihasilkan algoritma *Support Vector Machine* dan menghemat waktu pemrosesan sekitar 20 detik dari algoritma *Support Vector Machine* dengan PSO.

DAFTAR ISI

PERNYATAAN i
PERSETUJUAN PEMBIMBING ii
PENGESAHAN iii
MOTTO DAN PERSEMBAHAN iv
PRAKATA v
ABSTRAK viii
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
DAFTAR LAMPIRAN xiv
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Batasan Masalah 5
1.4 Tujuan Penelitian 5
1.5 Manfaat Penelitian 6
1.6 Sistematika Penulisan Skripsi 6
1.6.1 Bagian Awal Skripsi 6
1.6.2 Bagian Isi Skripsi 6
1.6.3 Bagian Akhir Skripsi 7
BAB 2 TINJAUAN PUSTAKA 9
2.1 *Data Mining* 9
2.1.1 Teknik Pembelajaran *Data Mining* 11
2.1.2 Tahapan *Data Mining* 13
2.2 Klasifikasi 14
2.2.1 Konsep Klasifikasi 14
2.2.2 *Support Vector Machine* 15
2.3 Algoritma *K-Means* 17
2.4 *Particle Swarm Optimization* 19
2.5 *Binary Particle Swarm Optimization* (BPSO) 21

2.6 *Confusion Matrix* 21
2.7 Penelitian Terkait 23
BAB 3 METODE PENELITIAN 25
3.1 Studi Pendahuluan 25
3.2 Pengambilan Data 26
3.3 Tahap Pengolahan Data 27
3.3.1 Tahapan Konversi Data, Transformasi Data dan *Data Cleaning* 27
3.3.2 Tahapan Algoritma *K-Means* 27
3.3.3 Tahapan *Particle Swarm Optimization* 29
3.3.4 Pembagian data *Training* dan data *Testing* 32
3.3.5 Tahapan *Support Vector Machine* 32
3.4 Tahap *Mining* Data 34
3.5 Tahap Evaluasi 34
3.6 Penarikan Kesimpulan 35
BAB 4 HASIL DAN PEMBAHASAN 37
4.1 Hasil Penelitian 37
4.1.1 Hasil Pengambilan Data 37
4.1.2 Pengolahan data 37
4.1.2.1 Missing Value Handling 39
4.1.2.2 Transformasi Data 41
4.1.2.3 Tahap Clustering 42
4.1.2.4 Tahap Seleksi Fitur 46
4.1.2.5 Tahap Pembagian Data 53
4.1.2.6 Tahap Evaluasi dengan Confusion Matrix 53
4.1.3 Tahap *Mining* Data 53
4.1.3.1 Hasil Algoritma SVM 54
4.1.3.2 Hasil K-Means pada SVM 54
4.1.3.3 Hasil PSO pada SVM 55
4.1.3.4 Hasil K-Means dan PSO pada SVM 56
4.2 Implementasi Sistem 58
4.2.1 Tahap Pembuatan Sistem 58

4.2.2 Implementasi Algoritma 58
4.2.2.1 Implementasi Algoritma SVM 58
4.2.2.2 Implementasi Algoritma K-Means 59
4.2.2.3 Implementasi PSO pada Algoritma SVM 60
4.2.2.4 Implementasi Algoritma K-Means dan PSO pada Algortima SVM 61
4.2.3 Implementasi *User Interface* 63
4.3 Pembahasan 69
BAB 5 KESIMPULAN DAN SARAN 72
5.1 Kesimpulan 72
5.2 Saran 74
DAFTAR PUSTAKA 75
LAMPIRAN 79

DAFTAR TABEL

Tabel 2.1 *Confusion Matrix* 22
Tabel 2.2 Penelitian terkait *State of the art*..... 23
Tabel 3.1 Deskripsi atribut *Chronic Kidney Disease Dataset* 26
Tabel 3.2. Pengujian *Confusion Matrix* 35
Tabel 4.1 *Dataset Chronic Kidney Disease* dengan format *.csv*..... 39
Tabel 4.2 Sampel *Missing Value* pada *Dataset Chronic Kidney Disease* 40
Tabel 4.3 Nilai untuk Mengisi *Missing Value* 41
Tabel 4.4 Transformasi Atribut..... 42
Tabel 4.5 Hasil Transformasi data 42
Tabel 4.6 Sampel data untuk proses *clustering*..... 43
Tabel 4.7 Hasil perhitungan jarak data terhadap pusat *cluster* 44
Tabel 4.8 Hasil Pengelompokkan untuk iterasi 1 45
Tabel 4.9 Data sampel yang akan diseleksi fitur 47
Tabel 4.10 Evaluasi dengan *confusion matrix* 53
Tabel 4.11 Akurasi Algoritma SVM 54
Tabel 4.12 Hasil akurasi setiap k-*cluster* 55
Tabel 4.13 Hasil akurasi dan waktu pemrosesan algoritma SVM dan PSO..... 56
Tabel 4.14 Hasil akurasi dan waktu pemrosesan algoritma SVM dengan *K-Means* dan PSO 57
Tabel 4.15 Hasil tiap metode yang digunakan 70

DAFTAR GAMBAR

Gambar 2.1 Proses *Data Mining* 13
Gambar 2.2 *Basic K-Means Algorithm* 18
Gambar 3.1 Tahapan penelitian 25
Gambar 3.2 Diagram alur algoritma *K-Means* Clustering 28
Gambar 3.3 Diagram alur algoritma PSO 30
Gambar 4.1 *Dataset Chronic Kidney Disease* dengan format *.arff* 38
Gambar 4.2 *Source code* Algoritma SVM 59
Gambar 4.3 *Source code* Algoritma *K-Means* 60
Gambar 4.4 *Source code* PSO pada Algoritma SVM 60
Gambar 4.5 *Source code* Implementasi *K-Means* dan PSO pada Algoritma SVM 61
Gambar 4.6 Tampilan *Register User* baru 64
Gambar 4.7 Tampilan *Login User* 65
Gambar 4.8 Tampilan *menu Home* 65
Gambar 4.9 Tampilan *Original Datasets* 66
Gambar 4.10 Tampilan *Handled Datasets* 66
Gambar 4.11 Tampilan *sub menu K-Means* 67
Gambar 4.12 Tampilan *sub menu* PSO 67
Gambar 4.13 Tampilan *sub menu Purposed Method* 68
Gambar 4.14 Tampilan *menu* About 68
Gambar 4.15 Tampilan *sub menu Developer Profile* 68

DAFTAR LAMPIRAN

Lampiran 1. *Source Code views.py* 80
Lampiran 2. *Source Code index.html* 86
Lampiran 3. *Source Code datasets-original.html* 89
Lampiran 4. *Source Code datasets-handled.html* 91
Lampiran 5. *Source Code kmeans.html* 93
Lampiran 6. *Source Code pso.html* 95
Lampiran 7. *Source Code purposed.html* 96
Lampiran 8. *Source Code about.html* 97
Lampiran 9. *Source Code author-profile.html* 98
Lampiran 10. *Source Code login.html* 99
Lampiran 11. *Source Code register.html* 101
Lampiran 12. *Chronic Kidney Disease Datasets* 103
Lampiran 13. Surat Keputusan Penetapan Dosen Pembimbing 117

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pada zaman teknologi sekarang ini, data yang dihasilkan oleh komputer telah terdistribusi secara global dan memiliki volume cukup yang besar. *Datasets* termasuk data yang memiliki jumlah data yang banyak dan sulit untuk dikelola, diperoleh, disimpan, dianalisis dan divisualisasikan (Sreedhar, *et al.*, 2017: 1). Sekumpulan data tersebut apabila tidak digunakan maka hanya menjadi kumpulan data yang tidak bermanfaat. untuk mengekstrak informasi inti dari kumpulan data tersebut maka digunakan metode *data mining* (Agarwal, 2013: 1).

Data mining memungkinkan orang untuk menemukan dan menafsirkan pola-pola yang dapat membantu dalam membuat keputusan berdasarkan informasi yang telah diproses menggunakan teknik *data mining* (North, 2012: 3). *Data mining* digunakan dalam berbagai bidang, seperti: telekomunikasi, kesehatan, bioinformatika, perbankan, pemasaran, biologi, asuransi, perencanaan kota, studi gempa bumi, klasifikasi dokumen web dan layanan transportasi (Sreedhar *et al.*, 2017: 1-2).

Dalam bidang kesehatan, *data mining* dapat digunakan untuk diagnosis penyakit seperti diagnosis penyakit ginjal kronis (Kunwar *et al.*, 2016: 1-6), diabetes melitus (Kumari & Singh, 2013: 1-3), kanker payudara (Gupta *et al.*, 2011: 1-8), penyakit jantung (Soni *et al.*, 2011: 1-6) dan lain-lain. Salah satu teknik *data mining* yang digunakan untuk memprediksi sebuah keputusan, yaitu klasifikasi

(Agarwal, 2013: 1). Klasifikasi juga dapat digunakan untuk memprediksi suatu penyakit dari rekam medis pasien (Mirqotussa'adah *et al*., 2017: 136). Teknik klasifikasi memiliki beberapa algoritma antara lain klasifikasi berbasis *fuzzy logic*, klasifikasi *bayes*, *decision tree*, *support vector machine*, *artificial neural network*, *k-nearest neighbour*, (Suyanto, 2017: 126).

Dalam bidang medis, penyakit ginjal terbagi menjadi dua jenis, yaitu penyakit ginjal kronis *(Chronic Kidney Disease)* dan penyakit ginjal akut *(Acute Kidney Disease)* yang terjadi ketika ginjal tidak dapat menyaring limbah dari darah (Rubini & Eswaran, 2015: 49). Penderita penyakit ginjal kronis kian bertambah seiring pesatnya laju pertumbuhan penduduk di seluruh dunia, bahkan dalam kurun waktu 10 tahun, *Global Burden of Disease* mencatat bahwa penyakit CKD naik 9 peringkat dari awalnya peringkat 27 ke peringkat 18 (Fadilla et al., 2018: 3398). Tes laboratorium diperlukan dalam proses deteksi dini penyakit CKD, untuk mengetahui kadar serum *creatinine*, kadar *urea*, *albumin* dan menentukan nilai filtrasi *glomerulus* (GFR) yang menjadi indikator kuat diagnosis penyakit ginjal kronis (Salekin & Stankovic, 2016: 264). Pemeriksaan medis yang dilakukan pada pasien mengasilkan data yang sangat besar. Namun, dalam volume data yang sangat besar masih terdapat beberapa data yang hilang, maka dari itu diperlukan teknik klasifikasi yang bagus dan menghasilkan akurasi tinggi untuk mendeteksi penyakit ginjal kronis berdasarkan *dataset* (Abedalkhader & Abdulrahman, 2017: 55).

Menurut Charleonnan *et al.* (2016: 80-83) *Support Vector Machine* memiliki akurasi yang lebih baik dibandingkan dengan algoritma KNN, *Decision Tree* dan *Linear Regresion*. *Support Vector Machine* (SVM) adalah salah satu

algoritma yang populer dan efektif dalam beberapa tahun terakhir. SVM digunakan untuk menemukan *hyperplane* yang optimal yang memisahkan antara dua kelas, sehingga bisa memberi kemampuan generalisasi yang baik. Untuk menemukan *hyperplane* optimal, kita biasanya mengambil sebagian besar data label. Tetapi pada data percobaan yang bersekala besar dapat menjadikan tingkat kompleksitas pada proses komputasi semakin tinggi (Yao *et al.*, 2013: 1).

Akurasi dan efisiensi waktu sangat penting dalam pengklasifikasian (Mirqotussa'adah *et al.*, 2017: 136). Untuk meningkatkan efisiensi dapat dilakukan *clustering* pada *dataset* dimana *clustering* tersebut bertujuan untuk mengelompokkan objek ke dalam *cluster* dimana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain. Dengan kata lain, metode ini berusaha untuk meminimalkan variasi antar data yang ada di dalam suatu *cluster* dan memaksimalkan variasi dengan data yang ada di *cluster* lainnya. *K-Means* adalah algoritma pengelompokan dalam bidang *data mining*. Ini digunakan untuk menganalisis *cluster*, dan memiliki efisiensi tinggi pada pembagian data terutama dalam *dataset* besar (Yao *et al.*, 2013: 3).

Sedangkan untuk meningkatkan akurasi, dapat dilakukan dengan menghapus subset fitur yang tidak relevan dan berlebihan pada *dataset* (Tran *et al.*, 2014: 605). Seleksi fitur merupakan tahap *preprocessing* dari *data mining*. *Particle Swarm Optimization* (PSO) merupakan salah satu optimasi *metaheuristik* untuk seleksi fitur. Contoh lain dari algoritma *metaheuristik* adalah *Ant Colony Optimization* (ACO), Algoritma Genetika, *Evolutionary Programming* (EP) &

Differential Evolution (DE) (Ashari *et al.*, 2016: 149). PSO telah terbukti lebih kompetitif daripada algoritma genetika dan algoritma C4.5 dalam beberapa kasus, terutama di bidang optimasi (Sousa *et al.*, 2004: 1). PSO masih terus dikembangkan untuk mendorong proses optimasi yang lebih baik lagi.

Berdasarkan uraian permasalahan di atas, maka penelitian ini berfokus untuk meningkatkan akurasi *Support Vector Machine* menggunakan *K-Means Clustering* berdasarkan seleksi fitur *Particle Swarm Optimazion* dengan judul **"Optimasi Algoritma *Support Vector Machine* Berbasis Algoritma *K-Means* dan *Particle Swarm Optimization* pada Diagnosis Penyakit Ginjal Kronis".**

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana cara kerja *K-Means Clustering* dan *Particle Swarm Optimization* dalam meningkatkan akurasi dan efisiensi waktu dari *Support Vector Machine* untuk mendiagnosis penyakit ginjal kronis?
2. Bagaimana hasil akurasi dan efisiensi waktu yang diperoleh terhadap proses diagnosis penyakit ginjal kronis menggunakan algoritma *Support Vector Machine* yang dioptimasi menggunakan algoritma *K-Means Clustering* dan seleksi fitur menggunakan *Particle Swarm Optimization*?

1.3 Batasan Masalah

Pada penelitian ini diperlukan batasan-batasan agar tujuan penelitian dapat tercapai. Adapun batasan masalah yang dibahas pada penelitian ini adalah:

1. Algoritma klasifikasi yang digunakan dalam penelitian ini adalah *Support Vector Machine.*
2. Data yang digunakan dalam penelitian ini adalah dataset *Chronic Kidney Disease Data Set* yang diambil dari UCI *Machine Learning Repository*.
3. Algoritma *K-Means Clustering* digunakan untuk mengatasi persebaran data dan meningkatkan efisiensi waktu.
4. *Particle Swarm Optimization* digunakan untuk mengatasi masalah fitur yang tidak relevan dan berlebihan.
5. Jenis PSO yang digunakan adalah *Binary Particle Swarm Optimization* (BPSO).
6. Bahasa pemrograman yang digunakan dalam penelitian ini adalah *Python*.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah sebagai berikut.

1. Untuk mengetahui cara kerja *K-Means Clustering* dan *Particle Swarm Optimization* dalam meningkatkan akurasi dan efisiensi waktu dari *Support Vector Machine* untuk mendiagnosis penyakit ginjal kronis.
2. Untuk mengetahui hasil akurasi dan efisiensi waktu yang diperoleh terhadap proses diagnosis penyakit ginjal kronis menggunakan algoritma *Support Vector Machine* yang dioptimasi menggunakan algoritma *K-*

Means Clustering dan seleksi fitur menggunakan *Particle Swarm Optimization*.

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut.

1. Dapat memprediksi penyakit ginjal kronis dengan menerapkan algoritma *Support Vector Machine* yang telah dioptimasi menggunakan algoritma *K-Means Clustering* dan seleksi fitur menggunakan *Particle Swarm Optimization*.
2. Dalam lingkungan akademis diperoleh kontribusi pengetahuan tentang cara kerja untuk meningkatkan akurasi dan hasil akurasi yang diperoleh pada optimasi algoritma *Support Vector Machine* dengan memanfaatkan algoritma *K-Means Clustering* dan seleksi fitur menggunakan *Particle Swarm Optimization*.

1.6 Sistematika Penulisan Skripsi

Secara garis besar sistematika penulisan skripsi ini dibagi menjadi tiga bagian yaitu sebagai berikut:

1.6.1 Bagian Awal Skripsi

Bagian awal skripsi terdiri dari halaman judul, halaman pengesahan, halaman pernyataan, halaman motto dan persembahan, abstrak, kata pengantar, daftar isi, daftar gambar, daftar tabel dan daftar lampiran.

1.6.2 Bagian Isi Skripsi

Bagian isi skripsi terdiri dari lima bab yaitu sebagai berikut.

1. BAB 1: PENDAHULUAN

   Bab ini terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat penelitian, serta sistematika penulisan skripsi.

2. BAB 2: TINJAUAN PUSTAKA

   Pada bab ini terdiri dari landasan teori dan penelitian terkait.

3. BAB 3: METODE PENELITIAN

   Pada bab ini terdiri dari studi literatur, pengambilan data, analisis data, metode yang digunakan, perancangan sistem, dan penarikan kesimpulan.

4. BAB 4: HASIL DAN PEMBAHASAN

   Pada bab ini menjelaskan tentang hasil dan pembahasan yang diperoleh dari penelitian optimasi algoritma *Support Vector Machine* menggunakan algoritma *K-Means* dan seleksi fitur *Particle Swarm Optimization* pada diagnosis penyakit kanker payudara.

5. BAB 5: PENUTUP

   Pada bab ini menguraikan tentang kesimpulan dari permasalahan dan saran untuk kemajuan dan keperluan pengembangan penelitian.

1.6.3 Bagian Akhir Skripsi

Pada bagian ini skripsi berisi daftar pustaka yang merupakan informasi mengenai buku-buku, sumber pustaka dan referensi yang digunakan penulis serta lampiran-lampiran yang mendukung dalam penulisan skripsi ini.

BAB 2

TINJAUAN PUSTAKA

2.1 *Data Mining*

Data mining adalah kegiatan menemukan pola yang menarik dari data dalam jumlah besar. Data dapat disimpan dalam *database*, data *warehouse*, atau penyimpanan informasi lainnya (Han, 2012: 8). Menurut Molina (2009: 1093) *Data mining* adalah proses pemeriksaan data dan menemukan aturan atau model sederhana yang dapat meringkas data. Dalam *data mining*, data dengan jumlah yang besar disimpan secara elektronik dan proses pencariannya harus otomatis. Pola-pola yang ditemukan harus mengarah pada keuntungan, seperti memberikan keuntungan ekonomi (Witten, 2011: 5).

Data mining dapat diterapkan pada semua jenis data, selama data tersebut berguna untuk target *system*/aplikasi. Secara umum, bentuk data untuk *data mining* adalah data dari *database*, data *warehouse* dan data transaksional. *Data mining* juga dapat diterapkan ke bentuk data lain, seperti data urutan, grafik atau data jaringan, data spasial, data spasial, data teks dan data multimedia (Han, 2012: 8). Secara umum, *data mining* digunakan dalam kegiatan deskriptif dan prediktif. Deskriptif berarti *data mining* digunakan unutk mencari pola-pola yang dapat dipahami manusia yang menjelaskan karakteristik data. Sedangkan prediktif berarti *data mining* digunakan untuk membentuk sebuah model pengetahuan yang akan digunakan untuk melakukan prediksi (Suyanto, 2017: 3).

Menurut Deshpande (2015: 11), terdapat beberapa teknik pengolahan data dalam *data mining*, seperti sebagai berikut.

1. Klasifikasi

   Memprediksi apakah suatu data termasuk ke dalam salah satu kelas data yang telah ditetapkan. Contoh algortima klasifikasi adalah *decision tree, neural network, Bayesian model, k-nearest neighboard.*

2. Regresi

   Memprediksi label target numerik dari suatu titik data (*data point*). Contoh algoritma dari regresi adalah *Linear Regression, Logistic regression.*

3. *Anomaly Detection*

   Memprediksi apakah titik data adalah *outlier* jika dibandingkan dengan tiitk data lain. Contoh algoritmanya adalah *Distance Based, Density Based, Local Outlier Factor* (LOF).

4. *Time Series*

   Memprediksi nilai target untuk masa yang akan datang berdasarkan nilai dimasa lampau. Contoh algoritmanya adalah *Exponential Smoothing, Autoregressive Integrated Moving Average* (ARIMA).

5. Klasterisasi

   Mengidentifikasi kelompok data dalam *dataset* berdasarkan atribut dalam *dataset.* Contoh algoritmanya adalah *K-Means, Density-Based Clustering.*

6. *Association Analysis*

Mengidentifikasi hubungan dalam satu set item berdasarkan data transaksi. Contoh algoritmanya adalah *Frequent Pattern Growth (FP-Growth), Apriori algorithm.*

2.1.1 Teknik Pembelajaran *Data Mining*

Teknik yang digunakan dalam *data mining* berkaitan dengan "penemuan" (*discovery*) dan "pembelajaran" (*learning*) yang terbagi dalam tiga metode utama pembelajaran yaitu (Retno, 2017: 7):

1. *Supervised Learning*

Teknik ini melibatkan fase pelatihan dimana data pelatihan historis yang karakternya dipetakan ke hasil-hasil yang telah diketahui dan diolah dalam algoritma *data mining.* Proses ini melatih algoritma untuk mengenali variabel-variabel yang nantinya digunakan sebagai dasar dalam membuat perkiraan ketika diberikan data baru. Contoh *supervised learning* yaitu *linear regression, decision tree, naïve bayes classifier, artificial neural network,* dan *support vector machine*.

2. *Unsupervised Learning*

Teknik ini bergantung pada penggunaan algoritma yang mendeteksi semua pola, seperti *associations* dan *sequences,* yang muncul dari kriteria penting yang spesifik dalam data *input.* Pendekatan ini mengarah pada pembuatan banyak aturan (*rules*) yang mengkarakterisasikan penemuan *associations, clusters,* dan *segments*. Aturan ini kemudian dianalisis untuk menemukan hal-hal yang

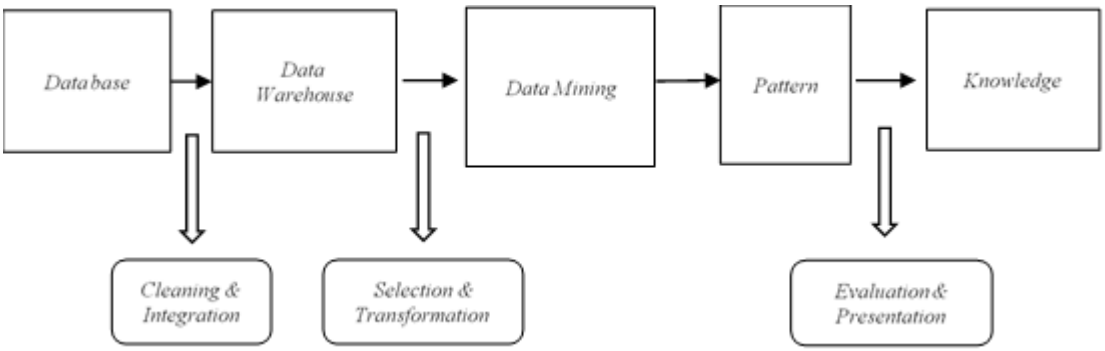
penting. Contoh *unsupervised learning* yaitu *K-Means, Fuzzy C-Means,* dan *Self Organizing Map*.

3. *Reinforcement Learning*

Teknik ini menyerupai kehidupan nyata yaitu seperti "*on-job-training*", dimana seorang pekerja diberikan sekumpulan tugas yang membutuhkan keputusan-keputusan. *Reinforcement Learning* sangat tepat digunakan dalam menyelesaikan masalah-masalah yang sulit dan bergantung pada waktu.

2.1.2 Tahapan *Data Mining*

Proses *data mining* merupakan urutan iteratif dari langkah-langkah yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Proses *Data Mining* (Han, 2012: 7).

Tahap *data mining* ada 7, yaitu:

1. Pembersihan Data *(Data Cleaning)*

   Merupakan proses untuk menangani nilai yang hilang, mengurangi *noise* ketika mengidentifikasi *outlier* data dan memperbaiki data yang tidak konsisten. *Missing value, noise* dan data yang tidak konsisten akan membuat data menjadi tidak akurat.

2. Integrasi Data (*Data Intregation*)

   Merupakan proses untuk menggabungkan data dari beberapa *database.* Integrasi data dapat meningkatkan akurasi dan kecepatan proses penambangan data. Namun, integrasi data harus dilakukan dengan hati-hati agar mengurangi dan menghindari redudansi data dan inkonsistensi data.

3. Seleksi Data (*Data Selection*)

   Merupakan proses pengambilan data yang relevan dari *database* untuk dianalisis.

4. Transformasi Data (*Data Transformation*)

   Merupakan proses untuk mengubah data kedalam format yang dapat diproses oleh *data mining.* Sehingga proses penambangan data lebih efisien dan pola yang ditemukan mungkin lebih mudah dipahami.

5. Proses *Mining*

   Merupakan proses utama dimana metode diterapkan untuk mendapatkan pola dan pengetahuan dari data.

6. Evaluasi Pola (*Evaluation Pattern*)

   Untuk mengidentifikasi pola yang benar-benar berharga.

7. Presentasi Pengetahuan (*Knowledge Presentation*)

   Untuk menyajikan pengetahuan yang didapat kepada pengguna.

2.2 Klasifikasi

2.2.1 Konsep Klasifikasi

Klasifikasi adalah suatu proses untuk menyatakan suatu objek data sebagai salah satu kategori (kelas) yang telah didefinisikan sebelumnya (Suyanto, 2017: 115). Menurut Han (2012: 327) klasifikasi adalah metode *data mining* yang dapat digunakan untuk proses pencarian sekumpulan model (fungsi) yang dapat menjelaskan dan membedakan kelas-kelas data atau konsep, yang tujuannya supaya model tersebut dapat digunakan memprediksi objek kelas yang labelnya tidak

diketahui atau dapat memprediksi kecenderungan data-data yang muncul di masa depan.

Klasifikasi memiliki 2 tahap, yaitu tahap *learning* dan tahap klasifikasi (Han, 2012: 328). Pada tahap *learning,* algoritma klasifikasi akan membangun model klasifikasi dengan menganalisis atau belajar dari *dataset training* yang terdiri dari tupel basis data dan label kelas terkait. Pada tahap klasifikasi, model klasifikasi yang telah dibangun pada tahap *learning* akan digunakan untuk mengklasifikasi *dataset. Dataset* yang digunakan merupakan *dataset testing* yang terdiri dari tupel uji dan label kelas terkait. *Dataset testing* berbeda dengan *dataset training,* yang berarti *dataset testing* tidak digunakan untuk membangun model klasifikasi.

2.2.2 *Support Vector Machine*

Support Vector Machine (SVM) pertama kali diusulkan oleh Vladimir Vapnik dan digunakan dalam bidang teori pembelajaran statistik dan minimalisasi risiko struktural (Nayak *et al.*, 2015: 170). SVM telah digunakan dalam berbagai masalah dunia nyata seperti pengkategorian teks, pengenalan digit tulisan tangan, pengenalan nada, klasifikasi gambar dan deteksi objek, analisis data ekspresi *gen-array* mikro, klasifikasi data (Srivastava & Bhambhu, 2010: 1).

Menurut Han (2012: 408), langkah awal suatu algoritma *Support Vector Machine* adalah pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan pada Persamaan (1).

$$W\,.X + b = 0 \tag{1}$$

Dimana W merupakan bobot vektor, yaitu $W = \{w_1, w_2, w_3, \dots, w_n\}$; n merupakan jumlah atribut dan b merupakan suatu skalar yang sering disebut bias.

Jika berdasarkan dua atribut masukan A_1 dan A_2 dengan tupel pelatihan X= (x_1, x_2), dimana $x_1$dan x_2 merupakan nilai atribut dari A_1 dan A_2, dan jika b dianggap sebagai bobot tambahan w_0, maka persamaan *hyperplane* pemisah ditunjukkan pada Persamaan (2).

$$w_0 + w_1x_1 + w_2x_2 = 0 \quad (2)$$

Dengan demikian, titik yang berada di atas *hyperplane* pemisah memenuhi Persamaan (3).

$$w_0 + w_1x_1 + w_2x_2 > 0 \quad (3)$$

Sedangkan titik yang berada dibawah *hyperplane* pemisah memenuhi Persamaan (4).

$$w_0 + w_1x_1 + w_2x_2 < 0 \quad (4)$$

Berdasarkan persamaan di atas, bobot dapat disesuaikan sehingga *hyperplane* yang mendefinisikan sisi dari *margin* seperti pada Persamaan (5).

$$h_1 : w_0 + w_1x_1 + w_2x_2 \geq 0 \, , untuk\ yi = +1$$

$$h_2 : w_0 + w_1x_1 + w_2x_2 \leq 0 \, , untuk\ yi = -1 \quad (5)$$

Untuk menemukan *Maximum Margin Hyperplane*pada SVM dapat menggunakan trik matematika yaitu *Lagrangian formulation* dan kemudian solusi dapat dipecahkan dengan kondisi *Karush-Kuhn-Tucke*r. Berdasarkan *Lagrangian formulation*, *Maksimum Margin Hyperplane* dapat ditulis ulang sebagai suatu batas keputusan (*decision boundary*) seperti pada Persamaan (6).

$$d(X^T) = \sum_{i=1}^{l} y_i a_i \ X_i X^T + b_0 \quad (6)$$

Dimana y_i adalah label kelas *support vector* X_i, X^T merupakan tupel tes. Notasi α_i dan b_0 merupakan parameter numerik yang ditentukan otomatis oleh optimalisasi algoritma SVM dan l merupakan jumlah dari *support vector*.

2.3 Algoritma *K-Means*

Algoritma *K-Means* adalah metode perulangan sederhana untuk mempartisi *dataset* ke beberapa *cluster* yang telah ditentukan oleh pengguna (Wu *et al.*, 2007: 6). Pembentukan model dengan *K-Means* pada data yang besar dapat memaksimalkan hasil pengelompokan (Anggodo *et al.*, 2017: 105). Algoritma *K-Means* mengambil parameter input, k sebagai jumlah *cluster* dan membagi *dataset* dari n objek ke dalam k *cluster*, sehingga objek yang dihasilkan dari satu *cluster* tidak sama dengan *cluster* lain dan mirip dengan objek dari *cluster* yang sama. Dalam Algoritma *K-Means* dimulai dengan objek k yang dipilih secara acak, mewakili pusat *cluster* k awal atau *mean*. Selanjutnya setiap objek ditugaskan ke satu *cluster* berdasarkan kedekatan objek dengan pusat *cluster*. Untuk menghitung jarak setiap data yang ada terhadap masing-masing *centroid* menggunakan rumus *Euclidian* hingga ditemukan jarak yang paling dekat dari setiap data dengan *centroid* (Purwar & Singh, 2015: 5).

Algoritma *K-Means* secara umum dapat dijelaskan pada Gambar 2.2 (Tan *et al.*, 2013: 535).

```
1: Pilih titik k sebagai centroid awal.
2: Ulangi
3: Bentuk k cluster dengan menetapkan setiap titik ke centroid
terdekat.
4: Hitung ulang centroid dari setiap cluster.
5: sampai centroid tidak berubah.
```

Gambar 2.2 *Basic K-Means Algorithm.*

Berikut tahapan dalam proses algoritma *K-Means* (Rahman *et al.*, 2017: 25):

1. Menentukan jumlah k-*cluster* yang akan dibentuk.
2. Membangkitkan k-*centroid* (titik pusat *cluster*) secara acak.
3. Menghitung jarak setiap data terhadap masing-masing *centroid.* Rumus yang digunakan yaitu rumus *Euclidean* dengan Persamaan (7).

$$D(x_2, x_1) = \sqrt{\sum_{i=1}^{n}(x_2 - x_1)}^2 \quad (7)$$

   Dimana:

   $D(x_2, x_1)$ = dimensi data (jarak data)

   x_1 = posisi pusat *cluster*

   x_2 = posisi objek data

4. Lalu dikelompokkan data berdasarkan jarak terdekat antara data dengan *centroid*.
5. Menentukan nilai *centroid* yang baru dengan menghitung rata-rata dari *cluster* yang bersangkutan menggunakan Persamaan (8).

$$C_k = \frac{1}{n_k}\sum d_i \quad (8)$$

Dimana:

n_k = jumlah data dalam *cluster* k

d_i = jumlah nilai jarak yang masuk dalam masing-masing *cluster*

6. Lakukan perulangan dari langkah 3–5 hingga anggota tiap *cluster* tidak ada yang berubah.

2.4 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) adalah teknik optimasi yang dikembangkan oleh Kennedy dan Eberhart pada tahun 1995, populasi dari PSO berdasarkan strategi optimisasi *stokastik*. PSO terinspirasi oleh perilaku sosial kawanan burung, ikan (Xue, 2013: 1657). PSO didasarkan pada prinsip bahwa setiap solusi dapat direpresentasikan sebagai partikel dalam kerumunan. Setiap partikel memiliki posisi dalam ruang pencarian, yang diwakili oleh vektor $x_i = (x_i1, x_i2, \dots, x_iD)$, dimana *D* adalah dimensi ruang pencarian. Partikel bergerak di ruang pencarian untuk mencari solusi optimal. Oleh karena itu, setiap partikel memiliki kecepatan, yang direpresentasikan sebagai $v_i = (v_i1, v_i2, \dots, v_iD)$. Selama melakukan pergerakan, setiap partikel memperbarui posisi dan kecepatannya sesuai dengan pengalaman masing-masing dari tempat sebelumnya. Posisi partikel sebelumnya dicatat sebagai yang terbaik (*pbest)*, dan posisi terbaik yang diperoleh oleh populasi sejauh ini disebut (*gbest)*. Berdasarkan *pbest* dan *gbest*, PSO mencari solusi optimal dengan memperbarui kecepatan dan posisi masing-masing partikel (Xue, 2013: 1657).

Algoritma ini dapat digunakan untuk menyelesaikan masalah optimasi diskrit dan berkelanjutan (Priliani *et al.*, 2018: 119). Selain itu, PSO

mempertahankan pengetahuan tentang solusi yang baik dari semua partikel. Karena kelebihannya, maka PSO baru-baru ini telah banyak diterapkan untuk masalah medis (Wang *et al.*, 2014: 15). Dibandingkan dengan algoritma genetika, PSO tidak memerlukan operator seperti *crossover* dan mutasi tetapi PSO hanya memerlukan operator matematika sederhana sehingga komputasi tidak membutuhkan memori yang banyak (Inbarani *et al*, 2014: 176).

Adapun parameter yang digunakan dalam proses algoritma PSO sebagai berikut (Sumathi & Surekha, 2010: 658-659):

1. Jumlah Partikel (*Number of Particle*) merupakan faktor penting dalam proses penyelesaian masalah optimasi pada algoritma PSO.
2. Bobot Inersia (*Inertia Weight*) merupakan faktor penting dalam perilaku konvergensi dari algoritma PSO dan digunakan untuk mengontrol kecepatan partikel.
3. Faktor Pembelajaran (*Learning Factors*). Parameter c_1 merupakan koefisien pengenalan diri, sedangkan parameter c_2 merupakan komponen sosial.
4. Rentang dan dimensi partikel (*Range and Dimension of Particle*). Rentang dan dimensi partikel ditentukan berdasarkan masalah yang akan dioptimalkan.
5. Kecepatan (*Velocity*). Perubahan maksimum pada satu partikel, dapat diambil selama iterasi dan didefinisikan sebagai kecepatan maksimum.
6. Kondisi berhenti (*Stop Condition*). Kondisi berhenti ketika kriteria telah terpenuhi seperti jumlah iterasi yang sudah terpenuhi, solusi yang diterima

ditemukan ketika mencapai toleransi *error*, dan tidak ada perkembangan setelah beberapa iterasi.

2.5 *Binary Particle Swarm Optimization* (BPSO)

Binary Particle Swarm Optimization (BPSO) diperkenalkan oleh Kennedy dan Eberhart pada tahun 1997 (Lee *et al.*, 2008: 1161). BPSO digunakan untuk menyelesaikan masalah optimasi diskrit dan ruang pencarian biner digunakan untuk menemukan solusi optimal global.

Dalam penerapan algortima PSO yang digunakan untuk solusi permasalahan seleksi fitur, maka dapat menggunakan *binary digit* untuk menunjukkan fitur. Fitur yang tidak terpilih dilambangkan dengan 0, sedangkan fitur terpilih dilambangkan dengan 1. Selain itu, dengan memperbarui posisi (x_{id}^{new}) dengan rumus Sigmoid (S) dengan Persamaan (9) (Tran, 2014: 608).

$$S = \frac{1}{1+e^{-v_{id}^{new}}} \tag{9}$$

$$x_{id}^{new} = \begin{cases} 1, if\ Sigmoid(S) > rand\ (0,1) \\ 0, Otherwise \end{cases}$$

2.6 *Confusion Matrix*

Menurut Han *et al.*, (2012: 365), *confusion matrix* adalah sebuah metode yang berguna untuk menganalisis seberapa baik *classifier* mengenali tupel dari kelas yang berbeda. *Confusion matrix* dilakukan dengan menghitung jumlah *predicted class* terhadap *actual class*. Hasil tersebut dinyatakan dalam *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

TP dan TN menyatakan bahwa *classifier* mengenali tupel dengan benar, artinya tupel positif dikenali sebagai positi f dan tupel negatif dikenali sebagai negatif. Sedangkan FP dan FN menyatakan bahwa *classifier* salah dalam mengenali tupel, artinya tupel positif dikenali sebagai negatif dan tupel negatif dikenali sebagai positif. Tabel *confusion matrix* dapat ditunjukan pada Tabel 2.1.

Tabel 2.1. *Confusion Matrix*

Klasifikasi		Kelas hasil prediksi		
		Ya	Tidak	Jumlah
Kelas actual	Ya	TP	FN	P
	Tidak	FP	TN	N

Akurasi merupakan persentase prediksi benar dari total data yang diklasifikasikan. Akurasi menunjukan seberapa baik model mengkorelasikan antara hasil dengan atribut dalam data yang telah disediakan. Pengukuran akurasi dapat dituliskan dengan Persamaan (10).

$$Accuracy = \frac{TP+TN}{P+N} \text{ x } 100\% \qquad (10)$$

Keterangan:

True Positive (TP) : jumlah kasus positif yang diklasifikasikan sebagai positif .

False Positive (FP) : jumlah kasus negatif yang diklasifikasikan sebagai positif.

True Negative (TN) : jumlah kasus negatif yang diklasifikasikan sebagai negatif.

False Negative (FN) : jumlah kasus positif yang diklasifikasikan sebagai negatif.

Positive (P) : jumlah kasus positif

Negative (N) : jumlah kasus negatif

2.7 Penelitian Terkait

Penelitian ini dikembangkan dari beberapa referensi yang memiliki keterkaitan dengan metode dan objek penelitian. Penggunaan referensi ini digunakan untuk memberikan batasan-batasan terhadap metode dan sistem yang nantinya akan dikembangkan lebih lanjut. Referensi dari penelitian yang terkait dengan penelitian yang diusulkan ditunjukan pada Tabel 2.2.

Tabel 2.2 Penelitian terkait dan s*tate of the art*

Method	Dataset	Feature Selector	Feature Extractor	Classifier	Validation Method	Evaluation Method
Charleonnan	Chronic Kidney Disease	-	-	SVM, KNN, Decission Tree, Linear Regression	5-Fold X Validation	Confusion Matrix
Ardjani	Corpus TMIT	PSO	-	SVM	10-Fold X Validation	Confusion Matrix
Somantri	Data dokumen jurnal mahasiswa D3 Teknik Komputer Politeknik Harapan Bersama Tegal	-	*K-Means*	SVM	10-Fold X Validation	AUC & ROC
Chamidah	Dataset Liver, Breast Cancer, Heart Disease	-	*K-Means*	SVM	5-Fold X Validation	-
Purposed Method	Chronic Kidney Disease	PSO	*K-Means*	SVM	-	Confusion Matrix

Charleonnan (2016) dalam penelitiannya menujukan bahwa SVM dapat memberikan hasil akurasi yang lebih baik dibandingkan KNN, *Decission Tree* dan *Linear Regression* ketika digunakan untuk klasifikasi penyakit ginjal kronis. Ardjani (2010) dalam penelitianya menunjukan penerapan PSO terhadap SVM sebagai seleksi fitur dan menghasilkan akurasi sebesar 86,50%. Soemantri (2016) dalam penelitianya menggunakan algoritma *K-Means* untuk optimasi algoritma SVM yang menghasilkan akurasi sebesar 86,21%. Chamidah (2018) dalam penelitianya menggunakan *K-Means* sebagai ekstraktor ciri yang tidak terlalu banyak menurunkan akurasi namun signifikan dalam menurunkan waktu komputasi.

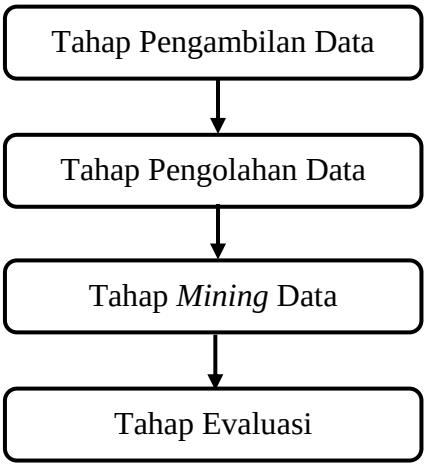
Penelitian ini menggunakan *Chronic Kidney Disease Dataset* menggunakan PSO sebagai seleksi fitur, *K-Means* sebagai ekstraktor ciri dan menggunakan SVM sebagai *classifier.* Metode ini dievaluasi menggunakan *Confusion Matrix.*

BAB 3

METODE PENELITIAN

3.1 Studi Pendahuluan

Pada tahap ini, akan dilakukan pengumpulan data, keterangan, informasi dan teori dalam buku serta rujukan dari artikel, jurnal, dan karya ilmiah lain. Data tersebut memiliki keterkaitan dengan objek dan algoritma yang digunakan dalam penelitian. Bahan referensi yang digunakan mencakup *Support Vector Machine* sebagai algoritma klasifikasi, *K-Means* sebagai proses *clustering* pada data kontinu, dan *Particle Swarm Optimization* sebagai seleksi fitur yang kemudian diterapkan untuk mendiagnosis penyakit ginjal kronis menggunakan *Chronic Kidney Disease Datasets* yang diperoleh dari UCI *machine learning repository*. Dalam peningkatan akurasi algoritma SVM menggunakan algoritma *K-Means* dan PSO untuk mendiagnosis penyakit ginjal kronis memiliki empat tahap, antara lain: Tahap pengambilan data, tahap pengolahan data, tahap *mining* dan tahap evaluasi seperti ditunjukan pada Gambar 3.1.



Gambar 3.1 Tahapan penelitian

3.2 Pengambilan Data

Pada penelitian ini, menggunakan *dataset Chronic Kidney Disease* (CKD) yang diambil dari UCI *Machine Learning Repository.* Data ini bersifat publik, sehingga bisa digunakan oleh siapapun. *Dataset Chronic Kidney Disease* terdiri dari 25 atribut dengan 11 atribut bertipe numerik dan 14 atribut bertipe nominal seperti yang ditunjukkan pada Tabel 3.1 *Dataset Chronic Kidney Disease* memiliki 400 sampel data dengan *missing value* lebih dari 15%. Tabel 3.1 menunjukan deskripsi dari *Chronic kidney disease dataset.*

Tabel 3.1 Deskripsi atribut *Chronic Kidney Disease Dataset*

No	Atribut	Deskripsi	Tipe
1	*Age*	Usia	Numerik
2	*Bp*	*blood pressure*	Numerik
3	*Sg*	*specific gravity*	Nominal
4	*Al*	*Albumin*	Nominal
5	*Su*	*Sugar*	Nominal
6	*Rbc*	*red blood cells*	Nominal
7	*Pc*	*pus cell*	Nominal
8	*Pcc*	*pus cell clumps*	Nominal
9	*Ba*	*Bacteria*	Nominal
10	*Bgr*	*blood glucose random*	Numerik
11	*Bu*	*blood urea*	Numerik
12	*Sc*	*serum creatinine*	Numerik
13	*Sod*	*Sodium*	Numerik
14	*Pot*	*Potassium*	Numerik
15	*Hemo*	*Haemoglobin*	Numerik
16	*Pcv*	*packed cell volume*	Numerik
17	*Wc*	*white blood cell count*	Numerik
18	*Rc*	*red blood cell count*	Numerik
19	*Htn*	*Hypertension*	Nominal
20	*Dm*	*diabetes mellitus*	Nominal
21	*Cad*	*coronary artery disease*	Nominal
22	*Appet*	*Appetite*	Nominal
23	*Pe*	*pedal edema*	Nominal
24	*Ane*	*Anemia*	Nominal
25	*Class*	*Class*	Nominal

3.3 Tahap Pengolahan Data

Pada penelitian ini dilakukan dalam beberapa tahapan, dimulai dari tahapan menkonversikan data dari ekstensi *.arff* menjadi *.csv,* tahapan transformasi data dari nominal ke numerik, *data cleaning*, tahapan *K-Means*, tahapan PSO serta tahapan klasifikasi menggunakan SVM.

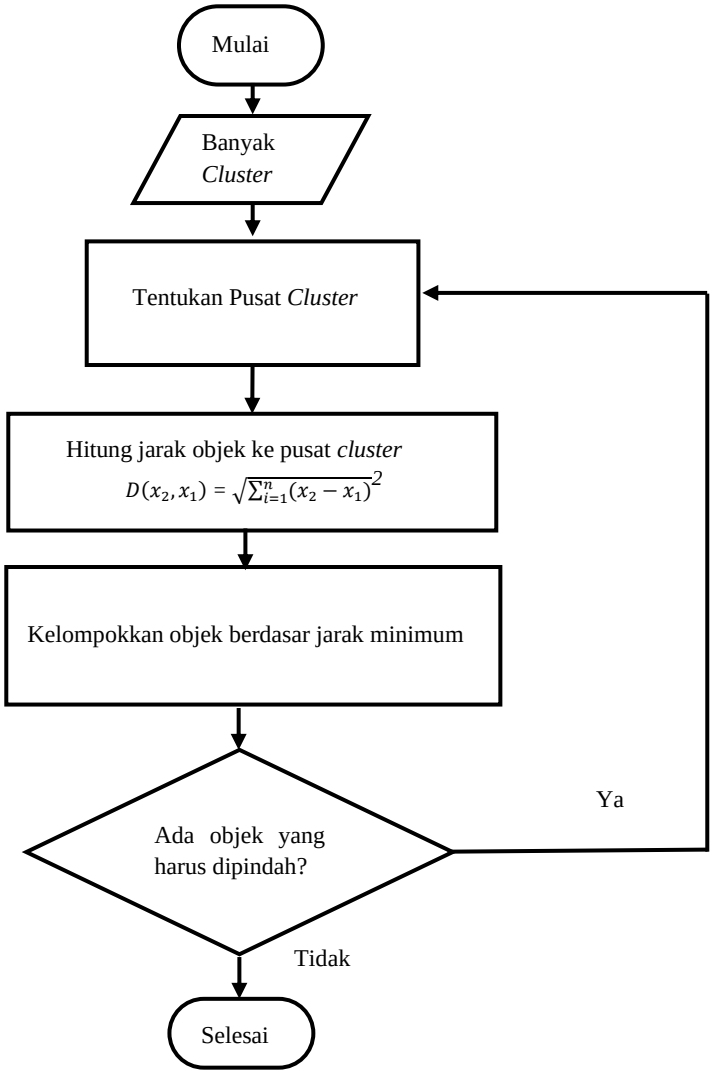
3.3.1 Tahapan Konversi Data, Transformasi Data dan *Data Cleaning*

Pada tahap ini dilakukan konversi data dari ekstensi *.arff* menjadi *.csv* menggunakan aplikasi WEKA (*Waikato Environment of Knowledge Analysis)*. Setelah dilakukan konversi data kemudian data di transformasi dari data yang bertipe nominal ke data bertipe numerik. Hal ini dilakukan agar mempermudah pengolahan *dataset* menggunakan bahasa pemrograman *Python.* Tahap selanjutnya adalah melakukan *cleaning* data dengan melakukan pengisian *missing value* pada *dataset*. Pengisian *missing value* dapat dapat ditangani dengan menggunakan metode *mean* (rata-rata), *median* (nilai tengah) dan *most frequent* (nilai yang sering muncul). Pada penelitian ini, *missing value* di isi menggunakan metode *most frequent* atau nilai yang sering muncul. Metode *most frequent* digunakan karena separuh dari dataset merupakan data yang bertipe nominal.

3.3.2 Tahapan Algoritma *K-Means*

Algoritma *K-Means* menurut Rajeshinigo dan Jebamalar (2015: 2757) digunakan untuk proses *clustering* pada atribut yang memiliki data kontinu menjadi kategorikal berupa *cluster.* Untuk menentukan jumlah k-*cluster* dilakukan dengan percobaan k = 2, 3, 4, 5. Untuk menilai kualitas jumlah k-*cluster*, peneliti

menggunakan klasifikasi SVM dan dipilih yang memiliki akurasi tertinggi yaitu dibagi menjadi 2 *cluster*. Adapun tahapan dalam proses algoritma *K-Means* seperti ditunjukan pada Gambar 3.2.



Gambar 3.2 Diagram alir algoritma *K-Means* Clustering (Hermawati, 2013).

1) Menentukan banyak *cluster* yang akan dibentuk.
2) Menentukan k-*centroid* (titik pusat *cluster*) secara acak.
3) Menghitung jarak setiap data terhadap masing-masing *centroid.* Rumus yang digunakan yaitu rumus *Euclidean* dengan Persamaan (11).

$$D(x_2, x_1) = \sqrt{\sum_{i=1}^{n}(x_2 - x_1)}^2 \quad (11)$$

Dimana:

$D(x_2, x_1)$ = dimensi data

x_1 = posisi titik 1

x_2 = posisi titik 2

4) Mengelompokkan data berdasarkan jarak terdekat antara data dengan *centroid*.
5) Menentukan nilai *centroid* yang baru dengan menghitung rata-rata dari *cluster* yang bersangkutan menggunakan Persamaan (12).

$$C_k = \frac{1}{n_k} \sum d_i \quad (12)$$

Dimana:

n_k = jumlah data dalam *cluster* k

d_i = jumlah nilai jarak yang masuk dalam masing-masing *cluster*

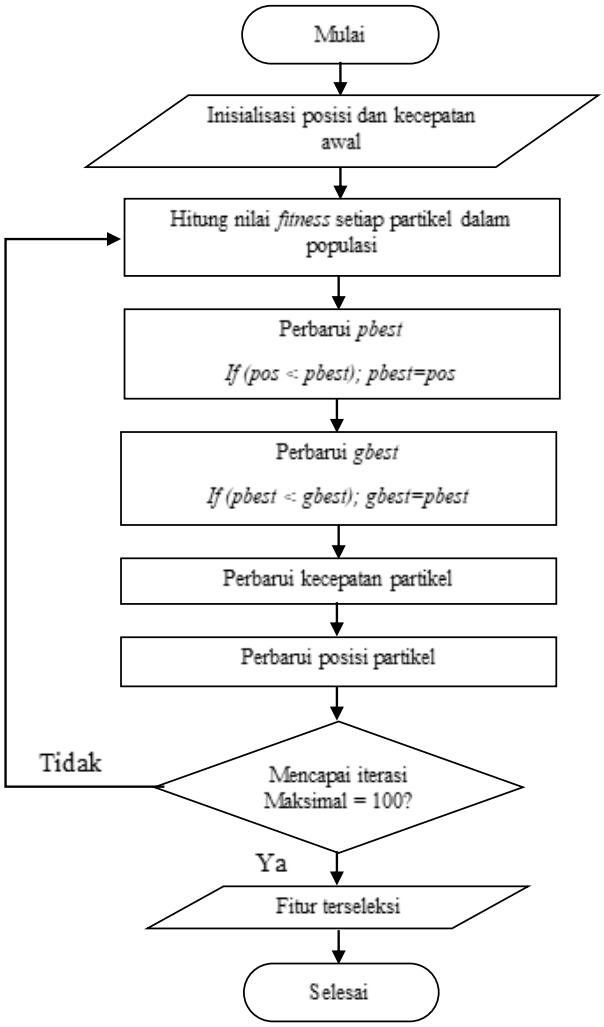
6) Melakukan perulangan dari langkah 3–5 hingga anggota tiap *cluster* tidak ada yang berubah.

Data yang dihasilkan dari proses *K-Means* ini akan digunakan pada proses selanjutnya yaitu proses seleksi fitur menggunakan algoritma PSO.

3.3.3 Tahapan *Particle Swarm Optimization*

Setelah dilakukan proses *clustering* akan dilakukan seleksi fitur menggunakan algoritma *Particle Swarm Optimization* (PSO). Adapun tahapan algoritma *Particle Swarm Optimization* (PSO) yang dilakukan seperti ditunjukan pada Gambar 3.3.

PSO sebagai algoritma yang digunakan untuk optimasi, menyediakan prosedur pencarian berbasis populasi dimana individu disebut dengan partikel (Sumathi & Surekha, 2010: 655). Partikel yang mewakili solusi kandidat dapat berpindah ke posisi optimal dengan memperbarui posisi dan kecepatannya (Arifin, 2017: 158).



Gambar 3.3 Diagram alir algoritma PSO.

1) Inisialisasi kecepatan, posisi, *pbest*, *gbest* setiap partikel secara acak yang diatur dalam rentang yang telah ditentukan.

2) Hitung *fitness* dari partikel menggunakan rumus fungsi *fitness* pada Persamaan (13).

$$Fitness = \frac{\text{jumlah } instance \text{ yang diklasifikasikan benar}}{\text{jumlah } instance} \quad (13)$$

3) Perbarui memori dengan memodernisasi *pBest dan gBest* berdasarkan fungsi *fitness* dengan Persamaan (14) dan Persamaan (15).

$$If\ (pos > pBest): pBest = pos \quad (14)$$

$$If\ (pos > gBest): gBest = pos \quad (15)$$

4) Perbarui kecepatan (v_{id}^{new}) dengan Persamaan (16).

$$v_{id}^{new} = w\,.\,v_{id}^{old} + c_1.r_1\left(pb_{id}^{old} - x_{id}^{old}\right) + c_2.r_2\,(gb_d^{old} - x_{id}^{old}) \quad (16)$$

Dimana v_{id}^{old} adalah kecepatan lama. x_{id}^{old} mewakili posisi partikel i. *pBest* dan *gBest* adalah posisi dengan nilai objektif terbaik yang ditemukan oleh partikel i dan seluruh populasi. w digunakan untuk mengontrol perilaku konvergensi PSO sementara r_1 dan r_2 adalah parameter acak berkisar antara 0 dan 1. c_1 dan c_2 menunjukkan gerakan dan kontrol partikel.

5) Perbarui posisi dengan Persamaan (17).

Perbarui posisi dengan rumus Sigmoid (*S*) dari kecepatan yang telah diperbarui di atas dengan Persamaan (17). Dalam penerapan algortima PSO digunakan untuk solusi permasalahan seleksi fitur, maka dapat menggunakan *binary digit* untuk menunjukkan fitur. Fitur yang tidak terpilih dilambangkan dengan 0, sedangkan fitur terpilih dilambangkan dengan 1.

$$S = \frac{1}{1 + e^{-v_{id}^{new}}} \quad (17)$$

$$x_{id}^{new} = \begin{cases} 1, if\ Sigmoid(S) > rand\ (0,1) \\ 0, Otherwise \end{cases}$$

Dalam penelitian ini menggunakan inisialisasi populasi partikel secara acak yaitu

$$x_{id}^{new} = \begin{cases} 1, if\ U(0,1) > \ 0.5 \\ 0, Otherwise \end{cases}$$

6) Perulangan berhenti ketika mencapai iterasi maksimal. Atribut hasil seleksi didapat dengan mengambil nilai *fitness* tertinggi (dengan nilai *cost* terendah) dari semua iterasi.

Data yang dihasilkan dari seleksi fitur ini akan digunakan pada proses selanjutnya yaitu proses klasifikasi menggunakan algoritma SVM.

3.3.4 Pembagian data *Training* dan data *Testing*

Tahap *split* data atau pembagian data menggunakan metode *random sampling*. *Random sampling* menurut Kerlinger (2006: 188) adalah metode penarikan dari sebuah populasi atau semesta dengan cara tertentu sehingga setiap anggota populasi atau semesta tadi memiliki peluang yang sama untuk terpilih atau terambil. Dengan membagi data latih sebesar 70% dan data uji 30%.

3.3.5 Tahapan *Support Vector Machine*

Hasil dari seleksi fitur, yaitu atribut terpilih akan digunakan dalam proses klasifikasi menggunakan algoritma SVM untuk membuat suatu model dari *set* data *training* yang digunakan untuk memprediksi kelas dari suatu data baru.

Adapun tahapan-tahapan algoritma SVM dalam mengklasifikasikan *dataset* adalah sebagai berikut:

1. Mempersiapkan *data training. Data training* terdiri dari 70% dari keseluruhan *dataset*.
2. Menemukan batas antar kelas. Ketika setiap titik dalam kelas dihubungkan dengan titik yang lain, maka akan muncul garis yang memisahkan antar kelas tersebut. Batas ini dikenal dengan *convex hull*. Setiap kelas memiliki *convex hull* sendiri dan karena kelas (diasumsikan) linear terpisah, *hull* ini tidak saling berpotongan.
3. Menentukan *hyperplane* yang memaksimalkan *margin* antar kelas. Dapat dilakukan dengan cara sebagai berikut:
   a. Pertama, *hyperplane* apapun dinyatakan dalam dua atribut, x_1 dan x_2, seperti ditunjukan pada Persamaan (18).

   $$w \,.\, x + b = 0 \tag{18}$$

   Keterangan:

   w = bobot $\left(w = (w_1, w_2, \dots, w_n)\right)$

   x = jumlah atribut $(x_1, x_2, \dots, x_n)$

   b = bias

   b. Sebuah *hyperplane* optimal, didefinisikan secara unik oleh $b_0 + w_0 . x = 0$. Setelah mendefinisikan *hyperplane* dalam mode ini, dapat menunjukan *margin*-nya sesuai dengan Persamaan (19).

   $$margin = \frac{2}{\sqrt{w_0}} \,.\, w_0 \tag{19}$$

   c. Memaksimalkan kuantitas ini membutuhkan pemograman kuadrat, yang merupakan proses yang berkedudukan kuat dalam teori optimasi matematika. Selanjutnya *w* dapat dengan mudah dinyatakan dalam beberapa

contoh data pelatihan, yang dikenal sebagai *support vector*, sesuai dengan Persamaan (20).

$$|w_0 = \sum y_i x_i| \tag{20}$$

dimana y_i adalah label kelas (+1 dan -1 untuk klasifikasi biner), dan x_i disebut dengan *support vector* sedangkan i adalah koefisien yang nol hanya untuk *support vector*.

4. Menetapkan batas dan *hyperplane*, setiap tes baru dapat diklasifikasikan dengan menghitung di sisi mana hasil data tersebut dalam *hyperplane*. Hal ini dapat ditemukan dengan menggantikan contoh tes x kedalam persamaan *hyperplane* tersebut. Jika terhitung +1, maka termasuk kelas positif dan jika terhitung -1 maka termasuk ke dalam kelas negatif. Kemudian model akan dievaluasi untuk mengetahui hasil akurasi.

3.4 Tahap *Mining* Data

Pada tahap ini dilakukan pengklasifikasian menggunakan algoritma SVM pada *dataset* tanpa menerapkan algoritma *K-Means* dan juga PSO serta dilakukan juga pengklasifikasian pada *dataset* menggunakan algoritma SVM dengan menerapkan algoritma *K-Means* dan algoritma PSO untuk mengetahui peningkatan akurasi yang dihasilkan.

3.5 Tahap Evaluasi

Perhitungan akurasi dilakukan menggunakan *confusion matrix* dengan cara menghitung jumlah data yang diklasifikasikan dengan benar dibagi dengan jumlah prediksi yang dilakukan. Adapun langkahnya adalah sebagai berikut.

1. Masukan hasil pengujian klasifikasi pada tabel *confusion matrix* seperti pada Tabel 3.2.

Tabel 3.2. Pengujian *Confusion Matrix*

Klasifikasi		Kelas hasil prediksi		
		Ya	Tidak	Jumlah
Kelas actual	Ya	TP	FN	P
	Tidak	FP	TN	N

Keterangan:

True Positive (TP) : jumlah kasus positif yang diklasifikasikan sebagai positif .

False Positive (FP) : jumlah kasus negatif yang diklasifikasikan sebagai positif.

True Negative (TN) : jumlah kasus negatif yang diklasifikasikan sebagai negatif.

False Negative (FN) : jumlah kasus positif yang diklasifikasikan sebagai negatif.

Positive (P) : jumlah kasus positif

Negative (N) : jumlah kasus negatif

2. Hitung nilai akurasi menggunakan Persamaan (21).

$$Accuracy = \frac{TP+TN}{P+N} \text{ x } 100\% \quad (21)$$

3. Nyatakan kesimpulan dari hasil akurasi yang diperoleh

3.6 Penarikan Kesimpulan

Penarikan kesimpulan didasarkan pada studi pendahuluan, pengumpulan data dan pengembangan aplikasi serta hasil analisis dari penelitian. Simpulan yang diperoleh adalah tentang bagaimana pengaruh penerapan algoritma *K-Means* dan PSO terhadap tingkat akurasi dan waktu pemrosesan pada algoritma SVM dalam

mendiagnosis penyakit ginjal kronis. Sehingga dapat diketahui pengaruh dan hasil akurasi dan lama waktu pemrosesan dari sebelum dan sesudah penerapan algoritma *K-Means* dan PSO pada algoritma SVM.

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

Dalam peningkatan akurasi algoritma SVM menggunakan algoritma *K-Means* dan PSO untuk mendiagnosis penyakit ginjal kronis memiliki empat tahap, antara lain: pengambilan data, pengolahan data, *mining* dan tahap evaluasi. Adapun penjelasan yang lebih lengkap terkait ke empat tahap tersebut akan diuraikan sebagai berikut.

4.1.1 Hasil Pengambilan Data

Pada penelitian ini, menggunakan *dataset Chronic Kidney Disease* (CKD) yang diambil dari UCI *Machine Learning Repository.* Data ini bersifat publik, sehingga bisa digunakan oleh siapapun. *Dataset Chronic Kidney Disease* terdiri dari 25 atribut dengan 11 atribut bertipe numerik dan 14 atribut bertipe nominal seperti yang ditunjukkan pada Tabel 3.1 *Dataset Chronic Kidney Disease* memiliki 400 sampel data dengan *missing value* lebih dari 15%.

4.1.2 Pengolahan data

Dataset Chronic Kidney Disease yang diperoleh dari UCI *Machine Learning Repository* ini memiliki format *.arff* sebagaimana ditunjukkan pada Gambar 4.1. Pengolahan data dilakukan dengan mengkonversikan ekstensi dari *dataset* ke ekstensi *.csv* yang merupakan standar ASCII *(American Standard Code for Information Interchange)*. Oleh karena itu, format *csv* dijadikan sebagai standar dalam pengolahan data.

```
@relation Chronic_Kidney_Disease

@attribute 'age' numeric
@attribute 'bp'  numeric
@attribute 'sg' {1.005,1.010,1.015,1.020,1.025}
@attribute 'al' {0,1,2,3,4,5}
@attribute 'su' {0,1,2,3,4,5}
@attribute 'rbc' {normal,abnormal}
@attribute 'pc' {normal,abnormal}
@attribute 'pcc' {present,notpresent}
@attribute 'ba' {present,notpresent}
@attribute 'bgr'  numeric
@attribute 'bu' numeric
@attribute 'sc' numeric
@attribute 'sod' numeric
@attribute 'pot' numeric
@attribute 'hemo' numeric
@attribute 'pcv' numeric
@attribute 'wbcc' numeric
@attribute 'rbcc' numeric
@attribute 'htn' {yes,no}
@attribute 'dm' {yes,no}
@attribute 'cad' {yes,no}
@attribute 'appet' {good,poor}
@attribute 'pe' {yes,no}
@attribute 'ane' {yes,no}
@attribute 'class' {ckd,notckd}

@data
48,80,1.020,1,0,?,normal,notpresent,notpresent,121,36,1.2,?,?,15.4,44,7800,5.
2,yes,yes,no,good,no,no,ckd
7,50,1.020,4,0,?,normal,notpresent,notpresent,?,18,0.8,?,?,11.3,38,6000,?,no,
no,no,good,no,no,ckd
62,80,1.010,2,3,normal,normal,notpresent,notpresent,423,53,1.8,?,?,9.6,31,750
0,?,no,yes,no,poor,no,yes,ckd
48,70,1.005,4,0,normal,abnormal,present,notpresent,117,56,3.8,111,2.5,11.2,32
,6700,3.9,yes,no,no,poor,yes,yes,ckd
51,80,1.010,2,0,normal,normal,notpresent,notpresent,106,26,1.4,?,?,11.6,35,73
00,4.6,no,no,no,good,no,no,ckd
60,90,1.015,3,0,?,?,notpresent,notpresent,74,25,1.1,142,3.2,12.2,39,7800,4.4,
yes,yes,no,good,yes,no,ckd
68,70,1.010,0,0,?,normal,notpresent,notpresent,100,54,24.0,104,4.0,12.4,36,?,
?,no,no,no,good,no,no,ckd
24,?,1.015,2,4,normal,abnormal,notpresent,notpresent,410,31,1.1,?,?,12.4,44,6
900,5,no,yes,no,good,yes,no,ckd
52,100,1.015,3,0,normal,abnormal,present,notpresent,138,60,1.9,?,?,10.8,33,96
00,4.0,yes,yes,no,good,no,yes,ckd
```

Gambar 4.1 *Dataset Chronic Kidney Disease* dengan format *.arff*

Pengkonversian ekstensi dilakukan dengan menggunakan salah satu tools data *mining*, yaitu WEKA (*Waikato Environment for Knowledge Analysis).*

Tampilan data dengan ekstensi *.csv* dalam *Microsoft Excel* dapat dilihat pada Tabel 4.1.

Tabel 4.1 *Dataset Chronic Kidney Disease* dengan format *.csv*

Id	Age	bp	Sg	al	su	…	dm	Cad	appet	pe	Ane	Class
0	48	80	1.020	1	0	…	yes	No	good	no	No	Ckd
1	7	50	1.020	4	0	…	no	No	good	no	No	Ckd
2	62	80	1.010	2	3	…	yes	No	poor	no	Yes	Ckd
3	48	70	1.005	4	0	…	no	No	poor	yes	Yes	Ckd
4	51	80	1.010	2	0	…	no	No	good	no	No	Ckd
5	60	90	1.015	3	0	…	yes	No	good	yes	No	Ckd
6	68	70	1.010	0	0	…	no	No	good	no	No	Ckd
7	24	?	1.015	2	4	…	yes	No	good	yes	No	Ckd
8	52	100	1.015	3	0	…	yes	No	good	no	Yes	Ckd
9	53	90	1.020	2	0	…	yes	No	poor	no	Yes	Ckd
…	...	…	…	…	…	…	…	…	…	…	…	…
391	41	80	1.025	0	0	…	no	No	good	no	No	Notckd
392	52	80	1.025	0	0	…	no	No	good	no	No	Notckd
393	36	80	1.025	0	0	…	no	No	good	no	No	Notckd
394	57	80	1.020	0	0	…	no	No	good	no	No	Notckd
395	43	60	1.025	0	0	…	no	No	good	no	No	Notckd
396	50	80	1.020	0	0	…	no	No	good	no	No	Notckd
397	55	80	1.020	0	0	…	no	No	good	no	No	Notckd
398	42	70	1.025	0	0	…	no	No	good	no	No	Notckd
399	12	80	1.020	0	0	…	no	No	good	no	No	Notckd

4.1.2.1 Missing Value Handling

Pada tahap ini, dilakukan penanganan terhadap *missing value* pada *dataset.* Pada penelitian ini, penanganan *missing value* dilakukan dengan mengisi data yang kosong menggunakan metode *most frequent* atau nilai yang sering muncul pada atribut tersebut. Pada *dataset Chronic Kidney Disease* terdapat *missing* value sebesar dari 15%. Dengan adanya *missing value* lebih dari 15%, akan sangat mempengaruhi kinerja dari model

klasifikasi yang telah dibentuk. *Missing value* pada *dataset Chronic Kidney Disease* ditandai dengan tanda tanya (?) seperti ditunjukkan pada Tabel 4.2.

Tabel 4.2 Sampel *Missing Value* pada *Dataset Chronic Kidney Disease*

Id	Age	bp	Sg	Al	su	rbc	pc	Pcc	ba	bgr	Bu	Sc
0	48	80	1.020	1	0	**?**	1	0	0	121	36	1.2
1	7	50	1.020	4	0	**?**	1	0	0	**?**	18	0.8
2	62	80	1.010	2	3	1	1	0	0	423	53	1.8
3	48	70	1.005	4	0	1	0	1	0	117	56	3.8
4	51	80	1.010	2	0	1	1	0	0	106	26	1.4
5	60	90	1.015	3	0	**?**	**?**	0	0	74	25	1.1
6	68	70	1.010	0	0	**?**	1	0	0	100	54	24
7	24	**?**	1.015	2	4	1	0	0	0	410	31	1.1
8	52	100	1.015	3	0	1	0	1	0	138	60	1.9
9	53	90	1.020	2	0	0	0	1	0	70	107	7.2

Pengisian *missing value* dilakukan dengan mengganti nilai yang *missing value* dengan nilai yang memiliki frekuensi muncul paling banyak dalam sebuah atribut. Nilai dengan frekuensi paling banyak muncul untuk setiap atribut ditunjukkan pada Tabel 4.3.

Tabel 4.3 Nilai untuk Mengisi *Missing Value*

No	Atribut	Nilai untuk *Missing Value*
1	*Age*	60
2	*Bp*	80
3	*Sg*	1020
4	*Al*	0
5	*Su*	0
6	*Rbc*	Normal
7	*Pc*	Normal
8	*Pcc*	*Notpresent*
9	*Ba*	*Notpresent*
10	*Bgr*	99
11	*Bu*	46
12	*Sc*	1.2
13	*Sod*	135
14	*Pot*	3.5
15	*Hemo*	15.0
16	*Pcv*	41
17	*Wc*	9800
18	*Rc*	5.2
19	*Htn*	*No*
20	*Dm*	*No*
21	*Cad*	*No*
22	*Appet*	*Good*
23	*Pe*	*No*
24	*Ane*	*No*

4.1.2.2 Transformasi Data

Transfomasi data dilakukan untuk mengubah data dari data yang bertipe nominal ke data bertipe numerik. Hal ini dilakukan agar mempermudah pengolahan *dataset* menggunakan bahasa pemrograman *Python.* Data yang diubah antara lain atribut *red blood cells, pus cell, pus cell clumps, bacteria, hypertension, diabetes mellitus, coronary artery disease, appetite, pedal edema, anemia.* Transformasi data pada atribut *dataset Chronic Kidney Disease* di tunjukan pada Tabel 4.4.

Tabel 4.4 Transformasi Atribut

Atribut	Transformasi
Sg	*1.005=1; 1.01=2; 1.015=3; 1.02=4; 1.025=4*
Rbc	*normal=1; abnormal=0;*
Pc	*normal=1; abnormal=0;*
Pcc	*present=1; notpresent=0;*
Ba	*present=1; notpresent=0;*
Htn	*yes=1; no=0;*
Dm	*yes=1; no=0;*
Cad	*yes=1; no=0;*
Appet	*good=1; poor=0;*
Pe	*yes=1; no=0;*
ane	*yes=1; no=0;*

Kemudian aturan tersebut di terapkan pada *dataset Chronic Kidney Disease*. Hasil dari transformasi data tersebut ditampilkan pada Tabel 4.5 sebagai berikut.

Tabel 4.5 Hasil Transformasi data

Id	Sg	Al	Su	Rbc	Pc	Pcc	Ba	Htn	Dm	Cad	Appet	Pe	Ane	Class
0	3	1	0	1	1	0	0	1	1	0	0	0	0	0
1	3	4	0	1	1	0	0	0	0	0	0	0	0	0
2	1	2	3	1	1	0	0	0	1	0	1	0	1	0
3	0	4	0	1	0	1	0	1	0	0	1	1	1	0
4	1	2	0	1	1	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
395	3	0	0	1	1	0	0	0	0	0	0	0	0	1
396	4	0	0	1	1	0	0	0	0	0	0	0	0	1
397	3	0	0	1	1	0	0	0	0	0	0	0	0	1
398	4	0	0	1	1	0	0	0	0	0	0	0	0	1
399	4	0	0	1	1	0	0	0	0	0	0	0	0	1

4.1.2.3 Tahap Clustering

Dalam tahap *clustering* akan dijelaskan cara mengolah data ke dalam bentuk *cluster* menggunakan algoritma *K-Means*. Tahap ini akan diambil atribut *age* sebagai contoh dalam proses *clustering*. Data yang digunakan berjumlah 10 data untuk sampel perhitungan *clustering* yaitu data ke-1 sampai data ke-10 dapat dilihat pada Tabel 4.6.

Tabel 4.6 Sampel data untuk proses *clustering*

Data-ke	***Age***
1	48
2	7
3	62
4	48
5	51
6	60
7	68
8	24
9	52
10	53

Adapun tahapan dalam proses algoritma *K-Means*:

1) Menentukan jumlah k-*cluster* yang akan dibentuk.

   Dalam penelitian ini jumlah k-*cluster* yang digunakan yaitu k=2.

2) Membangkitkan k-*centroid* (titik pusat *cluster*) secara acak.

   Pusat awal *cluste*r digunakan sebagai sampel perhitungan yaitu:

   1. Data ke-3 = c0 = 62
   2. Data ke-8 = c1 = 24

3) Menghitung jarak setiap data terhadap masing-masing *centroid.* Rumus yang digunakan yaitu rumus *Euclidean* dengan Persamaan (21).

$$D(x_2, x_1) = \sqrt{\sum_{i=1}^{n}(x_2 - x_1)}^2 \quad (21)$$

Dimana:

$D(x_2, x_1)$= dimensi data (jarak data)

x_1 = posisi pusat *cluster*

x_2 = posisi objek data

Id-0 = c0 = $\sqrt{(48 - 62)^2}$ = 14

c1 = $\sqrt{(48-24)^2}$ = 24

Id-1 = c0 = $\sqrt{(7-62)^2}$ = 55

c1 = $\sqrt{(7-24)^2}$ = 17

Ulangi langkah 3 sampai semua anggota atribut ter-*cluster* sehingga menghasilkan data seperti Tabel 4.7.

Tabel 4.7 Hasil perhitungan jarak data terhadap pusat *cluster*

Id	***Age***	**C0**	**C1**	**Jarak Terdekat**
0	48	14	24	14
1	7	55	17	17
2	62	0	38	0
3	48	14	24	14
4	51	11	27	11
5	60	2	36	2
6	68	6	44	6
7	24	38	0	0
8	52	10	28	10
9	53	9	29	9

Keterangan : C0 = *Cluster* 1, C1 = *Cluster* 2

4) Lalu dikelompokkan data berdasarkan jarak terdekat antara data dengan *centroid*. Hasil pengelompokkan dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Pengelompokkan untuk iterasi 1

Data ke-	C0	C1
1	1	
2		1
3	1	
4	1	
5	1	
6	1	
7	1	
8		1
9	1	
10	1	

Keterangan : C0 = *Cluster* 1, C1 = *Cluster* 2

5) Menentukan nilai *centroid* yang baru dengan menghitung rata-rata dari *cluster* yang dihasilkan pada iterasi 1 menggunakan Persamaan (22).

$$C_k = \frac{1}{n_k} \sum d_i \tag{22}$$

Dimana:

n_k= jumlah data dalam *cluster* k

d_i= jumlah nilai jarak yang masuk dalam masing-masing *cluster*

Adapun hasil diperhitungan untuk menentukan pusat *cluster* baru sebagai berikut:

1. *Cluster* 1 = $\frac{(\text{data 1+data 3+data 4+data 5+data 6+data 7+data 9+data 10})}{8}$

   $= \frac{48+62+48+51+60+68+52+53}{8} = 55{,}25$

2. *Cluster* 2 = $\frac{(\text{data 2+data 8})}{2}$

   $= \frac{7+24}{2} = 12{,}50$

6) Lakukan perulangan dari langkah 3–5 hingga anggota tiap *cluster* tidak ada berubah.
7) Hasil dari data yang ter-*cluster* ini akan digunakan pada proses selanjutnya yaitu pada proses seleksi fitur menggunakan algoritma PSO.

4.1.2.4 Tahap Seleksi Fitur

Tahapan PSO digunakan untuk mencari fitur terbaik yang ada pada *Chronic Kidney Disease Dataset.* Fitur terbaik merupakan fitur terpilih yang digunakan untuk proses selanjutnya.

Adapun tahapan algoritma PSO untuk proses seleksi fitur (atribut) adalah sebagai berikut.

1) Input data untuk seleksi fitur seperti Tabel 4.9. Dalam tabel tesebut digunakan 5 atribut sebagai sampel dari keseluruhan 25 atribut yang ada.

Tabel 4.9 Data sampel yang akan diseleksi fitur

Age	***Specific Grafity***	***Albumin***	***Red Blood Cell Count***	***Appetite***
0	3	1	0	1
2	3	4	0	1
1	1	2	0	0
0	0	4	1	0
0	1	2	0	1
1	2	3	0	1
1	1	0	0	1
2	2	2	0	1
0	2	3	1	1
0	3	2	1	0
0	1	2	0	1
1	1	3	1	0
1	2	3	1	0
1	3	0	0	0
1	1	3	1	0
0	2	3	1	1
0	2	2	0	1
0	3	0	0	0
1	4	0	0	1
1	2	1	1	1
1	2	2	1	0
1	3	0	1	1
0	4	4	1	1
2	1	0	0	0
0	2	4	0	0
1	4	0	1	1
1	2	0	1	0
1	1	3	1	1
1	3	1	0	1
1	0	1	0	1
1	3	0	0	1
1	2	3	1	0
1	1	1	1	0
1	3	2	0	0
1	1	1	0	0
1	3	2	0	0
1	2	1	0	1
1	3	0	1	0

2) Jumlah partikel yang ditetapkan sebanyak 30 partikel dan iterasi sebanyak 100.
3) Misal sampel sebanyak 3 partikel. Inisialisasi posisi setiap partikel (*xt*), bobot inersia (*w*) = 0.9, koefisien akselerasi 1 (c_1) = 0.5, dan koefisien akselerasi 2 (c_2) = 0.5. Pada tahap awal kecepatan partikel (*vt*) = 0. Posisi tiap partikel di isnisialisasi seperti berikut.

   Partikel 1 = [01111]

   Partikel 2 = [10110]

   Partikel 3 = [11011]
4) Hitung *fitness* dari partikel menggunakan algoritma SVM.

   Nilai *fitness* tiap partikel:

   Partikel 1 = [01111] = 0,925

   Partikel 2 = [10110] = 0,850

   Partikel 3 = [11011] = 0,800
5) Tuliskan nilai *pbest* masing-masing partikel berdasarkan nilai akurasi yang dihasilkan oleh SVM. Tentukan *gbest* berdasarkan nilai *pbest* tertinggi.

   pbest Partikel 1 = [01111] = 0,925

   pbest Partikel 2 = [10110] = 0,850

   pbest Partikel 3 = [11011] = 0,800

   Berdasarkan nilai *pbest* di atas maka nilai *gbest* = 0,925
6) Hitung kecepatan dan posisi partikel menggunakan Persamaan (23)

$$v_{id}^{new} = w \,.\, v_{id}^{old} + \; c_1 . r_1 \left(pb_{id}^{old} - x_{id}^{old} \right) + c_2 . r_2 \left(gb_d^{old} - x_{id}^{old} \right) \quad (23)$$

Dimana r_1 dan r_2 merupakan nilai random antara 0-1. Misal r_1= 0.6 dan r_2= 0.6.

Kecepatan partikel 1 = 0,9×0 + 0,5×0,6 (0,925-0,925) + 0,5×0,6 (0,925-0,925)

= 0

Kecepatan partikel 2 = 0,9×0 + 0,5×0,6 (0,850-0,850) + 0,5×0,6 (0,925-0,850)

= 0,0225

Kecepatan partikel 3 = 0,9×0 + 0,5×0,6 (0,800-0,800) + 0,5×0,6 (0,925-0,800)

= 0,0375

7) Perbarui posisi dengan Persamaan (24).

Perbarui posisi dengan rumus sigmoid (S) dari kecepatan yang telah diperbarui di atas dengan Persamaan (24). Dalam penerapan algortima PSO digunakan untuk solusi permasalahan seleksi fitur, maka dapat menggunakan *binary digit* untuk menunjukkan fitur. Fitur yang tidak terpilih dilambangkan dengan 0, sedangkan fitur terpilih dilambangkan dengan 1.

$$S = \frac{1}{1+ e^{-v_{id}^{new}}} \tag{24}$$

$$x_{id}^{new} = \begin{cases} 1, if\ Sigmoid(S) > rand\ (0,1) \\ 0, Otherwise \end{cases}$$

Dalam penelitian ini menggunakan inisialisasi populasi partikel secara acak yaitu *rand* (0:1) merupakan rentang nilai dari 0-1.

Partikel 1 = $S = \frac{1}{1+ e^{-0}}$ = 0,5

1. Misal *rand* (0:1) = 0,124

   if rand (0:1) < S : 1 *else* 0

   0,124 < 0,5 : 1 *else* 0

   True: 1

2. Misal *rand* (0:1) = 0,635

if rand (0:1) < S : 1 *else* 0

0,635 < 0,5 : 1 *else* 0

False: 0

3. Misal *rand* (0:1) = 0,697

if rand (0:1) < S : 1 *else* 0

0,697 < 0,5 : 1 *else* 0

False: 0

4. Misal *rand* (0:1) = 0,389

if rand (0:1) < S : 1 *else* 0

0,389 < 0,5 : 1 *else* 0

True: 1

5. Misal *rand* (0:1) = 0,358

if rand (0:1) < S : 1 *else* 0

0,358 < 0,5 : 1 *else* 0

True: 1

Posisi partikel 1 yang telah diperbaharui = [10011]

Sebelumnya posisi partikel 1 adalah = [01111]

Partikel 2 = $S = \frac{1}{1+e^{-0,0225}}$ = 0,444

1. Misal *rand* (0:1) = 0,525

if rand (0:1) < S : 1 *else* 0

0,525 < 0,444 : 1 *else* 0

False: 0

2. Misal *rand* (0:1) = 0,455

if rand (0:1) < S : 1 *else* 0

0,635 < 0,444 : 1 *else* 0

False: 0

3. Misal *rand* (0:1) = 0,214

if rand (0:1) < S : 1 *else* 0

0,214 < 0,444: 1 *else* 0

True: 1

4. Misal *rand* (0:1) = 0,327

if rand (0:1) < S : 1 *else* 0

0,327 < 0,444 : 1 *else* 0

True: 1

5. Misal *rand* (0:1) = 0,351

if rand (0:1) < S : 1 *else* 0

0,351 < 0,444 : 1 *else* 0

True: 1

Posisi partikel 2 yang telah diperbaharui = [00111]

Sebelumnya posisi partikel 2 adalah = [10110]

Partikel 3 = $S = \frac{1}{1+e^{-0,0375}}$ = 0,490

1. Misal *rand* (0:1) = 0,244

if rand (0:1) < S : 1 *else* 0

0,244 < 0,490 : 1 *else* 0

True: 1

2. Misal *rand* (0:1) = 0,981

if rand (0:1) < S : 1 *else* 0

0,981 < 0,490 : 1 *else* 0

False: 0

3. Misal *rand* (0:1) = 0,487

if rand (0:1) < S : 1 *else* 0

0,487 < 0,490 : 1 *else* 0

True: 1

4. Misal *rand* (0:1) = 0,495

if rand (0:1) < S : 1 *else* 0

0,495 < 0,490 : 1 *else* 0

False: 0

5. Misal *rand* (0:1) = 0,764

if rand (0:1) < S : 1 *else* 0

0,764 < 0,490 : 1 *else* 0

False: 0

Posisi partikel 3 yang telah diperbaharui = [10100]

Sebelumnya posisi partikel 3 adalah = [11011]

8) Perulangan berhenti ketika mencapai iterasi maksimal. Atribut hasil seleksi didapat dengan mengambil nilai *fitness* tertinggi dari semua iterasi.

9) Data yang dihasilkan dari proses seleksi fitur menggunakan PSO ini akan digunakan untuk proses selanjutnya yaitu proses klasifikasi menggunakan algoritma SVM.

4.1.2.5 Tahap Pembagian Data

Tahap pembagian data merupakan proses membagi *dataset* menjadi 2 bagian, yaitu data *training* dan data *testing* dengan proporsi 70:30. Dalam pembagian *dataset* ini, digunakan metode *splitter* yang terdapat pada *library sklearn*. Pembagian ini bersifat acak dengan konsistensi pengacakan tertentu.

4.1.2.6 Tahap Evaluasi dengan Confusion Matrix

Evaluasi dengan *confusion matrix* dilakukan untuk mengetahui tingkat akurasi dari perhitungan yang sudah dilakukan dalam penelitian. Adapun langkahnya adalah sebagai berikut.

1. Masukkan hasil pengujian pada tabel *confusion matrix* seperti pada Tabel 4.10.

Tabel 4.10 Evaluasi dengan *confusion matrix*

Klasifikasi		Kelas hasil prediksi		
		Ya	Tidak	Jumlah
Kelas aktual	Ya	70	0	71
	Tidak	1	49	49

2. Hitung nilai akurasi menggunakan Persamaan (20).

$$Accuracy = \frac{70+49}{71+49} \text{ x } 100\% \tag{20}$$

$$= 99{,}167\%$$

4.1.3 Tahap *Mining* Data

Pada tahap ini akan dilakukan tiga kali proses *mining* data. Pertama proses klasifikasi menggunakan algoritma SVM tanpa menggunakan *K-Means* dan PSO. Kedua yaitu proses klasifikasi menggunakan algoritma SVM dengan menerapkan

K-Means. Ketiga, proses klasifikasi menggunakan algoritma SVM dengan PSO. Keempat, proses klasifikasi menggunakan algoritma SVM menggunakan *K-Means clustering* dan PSO. Kemudian seluruh proses klasifikasi tersebut di masukkan dalam *confusion matrix* untuk mengetahui tingkat akurasi, kemudian diukur waktu pemrosesannya pada saat melakukan proses masing-masing klasifikasi.

4.1.3.1 Hasil Algoritma SVM

Penerapan klasifikasi yang pertama adalah dengan menerapkan algoritma SVM dengan perbandingan *data training*:*data testing* adalah 70:30 setelah dilakukan *handling missing value* pada *Chronic Kidney Disease Dataset.* Hasil akurasi yang didapatkan dari penerapan algoritma SVM dapat dilihat pada Tabel 4.11.

Tabel 4.11 Akurasi Algoritma SVM

Algoritma	Hasil Akurasi	Waktu Pemrosesan
SVM	55.0%	0,03 Detik

Penerapan algoritma SVM mendapatkan hasil akurasi sebesar 55.0% dengan waktu pemrosesan selama 0,03 detik. Hasil akurasi ini masih dapat ditingkatkan lagi menggunakan SVM yang di tambahkan dengan *K-Means clustering* dan PSO sehingga dapat menghasilkan akurasi yang lebih baik.

4.1.3.2 Hasil K-Means pada SVM

Algoritma *K-Means* digunakan sebagai proses *clustering* pada atribut yang memiliki data kontinu. Dalam *Chronic Kidney Disease Dataset* terdapat 11 atribut yang memiliki nilai kontinu yang kemudian dilakukan proses *clustering*. Dalam menentukan jumlah k-*cluster*, peneliti melakukan uji coba beberapa jumlah k-

cluster yaitu k = 2, 3, 4, dan 5. Lalu setiap percobaan akan dinilai kualitas jumlah k-*cluster*, dengan menggunakan klasifikasi SVM dan dipilih yang memiliki akurasi tertinggi. Dari hasil percobaan yang telah dilakukan, jumlah k-*cluster* = 2 memiliki akurasi paling tinggi yaitu sebesar **95,83%** dengan waktu pemrosesan rata-rata rata selama **0,63** detik. K-*cluster* = 2 dan k-*cluster* = 3 memiliki akurasi yang sama namun k-*cluster* = 2 memiliki waktu pemrosesan 0,23 detik lebih cepat dibandingkan dengan k-*cluster* = 3, hal itu disebabkan karena k-*cluster* = 2 memiliki persebaran data yang lebih sedikit dibanding k-*cluster* = 3. Hal tersebut menyebabkan waktu pemrosesan pada k-*cluster* = *2* menjadi lebih cepat. Hasil akurasi dan waktu pemrosesan untuk setiap k-*cluster* dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil akurasi dan waktu pemrosesan setiap k-*cluster*

Algoritma	Jumlah k-*cluster*	Akurasi	Waktu Pemrosesan
SVM+*K-Means*	**2**	**95,83%**	**0,63 Detik**
	3	95,83%	0,86 Detik
	4	92,50%	0,97 Detik
	5	90,83%	1,12 Detik

4.1.3.3 Hasil PSO pada SVM

Hasil yang didapatkan oleh PSO dalam melakukan seleksi fitur kemudian di gunakan untuk diklasifikasi menggunakan SVM dengan menggunakan pembagian data yang sama seperti yang diterapkan pada proses penerapan SVM sebelumnya. Hasil akurasi, waktu pemrosesan dan atribut yang terpilih yang dihasilkan pada proses ini seperti pada Tabel 4.13. Pada proses ini menunjukan lebih sedikit atribut yang terpilih maka akurasinya cenderung lebih tinggi.

Tabel 4.13 Hasil akurasi, waktu pemrosesan dan jumlah atribut terpilih pada algoritma SVM dan PSO

Algoritma	Eksekusi ke	Akurasi	Waktu Pemrosesan	Atribut Terpilih
SVM + PSO	1	99,17%	34,99 detik	15
	2	99,17%	32,48 detik	14
	3	99,17%	33,07 detik	13
	4	99,17%	33,28 detik	14
	5	99,17%	35,64 detik	15
	6	98,33%	34,66 detik	16
	7	99,17%	33,43 detik	15
	8	98,33%	34,19 detik	15
	9	99,17%	32,62 detik	15
	10	98,33%	34,38 detik	16
	Rata-rata	**98,92%**	**33,87 detik**	**14,8**

4.1.3.4 Hasil K-Means dan PSO pada SVM

Data yang telah dilakukan *handling missing value* selanjutnya akan di proses menggunakan algoritma *K-Means clustering.* Proses ini hanya di lakukkan kepada *dataset* yang memiliki nilai kontinu. Setelah itu dilakukan proses PSO untuk memilih fitur terbaik. Fitur yang terpilih tersebut kemudian digunakan untuk proses klasifikasi menggunakan algoritma SVM. Data yang dilakukan untuk proses klasifikasi tersebut merupakan *data training*, yaitu 70% dari keseluruhan data. Setelah di klasifikasi kemudian akan di lakukan penghitungan akurasi menggunakan 30% data yang tidak digunakan untuk *data training* menggunakan metode *confusion matrix* dan juga dilihat lama waktu pemrosesannya. Hasil akurasi dan waktu pemrosesan dari algoritma SVM dengan *K-Means* dan PSO dapat dilihat pada Tabel 4.14.

Tabel 4.14 Hasil akurasi dan waktu pemrosesan algoritma SVM dengan *K-Means* dan PSO

Algoritma	Eksekusi ke	Akurasi	Waktu Pemrosesan	Atribut Terpilih
SVM + *K-Means*+ PSO	1	99,17%	13,15 detik	20
	2	97,17%	12,68 detik	21
	3	99,17%	12,70 detik	21
	4	100%	12,84 detik	19
	5	97,50%	12,42 detik	21
	6	99,17%	12,68 detik	20
	7	100%	12,54 detik	19
	8	99,17%	13,15 detik	20
	9	98,33%	12,83 detik	21
	10	98,33%	13,17 detik	22
	Rata-rata	**98,80%**	**12,82 deik**	**20,4**

Hasil akurasi yang dihasilkan algoritma SVM tanpa menggunakan *K-Means* dan PSO adalah 55,0%. Kemudian akurasi tertinggi yang dihasilkan algoritma SVM dengan menggunakan PSO adalah 99,17% dengan 14 atribut terpilih dan waktu pemrosesan tercepat adalah 32,48 detik. Sementara hasil akurasi tertinggi pada algoritma SVM dengan menggunakan *K-Means* dan PSO adalah 100% dengan 19 atribut terpilih dan waktu pemrosesan tercepat adalah 12,54 detik. Sehingga terjadi peningkatan akurasi sekitar 43% antara sebelum menggunakan PSO dan sesudah menggunakan PSO dan terjadi percepatan waktu pemrosesan sekitar 20 detik dari hanya menggunakan SVM dan PSO dengan setelah menggunakan SVM dengan menggunakan *K-Means* dan PSO dalam melakukan klasifikasi penyakit ginjal kronis menggunakan *Chronic Kidney Disease Dataset.*

4.2 Implementasi Sistem

4.2.1 Tahap Pembuatan Sistem

Sistem digunakan sebagai alat uji sekaligus efisensi perhitungan dari metode. Sistem yang dibuat berbasis web dengan menggunakan bahasa *Python* dengan *Flask Framework*. Untuk bagian *user interface* menggunakan bahasa HTML (*Hyper Text Markup Language*) dan CSS (*Cascading Style Sheet*).

4.2.2 Implementasi Algoritma

Implementasi algoritma dalam penelitian ini menggunakan bahasa pemrograman *Python* dengan *Flask Framework*, *pyswarms documentation* dan *scikit-learn library*.

4.2.2.1 Implementasi Algoritma SVM

Implementasi dilakukan pada *Chronic Kidney Disease Dataset* dengan algoritma SVM. Dengan pembagian data sebesar 70:30 dengan menggunakan metode *splitting data* seperti pada Gambar 4.2.

```
#import modul yang digunakan
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

#baca dataset
df=pd.read_csv
('F:\SKRIPSWEET\Dataset\missing\chronic_kidney_disease_most.csv')

#membuat array dan memisahkan atribut dan kelas
df.drop(['id'],1,inplace=True)
clf=SVC(gamma='auto')
X=np.array(df.drop(['class'],1))
y=np.array(df['class'])

#membagi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state=5)
clf.fit(X_train, y_train)
clf_predictions = clf.predict(X_test)
```

```
print("Accuracy: {}%".format(clf.score(X_test, y_test) * 100))
```

Gambar 4.2 *Source code* Algoritma SVM

4.2.2.2 Implementasi Algoritma K-Means

Implementasi algoritma *K-Means* pada SVM digunakan untuk mengubah data menjadi bentuk *cluster*. Dalam penelitian ini *clustering* dilakukan pada atribut yang bernilai kontinu saja seperti pada Gambar 4.3.

```
#import modul yang digunakan
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.cluster import KMeans

#membaca dataset
df=pd.read_csv
('F:\SKRIPSWEET\Dataset\kmeans\chronic_kidney_disease.csv')

#membuat array dan memisahkan atribut dan kelas
df.drop(['id'],1,inplace=True)
clf=SVC(gamma='auto')
X=np.array(df.drop(['class'],1))
y=np.array(df['class'])

#proses klasterisasi
print(X.shape)
x_baru=[]
x_baru_cluster=[]

cluster_name=['a','b','c','d','e','f','g','h','i','j','k']

for i in range(X.shape[1]):
    x1=X[:,i]
    x1=np.array(x1).reshape(-1,1)
    kmeans = KMeans(n_clusters=2, random_state=5).fit(x1)
    x_baru.append(kmeans.labels_)
x_baru=np.array(x_baru).transpose()

for i, v in enumerate(cluster_name):
    x_baru_cluster=[]
    for x in x_baru[:,i]:
        if x == 0:
            x_baru_cluster.append(v+'1')
        elif x== 1:
            x_baru_cluster.append(v+'2')
        else:
            x_baru_cluster.append(v+'3')
    x_baru_cluster.append(x_baru_cluster)
```

```
#memperoleh data terklaster
x_baru_cluster=np.array(x_baru_cluster).transpose()
df=pd.DataFrame(x_baru)
#menyimpan dataset terklaster
df.to_csv('F:\SKRIPSWEET\Dataset\kmeans\chronic_kidney_disease_cluster
2.csv')
```

Gambar 4.3 *Source code* Algoritma *K-Means*

4.2.2.3 Implementasi PSO pada Algoritma SVM

Implementasi dilakukan dengan menerapkan PSO untuk melakukan seleksi fitur sehingga menghasilkan fitur terpilih akurasi dan waktu pemrosesan seperti pada Gambar 4.4.

```
#import modul yang digunakan
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pyswarms as ps
import time as time

#mulai menghitung waktu pemrosesan
start=time.time()

#membaca dataset
df=pd.read_csv
('F:\SKRIPSWEET\Dataset\preprocessing\chronic_kidney_disease_preproces
sed_full.csv')

#membuat array dan memisahkan atribut dan kelas
df.drop(['id'],1,inplace=True)
clf=SVC(gamma='auto')
X=np.array(df.drop(['classification'],1))
y=np.array(df['classification'])
print(X.shape)
def f_per_particle(m, alpha):
    total_features = nfitur

    # proses seleksi fitur PSO
    if np.count_nonzero(m) == 0:
        X_subset = X
    else:
        X_subset = X[:,m==1]
    xtrain,   xtes,   ytrain,   ytes   =   train_test_split(X_subset,y,
test_size=0.3, random_state=5, shuffle=True)
    clf.fit(xtrain, ytrain)
    P = (clf.predict(xtes) == ytes).mean()
    j = (alpha * (1.0 - P)
```

```
        + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))
    return j
def f(x, alpha=0.88):
    n_particles = x.shape[0]
    j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
    return np.array(j)
options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': 30, 'p':2}
nsampel, nfitur = X.shape
dimensions = nfitur
optimizer           =           ps.discrete.BinaryPSO(n_particles=30,
dimensions=dimensions, options=options)
cost, pos = optimizer.optimize(f, iters=100)

clf=SVC(gamma='auto')
X_selected_features = X[:,pos==1]
xtrain, xtes, ytrain, ytes = train_test_split(X_selected_features, y,
test_size=0.3, random_state=5, shuffle=True)
clf.fit(xtrain, ytrain)
subset_performance = (clf.predict(xtes) == ytes).mean()
end = time.time()
waktu= end-star

#akurasi yang diperoleh
print('Akurasi: %.3f' % (subset_performance))

#waktu yang dihabiskan
print('waktu:' (waktu))
```

Gambar 4.4 *Source code* PSO pada Algoritma SVM

4.2.2.4 Implementasi Algoritma K-Means dan PSO pada Algortima SVM

Implementasi dilakukan dengan menerapkan *K-Means* untuk melakukan *clustering* pada dataset dan PSO untuk melakukan seleksi fitur kemudian dilakukan proses *splitting data* dengan perbandingan 70:30. Kemudian atribut yang diperoleh di klasifikasi menggunakan algoritma SVM yang kemudian menghasilkan model. Model tersebut kemudian diuji menggunakan *data testing* sehingga menghasilkan akurasi. Proses ini juga dihitung untuk mengetahui berapa lama waktu pemrosesannya. Implementasi pada tahapan ini dapat dilihat seperti pada Gambar 4.5.

```
#import modul yang digunakan
import pandas as pd
import numpy as np
```

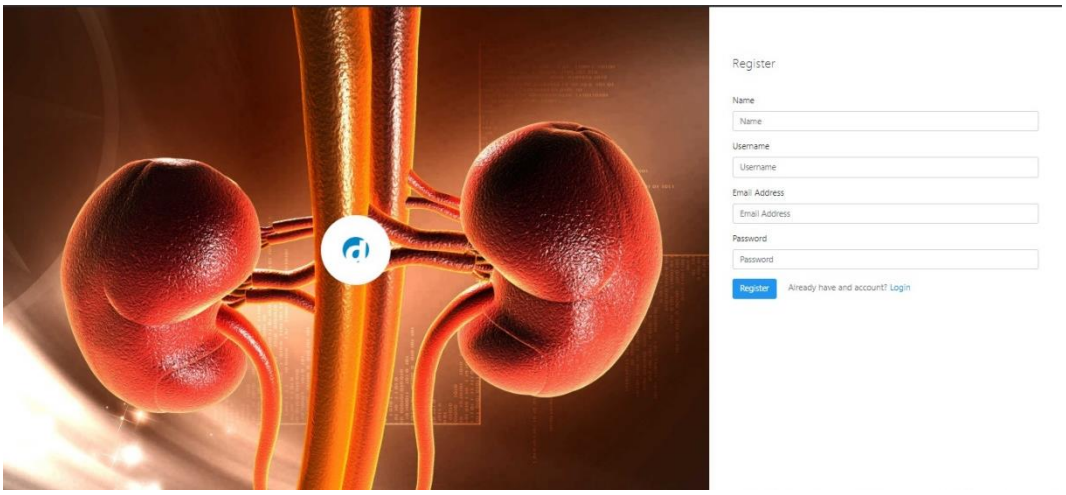
```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.cluster import KMeans
import pyswarms as ps
import time as time

#mulai menghitung waktu pemrosesan
start=time.time()

#baca dataset
df=pd.read_csv
('F:\SKRIPSWEET\Dataset\missing\chronic_kidney_disease_most.csv')

#membuat array dan memisahkan atribut dan kelas
df.drop(['id'],1,inplace=True)
clf=SVC(gamma='auto')
X=np.array(df.drop(['class'],1))
y=np.array(df['class'])

#proses klastering
print(X.shape)
x_baru=[]
x_baru_cluster=[]

cluster_name=['a','b','c','d','e','f','g','h','i','j','k','l','m','n',
'o','p','q','r','s']

for i in range(X.shape[1]):
    x1=X[:,i]
    x1=np.array(x1).reshape(-1,1)
    kmeans = KMeans(n_clusters=2, random_state=5).fit(x1)
    x_baru.append(kmeans.labels_)
x_baru=np.array(x_baru).transpose()

for i, v in enumerate(cluster_name):
    x_baru_cluster=[]
    for x in x_baru[:,i]:
        if x == 0:
            x_baru_cluster.append(v+'0')
        elif x== 1:
            x_baru_cluster.append(v+'1')
        else:
            x_baru_cluster.append(v+'2')
    x_baru_cluster.append(x_baru_cluster)

x_baru_cluster=np.array(x_baru_cluster).transpose()
print(x_baru)

#proses PSO
def f_per_particle(m, alpha):
    total_features = nfitur
```

```
    if np.count_nonzero(m) == 0:
        X_subset = x_baru
    else:
        X_subset = x_baru[:,m==1]

    xtrain,   xtes,   ytrain,   ytes   =   train_test_split(X_subset,y,
test_size=0.3, random_state=5, shuffle=True)
    clf.fit(xtrain, ytrain)
    P = (clf.predict(xtes) == ytes).mean()
    j = (alpha * (1.0 - P)
        + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))
    return j
def f(x, alpha=0.88):
    n_particles = x.shape[0]
    j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
    return np.array(j)

options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': 30, 'p':2}

nsampel, nfitur = X.shape
dimensions = nfitur
optimizer           =           ps.discrete.BinaryPSO(n_particles=30,
dimensions=dimensions, options=options)

cost, pos = optimizer.optimize(f, iters=100)
clf=SVC(gamma='auto')

X_selected_features = x_baru[:,pos==1]
xtrain, xtes, ytrain, ytes = train_test_split(X_selected_features, y,
test_size=0.3, random_state=5, shuffle=True)

clf.fit(xtrain, ytrain)
subset_performance = (clf.predict(xtes) == ytes).mean()
end = time.time()
waktu= end-star

#akurasi yang diperoleh
print('Akurasi: %.3f' % (subset_performance))

#waktu yang dihabiskan
print('waktu:' (waktu))
```

Gambar 4.5 *Source code* Implementasi *K-Means* dan PSO pada Algoritma SVM

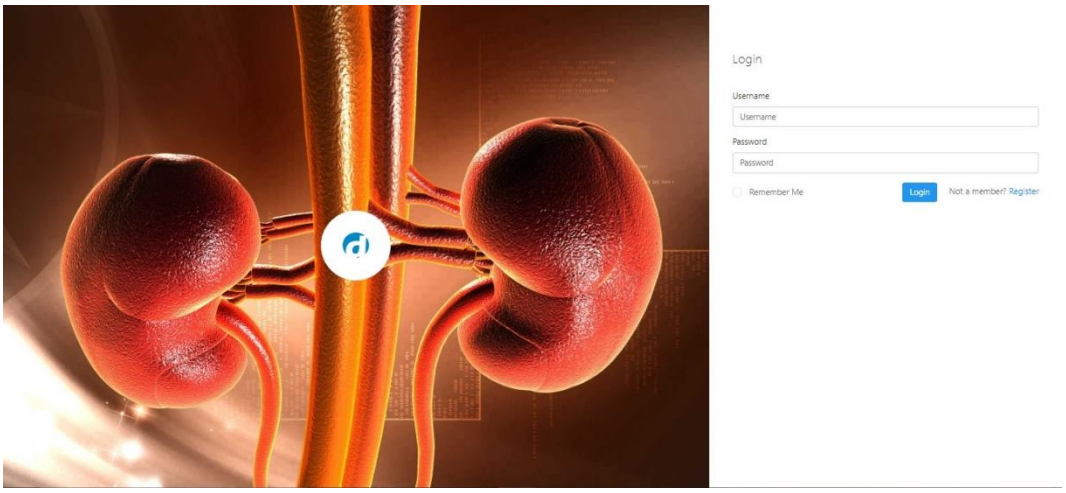
4.2.3 Implementasi *User Interface*

User Interface pada sistem ini berbasis *web* dengan menggunakan bahasa pemrograman HTML, *JavaScript* dan CSS. Sebelum menggunakan aplikasi ini, pengguna perlu melakukan registrasi terlebih dulu seperti pada Gambar 4.6.



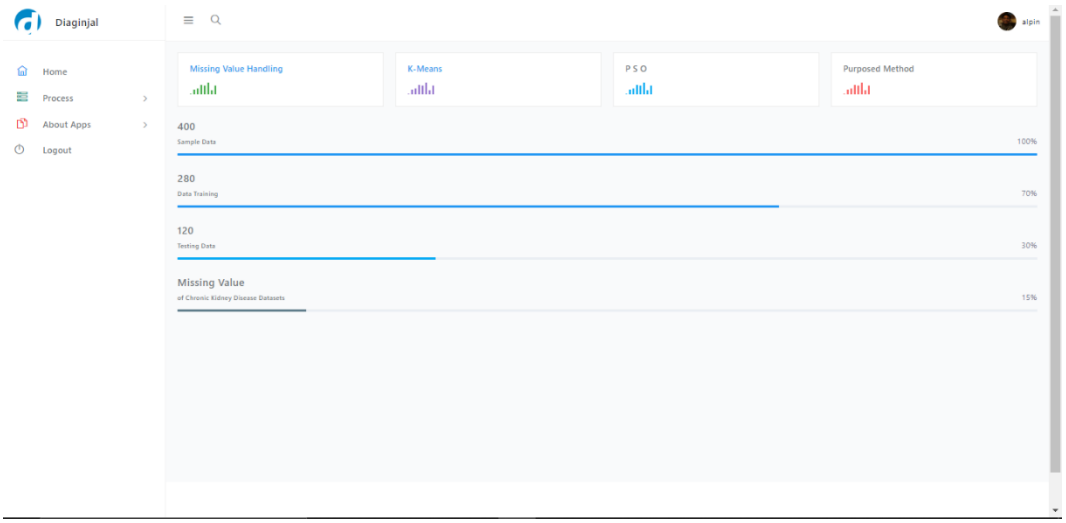
Gambar 4.6 Tampilan *Register User* baru

Setelah melakukan registrasi pengguna bisa langsung masuk kedalam sistem dengan melakukan *login* pada aplikasi seperti pada Gambar 4.7.



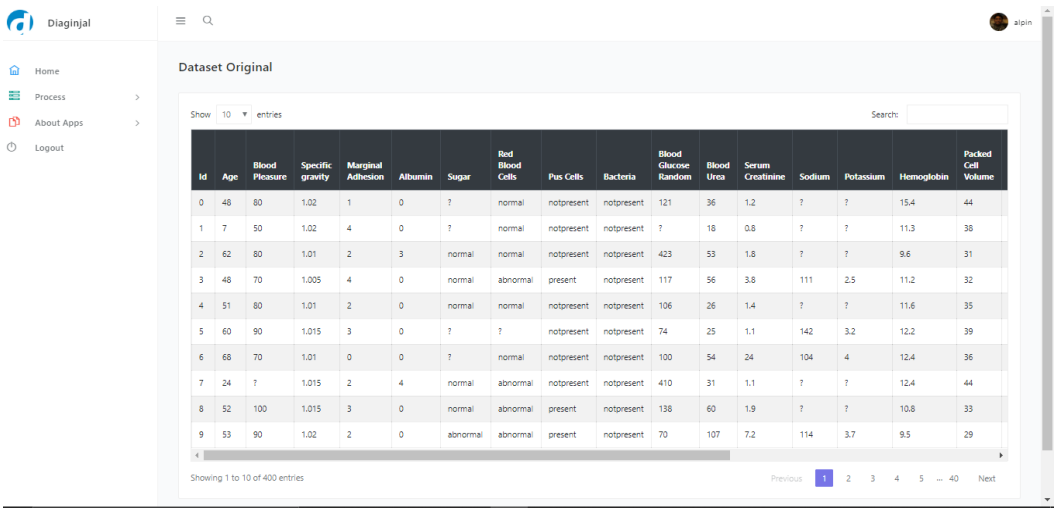
Gambar 4.7 Tampilan *Login User*

Pada sistem ini terdapat 4 *menu* utama yaitu *Home, Process, About Apps* dan *Logout*. Pada *menu Home* terdapat *shortcut* menuju *menu Handled Datasets, K-Means*, PSO dan *Proposed Method.* Di *menu Home* juga di tampilkan jumlah data yang ada dalam *dataset,* jumlah *missing value* pada *dataset* dan juga ditampilkan jumlah *training dan testing data* yang digunakan dalam penelitian seperti Gambar 4.8.



Gambar 4.8 Tampilan *menu Home*

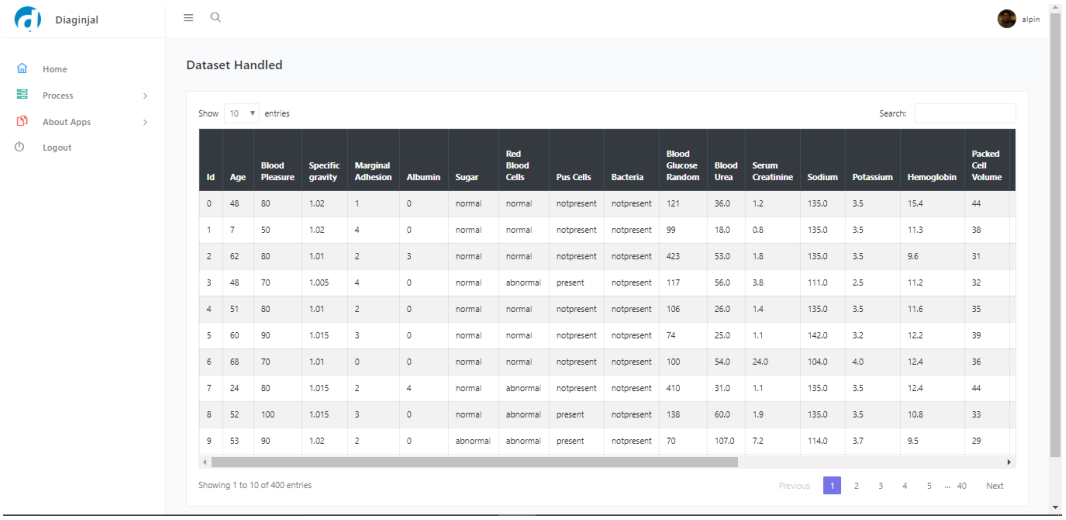
Pada *menu Process* terdapat beberapa *sub menu* yaitu *menu Datasets*, *K-Means*, PSO dan *Proposed Method*. Pada *sub menu Datasets* juga terdapat 2 *sub menu* yaitu *Original Datasets* dan *Handled Datasets*. *Original Datasets* merupakan *dataset* asli yang diperoleh dari UCI *Machine Learning Repository* seperti pada Gambar 4.9.



Id	Age	Blood Pleasure	Specific gravity	Marginal Adhesion	Albumin	Sugar	Red Blood Cells	Pus Cells	Bacteria	Blood Glucose Random	Blood Urea	Serum Creatinine	Sodium	Potassium	Hemoglobin	Packed Cell Volume
0	48	80	1.02	1	0	?	normal	notpresent	notpresent	121	36	1.2	?	?	15.4	44
1	7	50	1.02	4	0	?	normal	notpresent	notpresent	?	18	0.8	?	?	11.3	38
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	53	1.8	?	?	9.6	31
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	56	3.8	111	2.5	11.2	32
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	26	1.4	?	?	11.6	35
5	60	90	1.015	3	0	?	?	notpresent	notpresent	74	25	1.1	142	3.2	12.2	39
6	68	70	1.01	0	0	?	normal	notpresent	notpresent	100	54	24	104	4	12.4	36
7	24	?	1.015	2	4	normal	abnormal	notpresent	notpresent	410	31	1.1	?	?	12.4	44
8	52	100	1.015	3	0	normal	abnormal	present	notpresent	138	60	1.9	?	?	10.8	33
9	53	90	1.02	2	0	abnormal	abnormal	present	notpresent	70	107	7.2	114	3.7	9.5	29

Gambar 4.9 Tampilan *Original Datasets*

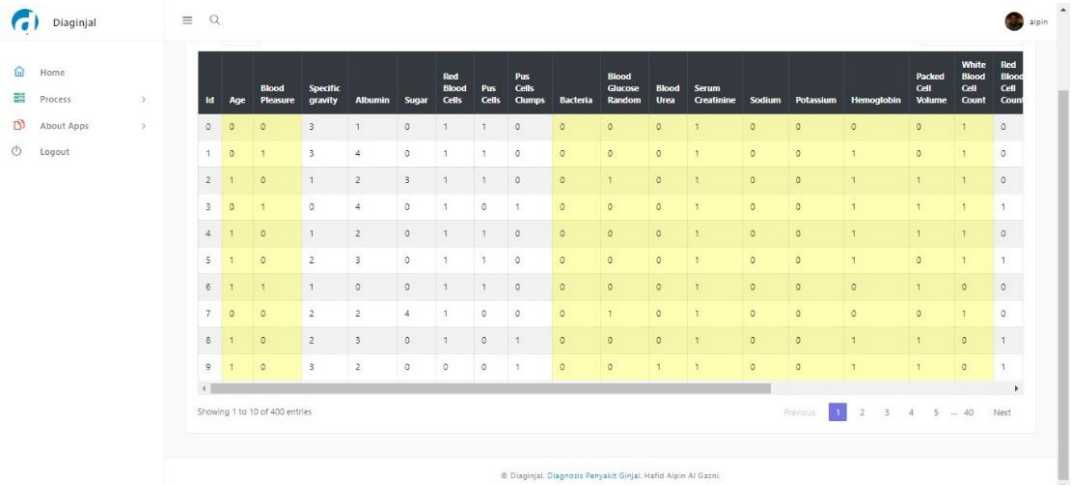
Pada *sub menu Datasets* juga terdapat *menu Handled Datasets* yaitu dataset yang telah dilakukan *handling missing value* menggunakan modus *(most frequent)*. Dapat dilihat pada Gambar 4.10.



Id	Age	Blood Pleasure	Specific gravity	Marginal Adhesion	Albumin	Sugar	Red Blood Cells	Pus Cells	Bacteria	Blood Glucose Random	Blood Urea	Serum Creatinine	Sodium	Potassium	Hemoglobin	Packed Cell Volume
0	48	80	1.02	1	0	normal	normal	notpresent	notpresent	121	36.0	1.2	135.0	3.5	15.4	44
1	7	50	1.02	4	0	normal	normal	notpresent	notpresent	99	18.0	0.8	135.0	3.5	11.3	38
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	53.0	1.8	135.0	3.5	9.6	31
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	56.0	3.8	111.0	2.5	11.2	32
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	26.0	1.4	135.0	3.5	11.6	35
5	60	90	1.015	3	0	normal	normal	notpresent	notpresent	74	25.0	1.1	142.0	3.2	12.2	39
6	68	70	1.01	0	0	normal	normal	notpresent	notpresent	100	54.0	24.0	104.0	4.0	12.4	36
7	24	80	1.015	2	4	normal	abnormal	notpresent	notpresent	410	31.0	1.1	135.0	3.5	12.4	44
8	52	100	1.015	3	0	normal	abnormal	present	notpresent	138	60.0	1.9	135.0	3.5	10.8	33
9	53	90	1.02	2	0	abnormal	abnormal	present	notpresent	70	107.0	7.2	114.0	3.7	9.5	29

Gambar 4.10 Tampilan *Handled Datasets*

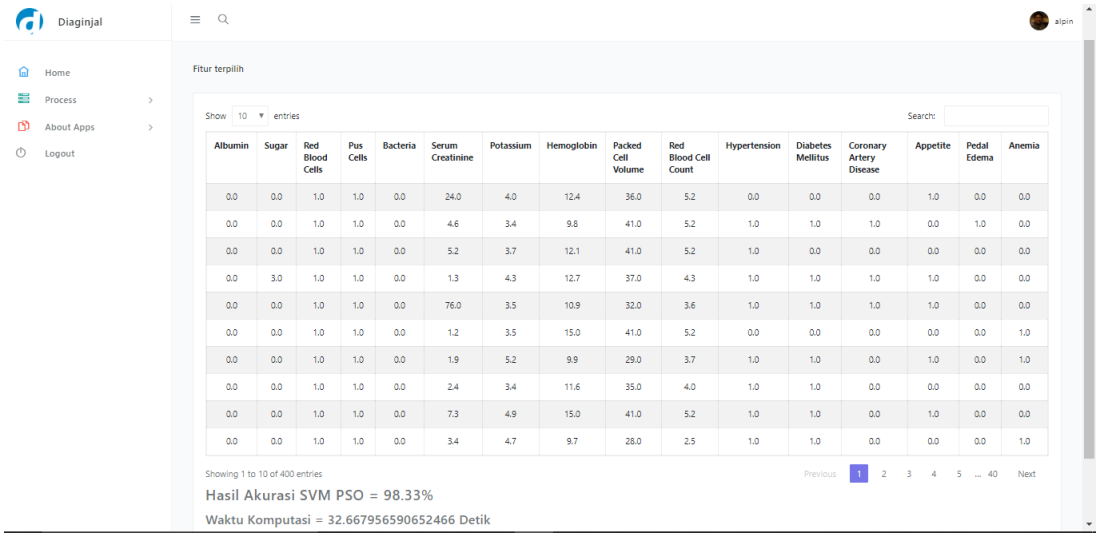
Sub menu selanjutnya pada *menu Process* adalah *menu K-Means.* Dalam *sub menu K-Means* ini ditampilkan *dataset* yang telah dilakukan proses *clustering* pada atribut yang bertipe nominal. Dapat dilihat pada Gambar 4.11.



Id	Age	Blood Pleasure	Specific gravity	Albumin	Sugar	Red Blood Cells	Pus Cells	Pus Cells Clumps	Bacteria	Blood Glucose Random	Blood Urea	Serum Creatinine	Sodium	Potassium	Hemoglobin	Packed Cell Volume	White Blood Cell Count	Red Blood Cell Coun
0	0	0	3	1	0	1	1	0	0	0	0	1	0	0	0	0	1	0
1	0	1	3	4	0	1	1	0	0	0	0	1	0	0	1	0	1	0
2	1	0	1	2	3	1	1	0	0	1	0	1	0	0	1	1	1	0
3	0	1	0	4	0	1	0	1	0	0	0	1	0	0	1	1	1	1
4	1	0	1	2	0	1	1	0	0	0	0	1	0	0	1	1	1	0
5	1	0	2	3	0	1	1	0	0	0	0	1	0	0	1	0	1	1
6	1	1	1	0	0	1	1	0	0	0	0	1	0	0	0	1	0	0
7	0	0	2	2	4	1	0	0	0	1	0	1	0	0	0	0	1	0
8	1	0	2	3	0	1	0	1	0	0	0	1	0	0	1	1	0	1
9	1	0	3	2	0	0	0	1	0	0	1	1	0	0	1	1	0	1

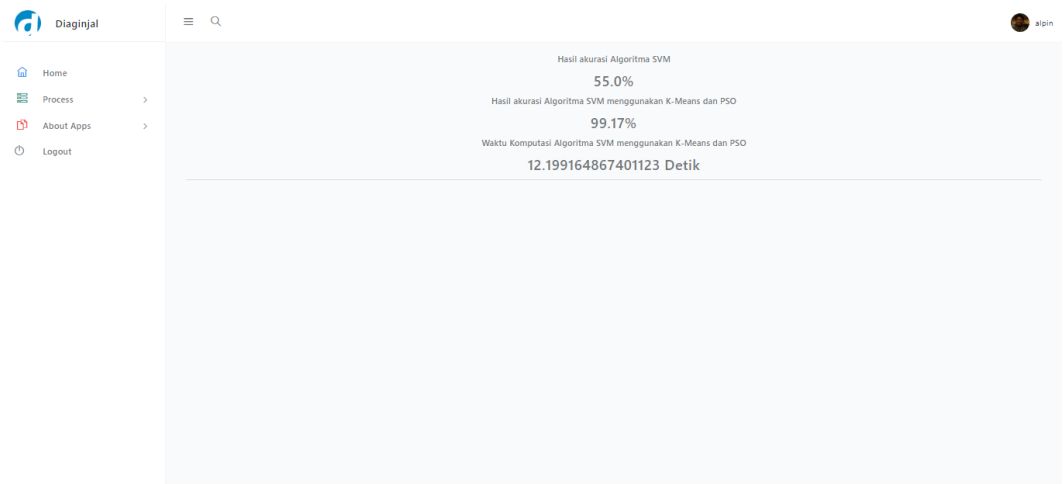
Gambar 4.11 Tampilan *sub menu K-Means*

Selain *sub menu K-Means* dalam *menu Process* juga terdapa *sub menu* PSO dimana dalam *sub menu* PSO menampilkan fitur yang terpilih dalam *dataset* dan juga menampilkan hasil akurasi juga waktu yang dihasilkan dalam memperoses algoritma PSO dengan *classifier* SVM seperti ditunjukan pada Gambar 4.12.



Gambar 4.12 Tampilan *sub menu* PSO

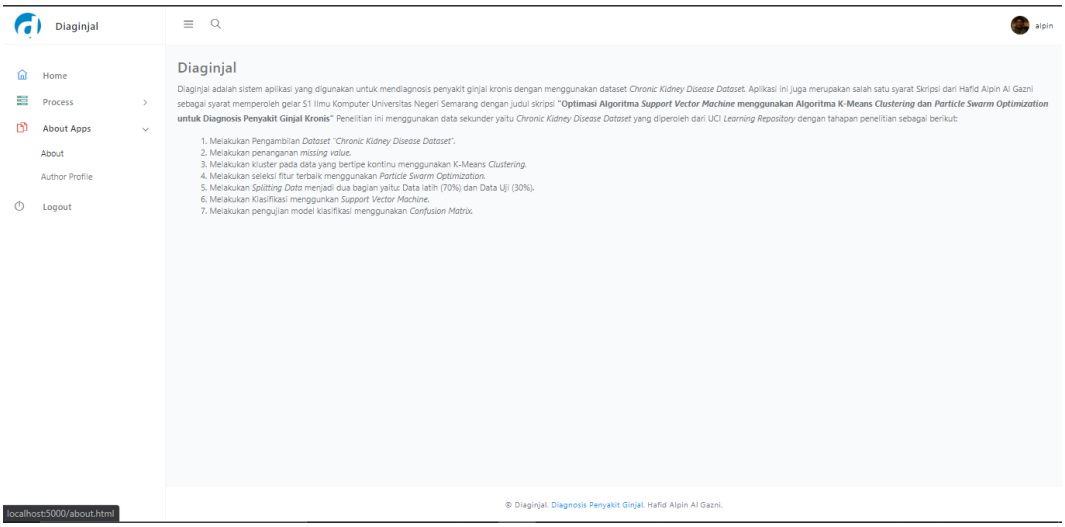
Sub menu terakhir pada *menu Process* adalah menu *Proposed Method* dimana pada *menu Proposed Method* ini menampilkan akurasi yang dihasilkan oleh algoritma SVM dan juga menampilkan akurasi serta waktu pemrosesan yang



Gambar 4.13 Tampilan *sub menu Purposed Method*

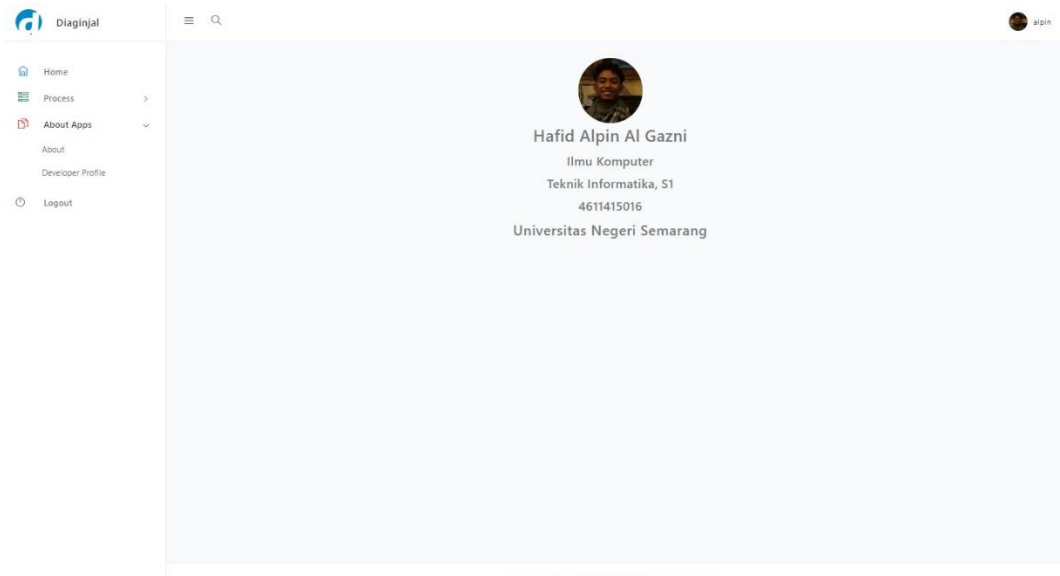
dihasilkan setelah menggunakan metode yang diusulkan yaitu algoritma SVM dengan *K-Means* dan PSO seperti ditunjukan pada Gambar 4.13.

Menu selanjutya adalah *menu About Apps* dimana terdapat 2 *sub menu* yaitu *About* dan *Developer Profile*. Dalam *menu About* berisi deskripsi mengenai sistem dan langkah-langkah penelitian yang dilakukan seperti pada Gambar 4.14.



Gambar 4.14 Tampilan *menu About*

Pada *menu About Apps* juga terdapat *sub menu Developer Profile* yang berisi tentang identitas peneliti seperti ditunjukan pada Gambar 4.15.



Gambar 4.15 Tampilan *sub menu Developer Profile*

Menu yang terakhir adalah *menu Logout* yang berfungsi untuk keluar dari aplikasi *Diaginjal.*

4.3 Pembahasan

Penelitian ini menerapkan algoritma *K-Means* untuk proses *clustering* pada setiap atribut kontinu pada *dataset* dan *Particle Swarm Optimization* untuk proses seleksi fitur, sebagai optimasi algoritma *Support Vector Machine* agar menghasilkan peningkatan akurasi pada proses klasifikasi. *Dataset* yang digunakan pada penelitian ini yaitu *Chronic Kidney Disease Dataset* dari UCI *Machine Learning Repository. Dataset* ini memiliki 25 Atribut dengan 400 sampel data. *Dataset* ini juga memiliki *missing value* sebesar 15% sehingga perlu dilakukan *handling missing value* untuk mengatasi hal tersebut.

Penelitian ini bertujuan untuk mengetahui cara kerja, hasil akurasi dan efisieansi waktu yang diperoleh dari optimasi algoritma SVM menggunakan algoritma *K-Means* dan seleksi fitur PSO pada diagnosis penyakit ginjal kronis.

Pada penelitian ini dilakukan perbandingan antara klasifikasi SVM tanpa menggunakan algoritma *K-Means* dan PSO dengan klasifikasi SVM menggunakan algoritma *K-Means* dan PSO untuk mengetahui pengaruh algoritma *K-Means* dan PSO dalam meningkatkan hasil akurasi dan waktu pemrosesan pada algoritma SVM.

Tahap *clustering* menggunakan algoritma *K-Means* dilakukan pada atribut yang memiliki nilai kontinu pada *dataset.* Pemilihan jumlah k-*cluster* berpengaruh pada hasil akurasi dan kecepatan waktu pemrosesan yang diperoleh, Dari percobaan yang telah dilakukan, jumlah k-*cluster* = 2 dan 3 dapat memberikan akurasi yang

lebih baik dibandingkan dengan jumlah k-*cluster* = 4 atau 5. Jumlah k-*cluster* = 2 dan 3 menghasilkan akurasi sebesar 95,83% namun k-*cluster* = 2 lebih unggul dalam kecepatan waktu pemrosesan, lebih cepat 0,23 detik. Dari percobaan yang telah dilakukan algoritma *K-Means* dapat mempercepat waktu pemrosesan sekitar 20 detik dibandingkan dengan klasifikasi SVM tanpa menggunakan algoritma *K-Means*.

Tahap seleksi fitur menggunakan PSO dilakukan untuk memilih fitur terbaik pada *dataset* yang nantinya akan digunakan untuk proses klasifikasi. Misal dari data yang memiliki 25 atribut nanti akan terpilih 19 atribut yang digunakan untuk proses klasifikasi. Hal ini dapat mengoptimalkan kinerja algoritma SVM dalam melakukan klasifikasi pada *dataset.* Setelah didapatkan sebuah model dari metode yang digunakan, lalu tahap pengujian akurasi menggunakan *confusion matrix.*

Penelitian ini mencatat tiap akurasi dan waktu pemrosesan yang dihasilkan dari proses *mining* data yang telah dilakukan. Adapun hasil tersebut dapat dilihat pada Tabel 4.15.

Tabel 4.15 Hasil tiap metode yang digunakan

Algoritma	Akurasi	Waktu Pemrosesan
SVM	55,00%	0,03 detik
SVM + *K-Means*	95,83%	0,63 detik
SVM +PSO	98,92%	33,87 detik
Proposed Method	**98,80%**	**12,82 detik**

Berdasarkan Tabel 4.13 maka diketahui adanya peningkatan akurasi dari metode SVM tanpa menggunakan *K-Means* hal itu disebabkan karena proses *K-*

Means ini mengubah atribut yang bernilai kontinu menjadi beberapa *cluster* sehingga meningkatkan akurasi pada saat proses klasifikasi menggunakan algoritma SVM namun mengalami perlambatan waktu pemrosesan dikarenakan proses *K-Means* itu sendiri. Algoritma PSO juga memberikan peingkatan akurasi yanng cukup signifikan, bahkan lebih tinggi dibandingkan dengan algoritma SVM menggunakan *K-Means*. Hal itu disebabkan karena PSO hanya menggunakan atribut yang terpilih saja dan membuang atribut yang memiliki redundan. Namun algoritma PSO ini membutuhkan waktu pemrosesan yang cukup lama. Pada metode yang diusulkan penulis menggabungkan PSO dan *K-Means* untuk optimasi algoritma SVM dengan tujuan agar tak hanya mendapatkan akurasi yang tinggi namun juga menghasilkan waktu pemrosesan yang cepat. PSO digunakan untuk menyeleksi fitur yang dibutuhkan untuk meningkatkan akurasi sedangkan *K-Means* digunakan untuk mengatasi persebaran data pada *dataset* sehingga menyebabkan percepatan pada waktu pemrosesan.

Dengan demikian, optimasi yang dilakukan pada algoritma SVM dengan menggunakan *K-Means* dan PSO (*proposed method*) dapat memberikan kontribusi pengetahuan tentang hasil akurasi yang akurat dan waktu pemrosesan yang cepat dalam diagnosis penyakit ginjal kronis dan dapat digunakan dalam proses optimasi klasifikasi pada algoritma SVM. Namun dalam penelitian ini juga terdapat kekurangan, yaitu jumlah fitur yang diperoleh pada proses seleksi fitur masih berubah-ubah sehingga perlu pengembangan lebih lanjut agar jumlah fitur yang terpilih dapat seminimal mungkin dengan mempertimbangkan fitur-fitur penting yang berpengaruh dalam diagnosis penyakit ginjal kronis.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian, kesimpulan yang diperoleh berdasarkan rumusan masalah yaitu:

1. Cara kerja algoritma *K-Means* yaitu dengan mengelompokkan setiap atribut kontinu pada *Chronic Kidney Disease Dataset* dan pemilihan jumlah k-*cluster* dapat mempengaruhi hasil akurasi dan waktu pemrosesan yang diperoleh. Dalam 4 percobaan yaitu dengan menggunakan 2, 3, 4 dan 5 *cluster*, jumlah k-*cluster* = 2 memberikan hasil akurasi dan waktu pemrosesan yang lebih baik dibanding jumlah *cluster* yang lain. *K-Means* berpengaruh dalam peningkatan waktu pemrosesan pada saat klasifikasi karena *K-Means* mengatasi masalah persebaran data pada dataset. Setelah dilakukan proses *clustering* dengan algoritma *K-Means,* selanjutnya dilakukan proses seleksi fitur menggunakan *Particle Swarm Optimization.* Dalam proses ini menghasilkan 19-22 fitur yang terpilih dari 24 fitur yang ada. Semakin sedikit atribut yang terpilih maka menghasilkan akurasi yang semakin tinggi. Hasil dari proses seleksi fitur kemudian dilakukan proses klasifikasi menggunakan algoritma SVM yang menghasilkan sebuah model. Model yang didapat kemudian dilakukan pengujian untuk mengetahui akurasi yang diperoleh. pada penelitian ini disamping

menghitung akurasi juga di hitung waktu pemrosesan pada saat melakukan pengujian klasifikasi.

2. Pada penelitian ini dilakukan percobaan dengan beberapa metode yaitu: dengan hanya menggunakan SVM, SVM dengan menggunakan PSO dan SVM dengan mengunakan *K-Means* dan PSO. Hasil klasifikasi algoritma SVM pada dataset *Chronic Kidney Disease* menghasilkan akurasi sebesar 55,0%. Lalu percobaan dilakukan dengan menggunakan algoritma SVM dan PSO menghasilkan rata-rata akurasi yang cukup tinggi yaitu 98,92% dengan rata-rata 14,8 atribut yang terpilih dan menghasilkan rata-rata lama waktu pemrosesan 33,87 detik. Sedangkan metode SVM dengan menambahkan proses *clustering* sebelum proses PSO dengan menggunakan algoritma *K-Means* pada atribut kontinu dalam *dataset* dapat menghasilkan rata-rata akurasi sebesar 98,80% dan rata-rata lama waktu pemrosesan 12,82 detik dengan rata rata 20,4 atribut yang terpilih. Pada metode SVM dengan *K-Means* dan PSO ini menghasilkan akurasi yang lebih rendah karena pada metode ini atribut yang terpilih lebih banyak dibanding yang hanya menggunakan PSO saja. Algoritma SVM dengan menggunakan *K-Means* dan PSO terbukti dapat meningkatkan akurasi sekitar 43,80% dari akurasi yang dihasilkan dari algoritma SVM dan meningkatkan waktu pemrosesan sekitar 20 detik dari algoritma SVM yang hanya mengunakan PSO dengan algoritma SVM yang menggunakan *K-Means* dan PSO. Maka dapat disimpulkan bahwa metode yang digunakan yaitu optimasi algoritma *Support Vector Machine* menggunakan algoritma *K-Means* dan seleksi fitur

Particle Swarm Optimization dapat mengoptimalkan dan meningkatkan akurasi serta waktu pemrosesan dalam proses klasifikasi algoritma SVM pada diagnosis penyakit ginjal kronis.

5.2 Saran

Saran yang diberikan oleh peneliti, yaitu:

1) Diharapkan dapat mengembangkan lebih lanjut mengenai optimasi algoritma *Support Vector Machine* agar memberikan akurasi dan efisiensi waktu lebih baik dalam proses klasifikasi.
2) Diharapkan untuk penelitian selanjutnya dapat melakukan eksplorasi seleksi fitur menggunakan metode PSO sehingga dapat mengembangkan metode PSO menjadi lebih baik.
3) Diharapkan untuk penelitian selanjutnya dapat melakukan eksplorasi pada algoritma *K-Means* sehingga menjadi lebih baik dalam meningkatkan efisiensi waktu pada saat klasifikasi.

DAFTAR PUSTAKA

Abedalkhader, W., & Abdulrahman, N. (2017). Missing Data Classification of Chronic Kidney Disease. *International Journal of Data Mining & Knowledge Management Process, 7*(5), 55-61.

Agarwal, S. (2013). Data mining: Data Mining Concepts and Techniques. *International Conference on Machine Intelligence and Research Advancement*, 203-207.

Anggodo, Y. F., Cahyaningrum, W., Fauzziyah, A. N., Khoiriyah, I. L., Kartikasari, O., & Choilissodin, I. (2017). Hybrid *K-Means* dan Particle Swarm Optimization Untuk Clustering Nasabah Kredit. *Jurnal Tekologi Informasi dan Ilmu Komputer (JTIIK), 4*(2), 104-110.

Ardjani, F., Sadouni, K., & Benyettou, M. (2010). Optimization of SVM Multi Class by Particle Swarm (PSO-SVM). *International Workshop on Database Technology and Applications.*

Ashari, I. A., Muslim, M. A., & Alamsyah. (2016). Comparison Performance of Genetic Algorithm and Ant Colony Optimization in Course Scheduling Optimizing. *Scientific Journal of Informatics, 3*(2), 149-158.

Chamidah, N. (2018). *K-Means* Sebagai Ekstraktor Ciri Pada Klasifikasi Data Dengan Algoritma SVM. Jurnal SIMETRIS, *3*(9), 1-3.

Charleonnan, A., Fufaung, T., Niyomwong, T., Chokchueypattanakit, W., Suwannawach, S., & Ninchawee, N. (2016). Predictive Analytics for Chronic Kidney Disease Using Machine Learning Techniques. *The 2016 Management and Innovation Technology International Conference (MITiCON-2016)*, 80-83.

Deshpande, B. (2015). *Predictive Analytics and Data Mining Concepts and Practice with RapidMiner.* Massachusetts: Morgan Kaufmann.

Fadilla, I., Adikara, P. P., & Perdana, R. S. (2018). Klasifikasi Penyakit Chronic Kidney Disease (CKD) dengan Menggunakan Metode Extreme Learning Machine (ELM). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2*(10), 3397-3405.

Gupta, S., Kumar, D., & Sharma, A. (2011). Data Mining Classification Techniques Applied for Breast Cancer Diagnosis and Prognosis. *Indian Journal of Computer Science and Engineering (IJCSE), 2*(2), 188-195.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques (Third Edition)*. United State of America: Elsevier.

Hermawati, F. A., 2013, Data Mining, Penerbit ANDI, Yogyakarta.

Inbarani, H. H., Azar, A. T., & Jothi, G. 2014. Supervised Hybrid Feature Selection Based on PSO and Rough Sets for Medical Diagnosis. *Computer Methods and Program in Biomedicine*, 113: 175-185.

Kerlinger. (2006). *Asas–Asas Penelitian Behaviour*. Edisi 3, Cetakan 7. Yogyakarta: Gadjah Mada University Press.

Kumari, S., & Singh, A. (2013). A Data Mining Approach for the Diagnosis of Diabetes Mellitus. *7th International Conference on Intelligent Systems and Control (ISCO)*, 373-375.

Kunwar, V., Chandel, K., Sabitha, A. S., & Bansal, A. (2016). Chronic Kidney Disease Analysis Using Data Mining Classification Techniques. *6th International Conference-Cloud System and Big Data Engineering (Confluence)*, 300-305.

Lee, S., Soak, S., Oh, S., Pedrycz, W., & Jeon, M. (2008). Modified Binary Particle Swarm Optimization. *Progress in Natural Science*, *18*(9), 1161-1166.

Mirqotussa'adah, Muslim, M. A., Sugiharti, E., Prasetiyo, B., & Alimah, S. (2017). Penerapan Dizcretization dan Teknik Bagging Untuk Meningkatkan Akurasi Klasifikasi Berbasis Ensemble pada Algoritma C4.5 dalam Mendiagnosa Diabetes. *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, *8*(2), 135-143.

Molina, H. G., Ullman, J. D., & Widom, J. (2009). *Database Systems The Complete Book* (2nd ed.). New Jersey: Pearson Prentice Hall.

Nayak, J., Naik, B., & Behera, H. S. (2015). A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges. *International Journal of Database Theory and Application*, *8*(1), 169-186.

North, M. A. (2012). *Data Mining for the Masses.* Global Text Project.

Purwar, A. & Singh, S. K. (2015). Hybrid Prediction Model with Missing Value Imputation for Medical Data. *Expert Systems With Applications*, *42*(13), 5621-5631.

Priliani, E. M., Putra, A. T., & Muslim, M. A. 2018. Forecasting Inflation Rate Using Support Vector Regression (SVR) Based Weight Attribute Particle

Swarm Optimization (WAPSO). *Scientific Journal of Informatics*, *5*(2), 118-127.

Rahman, A. T., Wiranto, & Anggrainingsih, R. 2017. Coal Trade Data Clusterung Using *K-Means* (Case Study PT. Global Bangkit Utama). *ITSMART: Jurnal Ilmiah Teknologi dan Informasi*, *6*(1), 24-31.

Rajeshinigo, D. & Jebamalar, J. P. A. (2017). Accuracy Improvement of C4.5 using *K-Means* Clustering. *International Journal of Science and Research (IJSR), 6*(6), 2755-2758.

Retno, T. V. 2017. *Data Mining Teori dan Aplikasi Rapidminer*. Yogyakarta: Penerbit Gaya Media.

Rubini, L. J. & Eswaran, D. (2015). Generating Comparative Analysis of Early Stage Prediction of Chronic Kidney Disease. *International Journal of Modern Engineering Research (IJMER), 5*(7), 49-55.

Salekin, A., & Stankovic, J. (2016). Detection of Chronic Kidney Disease and Selecting Important Predictive Attributes. *IEEE International Conference on Healthcare Informatics (ICHI)*, 262-270.

Somantri, O., Wiyono, S., & Dairoh (2016). Optimalisasi Support Vektor Machine (SVM) untuk Klasifikasi Tema Tugas Akhir Berbasis *K-Means*. TELEMATIKA, *13*(02), 59 – 68.

Soni, J., Ansari, U., Sharma, D., & Soni, S. (2011). Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction. *International Journal of Computer Applications*, *17*(8), 43-48.

Sousa, T., Silva, A., & Neves, A. (2004). Particle Swarm Based Data Mining Algorithms for Classification Tasks. *Parallel computing*, *30*(5-6), 767-783.

Sreedhar, C., Kasiviswanath, N., & Reddy, P. C. (2017). Clustering Large Datasets Using *K-Means* Modifed Inter and Intra Clustering (KM-I2C) in Hadoop. *Journal of Big Data*, *4*(27), 1-19.

Srivastava, D. K. & Bhambhu, L. (2010). Data Classification Using Support Vector Machine. *Journal of Theoretical and Applied Information Technology, 12*(1), 1-7.

Sumathi, S. & Paneerselvam, S. (2010). *Computational Intelligence Paradigms : Theory & Applications Using MATLAB.* Florida: CRC Press.

Suyanto. (2017). *Data Mining untuk Klasifikasi dan Klasterisasi Data.* Bandung: Penerbit Informatika.

Tan, P., Steinbach, M., Karpatne, A., & Kumar, V. 2013. *Introduction to Data Mining (Second Edition)*. London: Prentice Hall.

Tran, B., Xue, B., & Zhang, M. (2014). Overview of Particle Swarm Optimisation for Feature Selection in Classification. In *Asia-Pacific Conference on Simulated Evolution and Learning* 605-617.

Wang, K. J., Makond, B., Chen, K. H., & Wang, K. M. (2014). A Hybrid Classifier Combining SMOTE with PSO to Estimate 5-Year Survivability of Breast Cancer Patients. *Applied Soft Computing, 20*(1), 15-24.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining Practical Machine Learning Tools and Techniques* (3rd ed.). Burlington: Morgan Kaufmann.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ... & Zhou, Z. H. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, *14*(1), 1-37.

Xue, B., Zhang, M., & Browne, W. N. (2013). Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach. *IEEE Transactions on Cybernetics*, *43*(6), 1656-1671

Yao, Y., Liu, Y., Yu, Y., Xu, H., Lv, W., Li, Z & Chen, X. (2013). K-SVM: An Effective SVM Algorithm Based on *K-Means* Clustering. *Journal of* Computers, *8*(10).

LAMPIRAN

Lampiran 1. *Source Code views.py*

```
# Python modules
import os, logging

# Flask modules
from flask                import render_template, request, url_for, redirect,
send_from_directory
from  flask_login              import  login_user,  logout_user,  current_user,
login_required
from werkzeug.exceptions import HTTPException, NotFound, abort

# App modules
from app        import app, lm, db, bc
from app.models import User
from app.forms  import LoginForm, RegisterForm


# provide login manager with load_user callback
@lm.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
#dataset ori
@app.route('/datasets-original.html')
def datasets_original():
    import pandas as pd
    import numpy as np
    df=pd.read_csv
('F:\SKRIPSWEET\Dataset\original\chronic_kidney_disease.csv')

    X=np.array(df.drop(['class'],1))
    y=np.array(df['class'])
    n1=len(df)
    return render_template('pages/datasets-original.html',df=df,X=X,y=y,n1=n1)

    #dataset handled
@app.route('/datasets-handled.html')
def datasets_handled():
    import pandas as pd
    import numpy as np
    df1=pd.read_csv
('F:\SKRIPSWEET\Dataset\handled\chronic_kidney_disease_most.csv')

    X1=np.array(df1.drop(['class'],1))
    y1=np.array(df1['class'])
    n2=len(df1)
    return                                    render_template('pages/datasets-
handled.html',df1=df1,X1=X1,y1=y1,n2=n2)

    #dataset kmeans
@app.route('/kmeans.html')
def dataset_kmeans():
    import pandas as pd
    import numpy as np

    df1=pd.read_csv
('F:\SKRIPSWEET\Dataset\preprocessing\chronic_kidney_disease_preprocessed_ful
l2.csv')
```

```
    X1=np.array(df1.drop(['class'],1))
    y1=np.array(df1['class'])
    n5=len(df1)

    return render_template('pages/kmeans.html',df1=df1,X1=X1,y1=y1,n5=n5)
#pso
@app.route('/pso.html')
def pso():
    import pandas as pd
    import numpy as np
    from sklearn.svm import SVC
    from sklearn.model_selection import train_test_split
    from sklearn import metrics
    from sklearn.cluster import KMeans
    import pyswarms as ps
    import time as time

    start=time.time()
    df=pd.read_csv
('F:\SKRIPSWEET\Dataset\missing\chronic_kidney_disease_most.csv')

    df.drop(['id'],1,inplace=True)
    #print(df)
    clf=SVC(gamma='auto')
    X=np.array(df.drop(['class'],1))
    y=np.array(df['class'])

    print(X.shape)
    def f_per_particle(m, alpha):
        total_features = nfitur
        # Get the subset of the features from the binary mask
        if np.count_nonzero(m) == 0:
            X_subset = X
        else:
            X_subset = X[:,m==1]
        # Perform classification and store performance in P
        xtrain,    xtes,    ytrain,    ytes    =    train_test_split(X_subset,y,
test_size=0.3, random_state=5, shuffle=True)
        clf.fit(xtrain, ytrain)
        P = (clf.predict(xtes) == ytes).mean()
        # Compute for the objective function
        j = (alpha * (1.0 - P)
            + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))

        return j
    def f(x, alpha=0.88):
        n_particles = x.shape[0]
        j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
        return np.array(j)
    # Initialize swarm, arbitrary
    options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': 30, 'p':2}

    # Call instance of PSO
    nsampel, nfitur = X.shape
    dimensions = nfitur # dimensions should be the number of features
    optimizer  =  ps.discrete.BinaryPSO(n_particles=30,  dimensions=dimensions,
options=options)

    # Perform optimization
    cost, pos = optimizer.optimize(f, iters=100)
```

```
    X_selected_features = X[:,pos==1]  # subset
    attr = ["Age","Blood Pleasure","Specific gravity","Albumin","Sugar","Red
Blood Cells","Pus Cells",
    "Pus Cells Clumps","Bacteria","Blood Glucose Random","Blood Urea","Serum
Creatinine","Sodium","Potassium","Hemoglobin","Packed Cell Volume","White
Blood Cell Count",
    "Red Blood Cell Count","Hypertension","Diabetes Mellitus","Coronary Artery
Disease","Appetite","Pedal Edema","Anemia"]
    attrArr = np.array(attr)
    tempAttrSelected = attrArr[pos == 1]
    attrSelected = pd.DataFrame(tempAttrSelected)
    nAttr = attrSelected.shape[0]
    tempDataPSO = X[:, pos == 1]
    dataPSO = pd.DataFrame(tempDataPSO) #pembuatan data frame dari fitur
terpilih
    n4 = len(dataPSO) #jumlah instance fitur terpilih
    xtrain, xtes, ytrain, ytes = train_test_split(X_selected_features, y,
test_size=0.3, random_state=5, shuffle=True)

    clf.fit(xtrain, ytrain)

    # Compute performance
    subset_performance = (clf.predict(xtes) == ytes).mean()
    pso = np.around(subset_performance*100, decimals=2)
    end = time.time()
    waktu= end-start
    return    render_template(    'pages/pso.html',dataPSO=dataPSO,    n4=n4,
nAttr=nAttr, attrSelected=attrSelected, pso=pso, waktu=waktu)

# purposed
@app.route('/purposed.html')
def purposed():
    import pandas as  pd
    import numpy as np
    from sklearn.svm import SVC
    from sklearn.model_selection import train_test_split
    from sklearn import metrics
    from sklearn.metrics import confusion_matrix
    from sklearn.cluster import KMeans
    import pyswarms as ps
    import time as time

    df2=pd.read_csv
('F:\SKRIPSWEET\Dataset\missing\chronic_kidney_disease_most.csv')
    start=time.time()
    df=pd.read_csv
('F:\SKRIPSWEET\Dataset\preprocessing\chronic_kidney_disease_preprocessed_ful
l.csv')

    df.drop(['id'],1,inplace=True)
    #print(df)
    clf=SVC(gamma='auto')
    X=np.array(df.drop(['class'],1))
    y=np.array(df['class'])

    df2.drop(['id'],1,inplace=True)
    #print(df)
    clf=SVC(gamma='auto')
    A=np.array(df2.drop(['class'],1))
    B=np.array(df2['class'])
```

```
def f_per_particle(m, alpha):
        total_features = nfitur
        # Get the subset of the features from the binary mask
        if np.count_nonzero(m) == 0:
            X_subset = X
        else:
            X_subset = X[:,m==1]
        # Perform classification and store performance in P
        xtrain,   xtes,   ytrain,   ytes   =   train_test_split(X_subset,y,
test_size=0.3, random_state=5, shuffle=True)
        clf.fit(xtrain, ytrain)
        P = (clf.predict(xtes) == ytes).mean()
        # Compute for the objective function
        j = (alpha * (1.0 - P)
            + (1.0 - alpha) * (1 - (X_subset.shape[1] / total_features)))

        return j
    def f(x, alpha=0.88):
        n_particles = x.shape[0]
        j = [f_per_particle(x[i], alpha) for i in range(n_particles)]
        return np.array(j)
    # Initialize swarm, arbitrary
    options = {'c1': 0.5, 'c2': 0.5, 'w':0.9, 'k': 30, 'p':2}

    # Call instance of PSO
    nsampel, nfitur = X.shape
    dimensions = nfitur # dimensions should be the number of features
    optimizer  =  ps.discrete.BinaryPSO(n_particles=30,  dimensions=dimensions,
options=options)

    # Perform optimization
    cost, pos = optimizer.optimize(f, iters=100)
    # Create two instances of LogisticRegression
    clf=SVC(gamma='auto')

    # Get the selected features from the final positions
    X_selected_features = X[:,pos==1]  # subset
    xtrain,  xtes,  ytrain,  ytes  =  train_test_split(X_selected_features,  y,
test_size=0.3, random_state=5, shuffle=True)
    # Perform classification and store performance in P
    clf.fit(xtrain, ytrain)

    # Compute performance
    subset_performance = (clf.predict(xtes) == ytes).mean()
    A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2,
random_state=5)
    clf.fit(A_train, B_train)
    clf_predictions = clf.predict(A_test)
    svm=clf.score(A_test, B_test)
    hasilsvm= np.around(svm* 100, decimals=2)
    hasil= np.around(subset_performance*100, decimals=2)
    end = time.time()
    waktu= end-start
    return      render_template(      'pages/purposed.html',      hasil=hasil,
hasilsvm=hasilsvm, waktu=waktu)

# Logout user
@app.route('/logout.html')
def logout():
    logout_user()
```

```
    return redirect(url_for('index'))

# Register a new user
@app.route('/register.html', methods=['GET', 'POST'])
def register():

    # declare the Registration Form
    form = RegisterForm(request.form)

    msg = None

    if request.method == 'GET':

        return render_template( 'pages/register.html', form=form, msg=msg )

    # check if both http method is POST and form is valid on submit
    if form.validate_on_submit():

        # assign form data to variables
        username = request.form.get('username', '', type=str)
        password = request.form.get('password', '', type=str)
        email    = request.form.get('email'   , '', type=str)

        # filter User out of database through username
        user = User.query.filter_by(user=username).first()

        # filter User out of database through username
        user_by_email = User.query.filter_by(email=email).first()

        if user or user_by_email:
            msg = 'Error: User exists!'

        else:

            pw_hash = password #bc.generate_password_hash(password)

            user = User(username, email, pw_hash)

            user.save()

            msg  =  'User  created,  please  <a  href="'  +  url_for('login')  +
'">login</a>'

    else:
        msg = 'Input error'

    return render_template( 'pages/register.html', form=form, msg=msg )

# Authenticate user
@app.route('/login.html', methods=['GET', 'POST'])
def login():

    # Declare the login form
    form = LoginForm(request.form)

    # Flask message injected into the page, in case of any errors
    msg = None

    # check if both http method is POST and form is valid on submit
    if form.validate_on_submit():
```

```
        # assign form data to variables
        username = request.form.get('username', '', type=str)
        password = request.form.get('password', '', type=str)

        # filter User out of database through username
        user = User.query.filter_by(user=username).first()

        if user:

            #if bc.check_password_hash(user.password, password):
            if user.password == password:
                login_user(user)
                return redirect(url_for('index'))
            else:
                msg = "Wrong password. Please try again."
        else:
            msg = "Unknown user"

    return render_template( 'pages/login.html', form=form, msg=msg )

# App main route + generic routing
@app.route('/', defaults={'path': 'index.html'})
@app.route('/<path>')
def index(path):

    if not current_user.is_authenticated:
        return redirect(url_for('login'))

    content = None

    try:

        # @WIP to fix this
        # Temporary solution to solve the dependencies
        if  path.endswith(('.png',  '.svg'  '.ttf',  '.xml',  '.ico',  '.woff',
'.woff2')):
            return  send_from_directory(os.path.join(app.root_path,  'static'),
path)

        # try to match the pages defined in -> pages/<input file>
        return render_template( 'pages/'+path )

    except:

        return render_template('layouts/auth-default.html',
                               content=render_template( 'pages/404.html' ) )
```

Lampiran 2. *Source Code index.html*

```
{% extends "layouts/default.html" %}

{% block title %} Dashboard {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

            <div class="row gap-20 masonry pos-r">
              <div class="masonry-sizer col-md-6"></div>
              <div class="masonry-item w-100">
                <div class="row gap-20">
                  <!-- Missing value handled ==================== -->
                  <div class="col-md-3">
                    <div class="layers bd bgc-white p-20">
                      <div class="layer w-100 mB-10">
                      <a href="datasets-handled.html">
                        <h6 class="lh-1">Missing Value Handling</h6>
                        </a>
                      </div>
                      <div class="layer w-100">
                        <div class="peers ai-sb fxw-nw">
                          <div class="peer peer-greed">
                            <span id="sparklinedash"></span>
                          </div>
                          <div class="peer">
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>

                  <!-- kmeans clustering ==================== -->
                  <div class="col-md-3">
                    <div class="layers bd bgc-white p-20">
                      <div class="layer w-100 mB-10">
                      <a href="kmeans.html">
                        <h6 class="lh-1">K-Means</h6>
                        </a>
                      </div>
                      <div class="layer w-100">
                        <div class="peers ai-sb fxw-nw">
                          <div class="peer peer-greed">
                            <span id="sparklinedash2"></span>
                          </div>
                          <div class="peer">
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>

                  <!-- P S O ==================== -->
                  <div class="col-md-3">
                    <div class="layers bd bgc-white p-20">
                      <div class="layer w-100 mB-10">
                        <h6 class="lh-1">P S O</h6>
                      </div>
```

```
                      <div class="layer w-100">
                        <div class="peers ai-sb fxw-nw">
                          <div class="peer peer-greed">
                            <span id="sparklinedash3"></span>
                          </div>
                          <div class="peer">
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>

                  <!-- Purposed method ==================== -->
                  <div class="col-md-3">
                    <div class="layers bd bgc-white p-20">
                      <div class="layer w-100 mB-10">
                        <h6 class="lh-1">Purposed Method</h6>
                      </div>
                      <div class="layer w-100">
                        <div class="peers ai-sb fxw-nw">
                          <div class="peer peer-greed">
                            <span id="sparklinedash4"></span>
                          </div>
                          <div class="peer">

                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
              <div class="masonry-item col-12">

                          <div class="layers">
                            <div class="layer w-100">
                              <h5 class="mB-5">400</h5>
                              <small      class="fw-600      c-grey-700">Sample
Data</small>
                              <span    class="pull-right    c-grey-600    fsz-
sm">100%</span>
                              <div class="progress mT-10">
                                <div   class="progress-bar   bgc-deep-blue-500"
role="progressbar"  aria-valuenow="50"  aria-valuemin="0"  aria-valuemax="100"
style="width:100%;"> <span class="sr-only"></span></div>
                              </div>
                            </div>
                            <div class="layer w-100 mT-15">
                              <h5 class="mB-5">280</h5>
                              <small       class="fw-600       c-grey-700">Data
Training</small>
                              <span    class="pull-right    c-grey-600    fsz-
sm">70%</span>
                              <div class="progress mT-10">
                                <div     class="progress-bar     bgc-blue-500"
role="progressbar"  aria-valuenow="50"  aria-valuemin="0"  aria-valuemax="100"
style="width:70%;"> <span class="sr-only"></span></div>
                              </div>
                            </div>
                            <div class="layer w-100 mT-15">
```

```
                              <h5 class="mB-5">120</h5>
                              <small     class="fw-600     c-grey-700">Testing
Data</small>
                              <span    class="pull-right    c-grey-600    fsz-
sm">30%</span>
                              <div class="progress mT-10">
                                <div  class="progress-bar  bgc-light-blue-500"
role="progressbar"  aria-valuenow="50"  aria-valuemin="0"  aria-valuemax="100"
style="width:30%;"> <span class="sr-only"></span></div>
                              </div>
                            </div>
                            <div class="layer w-100 mT-15">
                              <h5 class="mB-5">Missing Value</h5>
                              <small   class="fw-600   c-grey-700">of   Chronic
Kidney Disease Datasets</small>
                              <span    class="pull-right    c-grey-600    fsz-
sm">15%</span>
                              <div class="progress mT-10">
                                <div   class="progress-bar   bgc-blue-grey-500"
role="progressbar"  aria-valuenow="50"  aria-valuemin="0"  aria-valuemax="100"
style="width:15%;"> <span class="sr-only"></span></div>
                              </div>
                            </div>
                          </div>
              </div>

                    </div>
                  </div>
                </div>
              </div>
            </div>

{% endblock content %}
```

Lampiran 3. *Source Code datasets-original.html*

```
{% extends "layouts/default.html" %}

{% block title %} Original Datasets {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}
                    <div class="container-fluid">
                    <h4 class="c-grey-900 mT-10 mB-30">Dataset Original</h4>
                    <div class="row">
                    <div class="col-md-12">
                    <div class="bgc-white bd bdrs-3 p-20 mB-20">
                        <table   class="table   table-striped   table-bordered
table-responsive" id="dataTable" cellspacing="0" width="100%">
                            <thead class="thead-dark">
                                <tr>
                                    <th>Id</th>
                                    <th>Age</th>
                                    <th>Blood Pleasure</th>
                                    <th>Specific gravity</th>
                                    <th>Marginal Adhesion</th>
                                    <th>Albumin</th>
                                    <th>Sugar</th>
                                    <th>Red Blood Cells</th>
                                    <th>Pus Cells</th>
                                    <th>Bacteria</th>
                                    <th>Blood Glucose Random</th>
                                    <th>Blood Urea</th>
                                    <th>Serum Creatinine</th>
                                    <th>Sodium</th>
                                    <th>Potassium</th>
                                    <th>Hemoglobin</th>
                                    <th>Packed Cell Volume</th>
                                    <th>White Blood Cell Count</th>
                                    <th>Red Blood Cell Count</th>
                                    <th>Hypertension</th>
                                    <th>Diabetes Mellitus</th>
                                    <th>Coronary Artery Disease</th>
                                    <th>Appetite</th>
                                    <th>Pedal Edema</th>
                                    <th>Anemia</th>
                                    <th>Class</th>
                                </tr>
                            </thead>

                            <tbody>
                                {% for i in range(n1): %}
                                    <tr>
                                        <td>{{ df.iloc[i,0] }}</td>
                                        <td>{{ df.iloc[i,1] }}</td>
                                        <td>{{ df.iloc[i,2] }}</td>
                                        <td>{{ df.iloc[i,3] }}</td>
                                        <td>{{ df.iloc[i,4] }}</td>
                                        <td>{{ df.iloc[i,5] }}</td>
                                        <td>{{ df.iloc[i,6] }}</td>
                                        <td>{{ df.iloc[i,7] }}</td>
                                        <td>{{ df.iloc[i,8] }}</td>
```

```
                                        <td>{{ df.iloc[i,9] }}</td>
                                        <td>{{ df.iloc[i,10] }}</td>
                                        <td>{{ df.iloc[i,11] }}</td>
                                        <td>{{ df.iloc[i,12] }}</td>
                                        <td>{{ df.iloc[i,13] }}</td>
                                        <td>{{ df.iloc[i,14] }}</td>
                                        <td>{{ df.iloc[i,15] }}</td>
                                        <td>{{ df.iloc[i,16] }}</td>
                                        <td>{{ df.iloc[i,17] }}</td>
                                        <td>{{ df.iloc[i,18] }}</td>
                                        <td>{{ df.iloc[i,19] }}</td>
                                        <td>{{ df.iloc[i,20] }}</td>
                                        <td>{{ df.iloc[i,21] }}</td>
                                        <td>{{ df.iloc[i,22] }}</td>
                                        <td>{{ df.iloc[i,23] }}</td>
                                        <td>{{ df.iloc[i,24] }}</td>
                                        <td>{{ df.iloc[i,25] }}</td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                    </div>
                    </div>
                    </div>

{% endblock content %}
```

Lampiran 4. *Source Code datasets-handled.html*

```
{% extends "layouts/default.html" %}

{% block title %} Blank Page {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}
                    <div class="container-fluid">
                    <h4 class="c-grey-900 mT-10 mB-30">Dataset Handled</h4>
                    <div class="row">
                    <div class="col-md-12">
                    <div class="bgc-white bd bdrs-3 p-20 mB-20">
                        <table   class="table   table-striped   table-bordered
table-responsive" id="dataTable" cellspacing="0" width="100%">
                            <thead class="thead-dark">
                                <tr>
                                    <th>Id</th>
                                    <th>Age</th>
                                    <th>Blood Pleasure</th>
                                    <th>Specific gravity</th>
                                    <th>Marginal Adhesion</th>
                                    <th>Albumin</th>
                                    <th>Sugar</th>
                                    <th>Red Blood Cells</th>
                                    <th>Pus Cells</th>
                                    <th>Bacteria</th>
                                    <th>Blood Glucose Random</th>
                                    <th>Blood Urea</th>
                                    <th>Serum Creatinine</th>
                                    <th>Sodium</th>
                                    <th>Potassium</th>
                                    <th>Hemoglobin</th>
                                    <th>Packed Cell Volume</th>
                                    <th>White Blood Cell Count</th>
                                    <th>Red Blood Cell Count</th>
                                    <th>Hypertension</th>
                                    <th>Diabetes Mellitus</th>
                                    <th>Coronary Artery Disease</th>
                                    <th>Appetite</th>
                                    <th>Pedal Edema</th>
                                    <th>Anemia</th>
                                    <th>Class</th>
                                </tr>
                            </thead>
                            <tbody>
                                {% for i in range(n2): %}
                                    <tr>
                                        <td>{{ df1.iloc[i,0] }}</td>
                                        <td>{{ df1.iloc[i,1] }}</td>
                                        <td>{{ df1.iloc[i,2] }}</td>
                                        <td>{{ df1.iloc[i,3] }}</td>
                                        <td>{{ df1.iloc[i,4] }}</td>
                                        <td>{{ df1.iloc[i,5] }}</td>
                                        <td>{{ df1.iloc[i,6] }}</td>
                                        <td>{{ df1.iloc[i,7] }}</td>
                                        <td>{{ df1.iloc[i,8] }}</td>
                                        <td>{{ df1.iloc[i,9] }}</td>
```

```
                                        <td>{{ df1.iloc[i,10] }}</td>
                                        <td>{{ df1.iloc[i,11] }}</td>
                                        <td>{{ df1.iloc[i,12] }}</td>
                                        <td>{{ df1.iloc[i,13] }}</td>
                                        <td>{{ df1.iloc[i,14] }}</td>
                                        <td>{{ df1.iloc[i,15] }}</td>
                                        <td>{{ df1.iloc[i,16] }}</td>
                                        <td>{{ df1.iloc[i,17] }}</td>
                                        <td>{{ df1.iloc[i,18] }}</td>
                                        <td>{{ df1.iloc[i,19] }}</td>
                                        <td>{{ df1.iloc[i,20] }}</td>
                                        <td>{{ df1.iloc[i,21] }}</td>
                                        <td>{{ df1.iloc[i,22] }}</td>
                                        <td>{{ df1.iloc[i,23] }}</td>
                                        <td>{{ df1.iloc[i,24] }}</td>
                                        <td>{{ df1.iloc[i,25] }}</td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                    </div>
                    </div>
                    </div>
                    </div>

{% endblock content %}
```

Lampiran 5. *Source Code kmeans.html*

```
{% extends "layouts/default.html" %}

{% block title %} K-Means {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

                    <div class="container-fluid">
                    <h4 class="c-grey-900 mT-10 mB-30">K-Means</h4>
                    <h8  class="c-grey-900  mT-10  mB-30">Dataset  Setelah  di
Cluster</h8>
                    <div class="row">
                    <div class="col-md-12">
                    <div class="bgc-white bd bdrs-3 p-20 mB-20">
                        <table  class="table  dataTable  table-striped  table-
bordered   table-responsive"   id="dataTable"   cellspacing="0"   width="100%"
align="center">
                            <thead class="thead-dark">
                                <tr>
                                    <th>Id</th>
                                    <th>Age</th>
                                    <th>Blood Pleasure</th>
                                    <th>Specific gravity</th>
                                    <th>Albumin</th>
                                    <th>Sugar</th>
                                    <th>Red Blood Cells</th>
                                    <th>Pus Cells</th>
                                    <th>Pus Cells Clumps</th>
                                    <th>Bacteria</th>
                                    <th>Blood Glucose Random</th>
                                    <th>Blood Urea</th>
                                    <th>Serum Creatinine</th>
                                    <th>Sodium</th>
                                    <th>Potassium</th>
                                    <th>Hemoglobin</th>
                                    <th>Packed Cell Volume</th>
                                    <th>White Blood Cell Count</th>
                                    <th>Red Blood Cell Count</th>
                                    <th>Hypertension</th>
                                    <th>Diabetes Mellitus</th>
                                    <th>Coronary Artery Disease</th>
                                    <th>Appetite</th>
                                    <th>Pedal Edema</th>
                                    <th>Anemia</th>
                                    <th>Class</th>
                                </tr>
                            </thead>
                            <tbody>
                                {% for i in range(n5): %}
                                    <tr>
                                        <td>{{ df1.iloc[i,0] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,1] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,2] }}</td>
                                        <td>{{ df1.iloc[i,3] }}</td>
```

```
                                        <td>{{ df1.iloc[i,4] }}</td>
                                        <td>{{ df1.iloc[i,5] }}</td>
                                        <td>{{ df1.iloc[i,6] }}</td>
                                        <td>{{ df1.iloc[i,7] }}</td>
                                        <td>{{ df1.iloc[i,8] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,9] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,10] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,11] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,12] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,13] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,14] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,15] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,16] }}</td>
                                        <td           class="warnain-kolom">{{
df1.iloc[i,17] }}</td>
                                        <td>{{ df1.iloc[i,18] }}</td>
                                        <td>{{ df1.iloc[i,19] }}</td>
                                        <td>{{ df1.iloc[i,20] }}</td>
                                        <td>{{ df1.iloc[i,21] }}</td>
                                        <td>{{ df1.iloc[i,22] }}</td>
                                        <td>{{ df1.iloc[i,23] }}</td>
                                        <td>{{ df1.iloc[i,24] }}</td>
                                        <td>{{ df1.iloc[i,25] }}</td>
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                </div>
                </div>
                </div>
                </div>
{% endblock content %}
```

Lampiran 6. *Source Code pso.html*

```
{% extends "layouts/default.html" %}

{% block title %} Blank Page {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}
                <div class="container-fluid">
                    <h4 class="c-grey-900 mT-10 mB-30">PSO</h4>
                    <h6 class="c-grey-900 mT-10 mB-30">Fitur terpilih</h6>
                    <div class="row">
                    <div class="col-md-12">
                    <div class="bgc-white bd bdrs-3 p-20 mB-20">
                        <table   class="table   table-striped   table-bordered
table-responsive" id="dataTable" cellspacing="0" width="100%">
                            <thead class="thead-dark">
                                <tr>
                                    {% for i in range(nAttr): %}
                                    <td><strong>{{    attrSelected.iloc[i,0]}}
</strong></td>
                                    {% endfor %}
                                </tr>
                            </thead>
                            <tbody>
                                {% for i in range(n4): %}
                                    <tr>
                                    {% for j in range(nAttr): %}
                                        <td><center>{{       dataPSO.iloc[i,j]
}}</center></td>
                                    {% endfor %}
                                    </tr>
                                {% endfor %}
                            </tbody>
                        </table>
                        <h3> Hasil Akurasi SVM PSO = {{pso}}%</h3>
                        <h4> Waktu Komputasi = {{waktu}} Detik</h4>
                    </div>
                    </div>
                    </div>
                    </div>


{% endblock content %}
```

Lampiran 7. *Source Code proposed.html*

```
{% extends "layouts/default.html" %}

{% block title %} Blank Page {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

<div class="container-fluid">
<table    class="table    table-striped    table-bordered    table-responsive"
id="dataTable" cellspacing="0" width="100%">
<thead class="thead-dark">
    <tr><center><h6> Hasil akurasi Algoritma SVM </h6></center>
    <center><h3>{{hasilsvm}}%</h3></center>
    <center><h6>  Hasil  akurasi  Algoritma  SVM  menggunakan  K-Means  dan
PSO</h6></center>
    <center><h3>{{hasil}}%</h3></center>
    <center><h6>  Waktu  Komputasi  Algoritma  SVM  menggunakan  K-Means  dan
PSO</h6></center>
    <center><h3>{{waktu}} Detik</h3></center>
</table>
</div>
{% endblock content %}
```

Lampiran 8. *Source Code about.html*

```
{% extends "layouts/default.html" %}

{% block title %} Blank Page {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}


    <h3>
    Diaginjal
    </h3>
    <p>
       Diaginjal adalah sistem aplikasi yang digunakan untuk mendiagnosis
penyakit ginjal kronis dengan menggunakan dataset <i> Chronic Kidney Disease
Dataset.</i> Aplikasi ini juga merupakan salah satu syarat Skripsi dari Hafid
Alpin Al Gazni sebagai syarat memperoleh gelar S1 Ilmu Komputer Universitas
Negeri Semarang dengan judul skripsi <strong>"Optimasi Algoritma <i>Support
Vector Machine</i> menggunakan Algoritma K-Means <i>Clustering</i> dan
<i>Particle Swarm Optimization</i> untuk Diagnosis Penyakit Ginjal
Kronis"</strong>

       Penelitian ini menggunakan data sekunder yaitu <i>Chronic Kidney
Disease Dataset</i> yang diperoleh dari UCI <i>Learning Repository</i> dengan
tahapan penelitian sebagai berikut:
       <ul> 1. Melakukan Pengambilan <i>Dataset "Chronic Kidney Disease
Dataset".</i></ul>
       <ul> 2. Melakukan penanganan <i>missing value.</i></ul>
       <ul> 3. Melakukan kluster pada data yang bertipe kontinu menggunakan
K-Means<i> Clustering.</i></ul>
       <ul> 4. Melakukan seleksi fitur terbaik menggunakan<i> Particle Swarm
Optimization.</i></ul>
       <ul> 5. Melakukan <i> Splitting Data</i> menjadi dua bagian yaitu:
Data latih (70%) dan Data Uji (30%).</ul>
       <ul> 6. Melakukan Klasifikasi menggunkan <i>Support Vector
Machine.</i></ul>
       <ul> 7. Melakukan pengujian model klasifikasi menggunakan <i>Confusion
Matrix.</i></ul>

    </p>

{% endblock content %}
```

Lampiran 9. *Source Code author-profile.html*

```
{% extends "layouts/default.html" %}

{% block title %} Blank Page {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}



   <div class="peer mR-10">
                 <center>

                         <img class="w-2000r bdrs-50p" <img style="margin-
top:10px" src="/static/assets/static/images/aku1.jpg" alt="">

                         <strong><h2>Hafid Alpin Al Gazni</h2></strong>
                         <strong><h4>Ilmu Komputer</h4></strong>
                         <strong><h4>Teknik Informatika, S1</h4></strong>
                         <strong><h4>4611415016</h4></strong>
                         <strong><h3>Universitas Negeri Semarang</h3></strong>


                 </center>
               </div>

{% endblock content %}
```

Lampiran 10. *Source Code login.html*

```
{% extends "layouts/auth-default.html" %}

{% block title %} Login {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

    <div class="d-n@sm- peer peer-greed h-100 pos-r bgr-n bgpX-c bgpY-c bgsz-
cv" style='background-image: url("/static/assets/static/images/kidney.jpg")'>
        <div class="pos-a centerXY">
          <div class="bgc-white bdrs-50p pos-r" style="width: 120px; height:
120px;">
            <img class="pos-a centerXY"
src="/static/assets/static/images/logo3.png" alt="">
          </div>
        </div>
      </div>
      <div class="col-12 col-md-4 peer pX-40 pY-80 h-100 bgc-white scrollable
pos-r" style="min-width: 320px;">


        {% if msg %}
            <h4 class="fw-300 c-grey-900 mB-40">{{ msg | safe }}</h4>
        {% else %}
            <h4 class="fw-300 c-grey-900 mB-40">Login</h4>
        {% endif %}

        <form method="post" action="">

          {{ form.hidden_tag() }}

          <div class="form-group">
            <label class="text-normal text-dark">Username</label>
            {{ form.username(placeholder="Username", class="form-control") }}
          </div>

          <div class="form-group">
            <label class="text-normal text-dark">Password</label>
            {{ form.password(placeholder="Password", class="form-control",
type="password") }}
          </div>

          <div class="form-group">
            <div class="peers ai-c jc-sb fxw-nw">
              <div class="peer">
                <div class="checkbox checkbox-circle checkbox-info peers ai-
c">
                  <input type="checkbox" id="inputCall1"
name="inputCheckboxesCall" class="peer">
                  <label for="inputCall1" class="peers peer-greed js-sb ai-
c">
                    <span class="peer peer-greed">Remember Me</span>
                  </label>
                </div>
              </div>
              <div class="peer">
                <button type="submit" class="btn btn-primary">Login</button>
```

```
                   
                <span class="text-small font-weight-semibold">Not a
member?</span>
                <a href="/register.html" class="text-black text-
small">Register</a>
              </div>
            </div>
          </div>
        </form>
      </div>
    </div>

{% endblock content %}
```

Lampiran 11. *Source Code register.html*

```
{% extends "layouts/auth-default.html" %}

{% block title %} Register {% endblock title %}

{% block stylesheets %}{% endblock stylesheets %}

{% block content %}

      <div class="peer peer-greed h-100 pos-r bgr-n bgpX-c bgpY-c bgsz-cv"
style='background-image: url("/static/assets/static/images/kidney.jpg")'>
        <div class="pos-a centerXY">
          <div class="bgc-white bdrs-50p pos-r" style="width: 120px; height:
120px;">
            <img class="pos-a centerXY"
src="/static/assets/static/images/logo3.png" alt="">
          </div>
        </div>
      </div>
      <div class="col-12 col-md-4 peer pX-40 pY-80 h-100 bgc-white scrollable
pos-r" style="min-width: 320px;">

        {% if msg %}
            <h4 class="fw-300 c-grey-900 mB-40">{{ msg | safe }}</h4>
        {% else %}
            <h4 class="fw-300 c-grey-900 mB-40">Register</h4>
        {% endif %}

        <form method="post" action="">

          {{ form.hidden_tag() }}

          <div class="form-group">
            <label class="text-normal text-dark">Name</label>
            {{ form.name(placeholder="Name", class="form-control") }}
          </div>

          <div class="form-group">
            <label class="text-normal text-dark">Username</label>
            {{ form.username(placeholder="Username", class="form-control") }}
          </div>

          <div class="form-group">
            <label class="text-normal text-dark">Email Address</label>
            {{ form.email(placeholder="Email Address", class="form-control")
}}
          </div>

          <div class="form-group">
            <label class="text-normal text-dark">Password</label>
            {{ form.password(placeholder="Password", class="form-control",
type="password") }}
          </div>

          <div class="form-group">
            <button type="submit" class="btn btn-primary">Register</button>

               
```

```
            <span class="text-small font-weight-semibold">Already have and
account?</span>
            <a href="/login.html" class="text-black text-small">Login</a>
          </div>

        </form>
      </div>

{% endblock content %}
```

Lampiran 12. *Chronic Kidney Disease Datasets*

```
@relation Chronic_Kidney_Disease

@attribute 'age' numeric
@attribute 'bp'  numeric
@attribute 'sg' {1.005,1.010,1.015,1.020,1.025}
@attribute 'al' {0,1,2,3,4,5}
@attribute 'su' {0,1,2,3,4,5}
@attribute 'rbc' {normal,abnormal}
@attribute 'pc' {normal,abnormal}
@attribute 'pcc' {present,notpresent}
@attribute 'ba' {present,notpresent}
@attribute 'bgr'  numeric
@attribute 'bu' numeric
@attribute 'sc' numeric
@attribute 'sod' numeric
@attribute 'pot' numeric
@attribute 'hemo' numeric
@attribute 'pcv' numeric
@attribute 'wbcc' numeric
@attribute 'rbcc' numeric
@attribute 'htn' {yes,no}
@attribute 'dm' {yes,no}
@attribute 'cad' {yes,no}
@attribute 'appet' {good,poor}
@attribute 'pe' {yes,no}
@attribute 'ane' {yes,no}
@attribute 'class' {ckd,notckd}

@data
48,80,1.020,1,0,?,normal,notpresent,notpresent,121,36,1.2,?,?,15.4,44,7800,5.
2,yes,yes,no,good,no,no,ckd
7,50,1.020,4,0,?,normal,notpresent,notpresent,?,18,0.8,?,?,11.3,38,6000,?,no,
no,no,good,no,no,ckd
62,80,1.010,2,3,normal,normal,notpresent,notpresent,423,53,1.8,?,?,9.6,31,750
0,?,no,yes,no,poor,no,yes,ckd
48,70,1.005,4,0,normal,abnormal,present,notpresent,117,56,3.8,111,2.5,11.2,32
,6700,3.9,yes,no,no,poor,yes,yes,ckd
51,80,1.010,2,0,normal,normal,notpresent,notpresent,106,26,1.4,?,?,11.6,35,73
00,4.6,no,no,no,good,no,no,ckd
60,90,1.015,3,0,?,?,notpresent,notpresent,74,25,1.1,142,3.2,12.2,39,7800,4.4,
yes,yes,no,good,yes,no,ckd
68,70,1.010,0,0,?,normal,notpresent,notpresent,100,54,24.0,104,4.0,12.4,36,?,
?,no,no,no,good,no,no,ckd
24,?,1.015,2,4,normal,abnormal,notpresent,notpresent,410,31,1.1,?,?,12.4,44,6
900,5,no,yes,no,good,yes,no,ckd
52,100,1.015,3,0,normal,abnormal,present,notpresent,138,60,1.9,?,?,10.8,33,96
00,4.0,yes,yes,no,good,no,yes,ckd
53,90,1.020,2,0,abnormal,abnormal,present,notpresent,70,107,7.2,114,3.7,9.5,2
9,12100,3.7,yes,yes,no,poor,no,yes,ckd
50,60,1.010,2,4,?,abnormal,present,notpresent,490,55,4.0,?,?,9.4,28,?,?,yes,y
es,no,good,no,yes,ckd
63,70,1.010,3,0,abnormal,abnormal,present,notpresent,380,60,2.7,131,4.2,10.8,
32,4500,3.8,yes,yes,no,poor,yes,no,ckd
68,70,1.015,3,1,?,normal,present,notpresent,208,72,2.1,138,5.8,9.7,28,12200,3
.4,yes,yes,yes,poor,yes,no,ckd
68,70,?,?,?,?,?,notpresent,notpresent,98,86,4.6,135,3.4,9.8,?,?,?,yes,yes,yes
,poor,yes,no,ckd
```

```
68,80,1.010,3,2,normal,abnormal,present,present,157,90,4.1,130,6.4,5.6,16,110
00,2.6,yes,yes,yes,poor,yes,no,ckd
40,80,1.015,3,0,?,normal,notpresent,notpresent,76,162,9.6,141,4.9,7.6,24,3800
,2.8,yes,no,no,good,no,yes,ckd
47,70,1.015,2,0,?,normal,notpresent,notpresent,99,46,2.2,138,4.1,12.6,?,?,?,n
o,no,no,good,no,no,ckd
47,80,?,?,?,?,?,notpresent,notpresent,114,87,5.2,139,3.7,12.1,?,?,?,yes,no,no
,poor,no,no,ckd
60,100,1.025,0,3,?,normal,notpresent,notpresent,263,27,1.3,135,4.3,12.7,37,11
400,4.3,yes,yes,yes,good,no,no,ckd
62,60,1.015,1,0,?,abnormal,present,notpresent,100,31,1.6,?,?,10.3,30,5300,3.7
,yes,no,yes,good,no,no,ckd
61,80,1.015,2,0,abnormal,abnormal,notpresent,notpresent,173,148,3.9,135,5.2,7
.7,24,9200,3.2,yes,yes,yes,poor,yes,yes,ckd
60,90,?,?,?,?,?,notpresent,notpresent,?,180,76,4.5,?,10.9,32,6200,3.6,yes,yes
,yes,good,no,no,ckd
48,80,1.025,4,0,normal,abnormal,notpresent,notpresent,95,163,7.7,136,3.8,9.8,
32,6900,3.4,yes,no,no,good,no,yes,ckd
21,70,1.010,0,0,?,normal,notpresent,notpresent,?,?,?,?,?,?,?,?,?,no,no,no,poo
r,no,yes,ckd
42,100,1.015,4,0,normal,abnormal,notpresent,present,?,50,1.4,129,4.0,11.1,39,
8300,4.6,yes,no,no,poor,no,no,ckd
61,60,1.025,0,0,?,normal,notpresent,notpresent,108,75,1.9,141,5.2,9.9,29,8400
,3.7,yes,yes,no,good,no,yes,ckd
75,80,1.015,0,0,?,normal,notpresent,notpresent,156,45,2.4,140,3.4,11.6,35,103
00,4,yes,yes,no,poor,no,no,ckd
69,70,1.010,3,4,normal,abnormal,notpresent,notpresent,264,87,2.7,130,4.0,12.5
,37,9600,4.1,yes,yes,yes,good,yes,no,ckd
75,70,?,1,3,?,?,notpresent,notpresent,123,31,1.4,?,?,?,?,?,?,no,yes,no,good,n
o,no,ckd
68,70,1.005,1,0,abnormal,abnormal,present,notpresent,?,28,1.4,?,?,12.9,38,?,?
,no,no,yes,good,no,no,ckd
?,70,?,?,?,?,?,notpresent,notpresent,93,155,7.3,132,4.9,?,?,?,?,yes,
yes,no,good,no,no,ckd
73,90,1.015,3,0,?,abnormal,present,notpresent,107,33,1.5,141,4.6,10.1,30,7800
,4,no,no,no,poor,no,no,ckd
61,90,1.010,1,1,?,normal,notpresent,notpresent,159,39,1.5,133,4.9,11.3,34,960
0,4.0,yes,yes,no,poor,no,no,ckd
60,100,1.020,2,0,abnormal,abnormal,notpresent,notpresent,140,55,2.5,?,?,10.1,
29,?,?,yes,no,no,poor,no,no,ckd
70,70,1.010,1,0,normal,?,present,present,171,153,5.2,?,?,?,?,?,?,no,yes,no,po
or,no,no,ckd
65,90,1.020,2,1,abnormal,normal,notpresent,notpresent,270,39,2.0,?,?,12.0,36,
9800,4.9,yes,yes,no,poor,no,yes,ckd
76,70,1.015,1,0,normal,normal,notpresent,notpresent,92,29,1.8,133,3.9,10.3,32
,?,?,yes,no,no,good,no,no,ckd
72,80,?,?,?,?,?,notpresent,notpresent,137,65,3.4,141,4.7,9.7,28,6900,2.5,yes,
yes,no,poor,no,yes,ckd
69,80,1.020,3,0,abnormal,normal,notpresent,notpresent,?,103,4.1,132,5.9,12.5,
?,?,?,yes,no,no,good,no,no,ckd
82,80,1.010,2,2,normal,?,notpresent,notpresent,140,70,3.4,136,4.2,13.0,40,980
0,4.2,yes,yes,no,good,no,no,ckd
46,90,1.010,2,0,normal,abnormal,notpresent,notpresent,99,80,2.1,?,?,11.1,32,9
100,4.1,yes,no,       no,good,no,no,ckd
45,70,1.010,0,0,?,normal,notpresent,notpresent,?,20,0.7,?,?,?,?,?,?,no,no,no,
good,yes,no,ckd
47,100,1.010,0,0,?,normal,notpresent,notpresent,204,29,1.0,139,4.2,9.7,33,920
0,4.5,yes,no,no,good,no,yes,ckd
35,80,1.010,1,0,abnormal,?,notpresent,notpresent,79,202,10.8,134,3.4,7.9,24,7
900,3.1,no,yes,no,good,no,no,ckd
```

```
54,80,1.010,3,0,abnormal,abnormal,notpresent,notpresent,207,77,6.3,134,4.8,9.
7,28,?,?,yes,yes,no,poor,yes,no,ckd
54,80,1.020,3,0,?,abnormal,notpresent,notpresent,208,89,5.9,130,4.9,9.3,?,?,?
,yes,yes,no,poor,yes,no,ckd
48,70,1.015,0,0,?,normal,notpresent,notpresent,124,24,1.2,142,4.2,12.4,37,640
0,4.7,no,yes,no,good,no,no,ckd
11,80,1.010,3,0,?,normal,notpresent,notpresent,?,17,0.8,?,?,15.0,45,8600,?,no
,no,no,good,no,no,ckd
73,70,1.005,0,0,normal,normal,notpresent,notpresent,70,32,0.9,125,4.0,10.0,29
,18900,3.5,yes,yes,no,good,yes,no,ckd
60,70,1.010,2,0,normal,abnormal,present,notpresent,144,72,3.0,?,?,9.7,29,2160
0,3.5,yes,yes,no,poor,no,yes,ckd
53,60,?,?,?,?,?,notpresent,notpresent,91,114,3.25,142,4.3,8.6,28,11000,3.8,ye
s,yes,no,poor,yes,yes,ckd
54,100,1.015,3,0,?,normal,present,notpresent,162,66,1.6,136,4.4,10.3,33,?,?,y
es,yes,no,poor,yes,no,ckd
53,90,1.015,0,0,?,normal,notpresent,notpresent,?,38,2.2,?,?,10.9,34,4300,3.7,
no,no,no,poor,no,yes,ckd
62,80,1.015,0,5,?,?,notpresent,notpresent,246,24,1.0,?,?,13.6,40,8500,4.7,yes
,yes,no,good,no,no,ckd
63,80,1.010,2,2,normal,?,notpresent,notpresent,?,?,3.4,136,4.2,13.0,40,9800,4
.2,yes,no,yes,good,no,no,ckd
35,80,1.005,3,0,abnormal,normal,notpresent,notpresent,?,?,?,?,?,9.5,28,?,?,no
,no,no,good,yes,no,ckd
76,70,1.015,3,4,normal,abnormal,present,notpresent,?,164,9.7,131,4.4,10.2,30,
11300,3.4,yes,yes,yes,poor,yes,no,ckd
76,90,?,?,?,?,normal,notpresent,notpresent,93,155,7.3,132,4.9,?,?,?,?,yes,yes
,yes,poor,no,no,ckd
73,80,1.020,2,0,abnormal,abnormal,notpresent,notpresent,253,142,4.6,138,5.8,1
0.5,33,7200,4.3,yes,yes,yes,good,no,no,ckd
59,100,?,?,?,?,?,notpresent,notpresent,?,96,6.4,?,?,6.6,?,?,?,yes,yes,no,good
,no,yes,ckd
67,90,1.020,1,0,?,abnormal,present,notpresent,141,66,3.2,138,6.6,?,?,?,?,yes,
no,no,good,no,no,ckd
67,80,1.010,1,3,normal,abnormal,notpresent,notpresent,182,391,32.0,163,39.0,?
,?,?,?,no,no,no,good,yes,no,ckd
15,60,1.020,3,0,?,normal,notpresent,notpresent,86,15,0.6,138,4.0,11.0,33,7700
,3.8,yes,yes,no,good,no,no,ckd
46,70,1.015,1,0,abnormal,normal,notpresent,notpresent,150,111,6.1,131,3.7,7.5
,27,?,?,no,no,no,good,no,yes,ckd
55,80,1.010,0,0,?,normal,notpresent,notpresent,146,?,?,?,?,9.8,?,?,?,no,no,
        no,good,no,no,ckd
44,90,1.010,1,0,?,normal,notpresent,notpresent,?,20,1.1,?,?,15.0,48,?,?,no,
        no,no,good,no,no,ckd
67,70,1.020,2,0,abnormal,normal,notpresent,notpresent,150,55,1.6,131,4.8,?,
        ?,?,?,yes,yes,no,good,yes,no,ckd
45,80,1.020,3,0,normal,abnormal,notpresent,notpresent,425,?,?,?,?,?,?,?,?,no,
no,no,poor,no,no,ckd
65,70,1.010,2,0,?,normal,present,notpresent,112,73,3.3,?,?,10.9,37,?,?,no,no,
no,good,no,no,ckd
26,70,1.015,0,4,?,normal,notpresent,notpresent,250,20,1.1,?,?,15.6,52,6900,6.
0,no,yes,no,good,no,no,ckd,
61,80,1.015,0,4,?,normal,notpresent,notpresent,360,19,0.7,137,4.4,15.2,44,830
0,5.2,yes,yes,no,good,no,no,ckd
46,60,1.010,1,0,normal,normal,notpresent,notpresent,163,92,3.3,141,4.0,9.8,28
,14600,3.2,yes,yes,no,good,no,no,ckd
64,90,1.010,3,3,?,abnormal,present,notpresent,?,35,1.3,?,?,10.3,?,?,?,yes,yes
,no,good,yes,no,ckd,
?,100,1.015,2,0,abnormal,abnormal,notpresent,notpresent,129,107,6.7,132,4.4,4
.8,14,6300,?,yes,no,no,good,yes,yes,ckd
```

```
56,90,1.015,2,0,abnormal,abnormal,notpresent,notpresent,129,107,6.7,131,4.8,9
.1,29,6400,3.4,yes,no,no,good,no,no,ckd
5,?,1.015,1,0,?,normal,notpresent,notpresent,?,16,0.7,138,3.2,8.1,?,?,?,no,no
,no,good,no,yes,ckd
48,80,1.005,4,0,abnormal,abnormal,notpresent,present,133,139,8.5,132,5.5,10.3
,36,    6200,4,no,yes,no,good,yes,no,ckd
67,70,1.010,1,0,?,normal,notpresent,notpresent,102,48,3.2,137,5.0,11.9,34,710
0,3.7,yes,yes,no,good,yes,no,ckd
70,80,?,?,?,?,?,notpresent,notpresent,158,85,3.2,141,3.5,10.1,30,?,?,yes,no,n
o,good,yes,no,ckd
56,80,1.010,1,0,?,normal,notpresent,notpresent,165,55,1.8,?,?,13.5,40,11800,5
.0,yes,yes,no,poor,yes,no,ckd
74,80,1.010,0,0,?,normal,notpresent,notpresent,132,98,2.8,133,5.0,10.8,31,940
0,3.8,yes,yes,no,good,no,no,ckd
45,90,?,?,?,?,?,notpresent,notpresent,360,45,2.4,128,4.4,8.3,29,5500,3.7,yes,
yes,no,good,no,no,ckd
38,70,?,?,?,?,?,notpresent,notpresent,104,77,1.9,140,3.9,?,?,?,?,yes,no,no,po
or,yes,no,ckd
48,70,1.015,1,0,normal,normal,notpresent,notpresent,127,19,1.0,134,3.6,?,?,?,
?,yes,yes,no,good,no,no,ckd
59,70,1.010,3,0,normal,abnormal,notpresent,notpresent,76,186,15,135,7.6,7.1,2
2,3800,2.1,yes,no,no,poor,yes,yes,ckd
70,70,1.015,2,?,?,?,notpresent,notpresent,?,46,1.5,?,?,9.9,?,?,?,no,yes,no,po
or,yes,no,ckd
56,80,?,?,?,?,?,notpresent,notpresent,415,37,1.9,?,?,?,?,?,?,no,yes,no,good,n
o,no,ckd
70,100,1.005,1,0,normal,abnormal,present,notpresent,169,47,2.9,?,?,11.1,32,58
00,5,yes,yes,no,poor,no,no,ckd
58,110,1.010,4,0,?,normal,notpresent,notpresent,251,52,2.2,?,?,?,?,13200,4.7,
yes,    yes,no,good,no,no,ckd
50,70,1.020,0,0,?,normal,notpresent,notpresent,109,32,1.4,139,4.7,?,?,?,?,no,
no,no,poor,no,no,ckd
63,100,1.010,2,2,normal,normal,notpresent,present,280,35,3.2,143,3.5,13.0,40,
9800,4.2,yes,no,yes,good,no,no,ckd
56,70,1.015,4,1,abnormal,normal,notpresent,notpresent,210,26,1.7,136,3.8,16.1
,52,12500,5.6,no,no,no,good,no,no,ckd
71,70,1.010,3,0,normal,abnormal,present,present,219,82,3.6,133,4.4,10.4,33,56
00,3.6,yes,yes,yes,good,no,no,ckd
73,100,1.010,3,2,abnormal,abnormal,present,notpresent,295,90,5.6,140,2.9,9.2,
30,7000,3.2,yes,yes,yes,poor,no,no,ckd
65,70,1.010,0,0,?,normal,notpresent,notpresent,93,66,1.6,137,4.5,11.6,36,1190
0,3.9,no,yes,no,good,no,no,ckd
62,90,1.015,1,0,?,normal,notpresent,notpresent,94,25,1.1,131,3.7,?,?,?,?,yes,
no,no,good,yes,yes,ckd
60,80,1.010,1,1,?,normal,notpresent,notpresent,172,32,2.7,?,?,11.2,36,?,?,no,
yes,yes,poor,no,no,ckd
65,60,1.015,1,0,?,normal,notpresent,notpresent,91,51,2.2,132,3.8,10.0,32,9100
,4.0,yes,yes,no,poor,yes,no,ckd
50,140,?,?,?,?,?,notpresent,notpresent,101,106,6.5,135,4.3,6.2,18,5800,2.3,ye
s,yes,no,poor,no,yes,ckd
56,180,?,0,4,?,abnormal,notpresent,notpresent,298,24,1.2,139,3.9,11.2,32,1040
0,4.2,yes,yes,no,poor,yes,no,ckd
34,70,1.015,4,0,abnormal,abnormal,notpresent,notpresent,153,22,0.9,133,3.8,?,
?,?,?,no,no,no,good,yes,no,ckd
71,90,1.015,2,0,?,abnormal,present,present,88,80,4.4,139,5.7,11.3,33,10700,3.
9,no,no,no,good,no,no,ckd
17,60,1.010,0,0,?,normal,notpresent,notpresent,92,32,2.1,141,4.2,13.9,52,7000
,?,no,no,no,good,no,no,ckd
76,70,1.015,2,0,normal,abnormal,present,notpresent,226,217,10.2,?,?,10.2,36,1
2700,4.2,yes,no,no,poor,yes,yes,ckd
```

```
55,90,?,?,?,?,?,notpresent,notpresent,143,88,2.0,?,?,?,?,?,?,yes,yes,no,poor,
yes,no,ckd
65,80,1.015,0,0,?,normal,notpresent,notpresent,115,32,11.5,139,4.0,14.1,42,68
00,5.2,no,no,no,good,no,no,ckd
50,90,?,?,?,?,?,notpresent,notpresent,89,118,6.1,127,4.4,6.0,17,6500,?,yes,ye
s,no,good,yes,yes,ckd
55,100,1.015,1,4,normal,?,notpresent,notpresent,297,53,2.8,139,4.5,11.2,34,13
600,4.4,yes,yes,no,good,no,no,ckd
45,80,1.015,0,0,?,abnormal,notpresent,notpresent,107,15,1.0,141,4.2,11.8,37,1
0200,4.2,no,no,no,good,no,no,ckd
54,70,?,?,?,?,?,notpresent,notpresent,233,50.1,1.9,?,?,11.7,?,?,?,no,yes,no,g
ood,no,no,ckd
63,90,1.015,0,0,?,normal,notpresent,notpresent,123,19,2.0,142,3.8,11.7,34,114
00,4.7,no,no,no,good,no,no,ckd
65,80,1.010,3,3,?,normal,notpresent,notpresent,294,71,4.4,128,5.4,10.0,32,900
0,3.9,yes,yes,yes,good,no,no,ckd
?,60,1.015,3,0,abnormal,abnormal,notpresent,notpresent,?,34,1.2,?,?,10.8,33,?
,?,no,no,no,good,no,no,ckd
61,90,1.015,0,2,?,normal,notpresent,notpresent,?,?,?,?,?,?,?,9800,?,no,yes,no
,poor,no,yes,ckd
12,60,1.015,3,0,abnormal,abnormal,present,notpresent,?,51,1.8,?,?,12.1,?,1030
0,?,no,no,no,good,no,no,ckd
47,80,1.010,0,0,?,abnormal,notpresent,notpresent,?,28,0.9,?,?,12.4,44,5600,4.
3,no,no,no,good,no,yes,ckd
?,70,1.015,4,0,abnormal,normal,notpresent,notpresent,104,16,0.5,?,?,?,?,?,?,n
o,no,no,good,yes,no,ckd
?,70,1.020,0,0,?,?,notpresent,notpresent,219,36,1.3,139,3.7,12.5,37,9800,4.4,
no,no,no,good,no,no,ckd
55,70,1.010,3,0,?,normal,notpresent,notpresent,99,25,1.2,?,?,11.4,?,?,?,no,no
,no,poor,yes,no,ckd
60,70,1.010,0,0,?,normal,notpresent,notpresent,140,27,1.2,?,?,?,?,?,?,no,no,n
o,good,no,no,ckd
72,90,1.025,1,3,?,normal,notpresent,notpresent,323,40,2.2,137,5.3,12.6,?,?,?,
no,yes,yes,poor,no,no,ckd
54,60,?,3,?,?,?,notpresent,notpresent,125,21,1.3,137,3.4,15.0,46,?,?,yes,yes,
no,good,yes,no,ckd
34,70,?,?,?,?,?,notpresent,notpresent,?,219,12.2,130,3.8,6.0,?,?,?,yes,no,no,
good,no,yes,ckd
43,80,1.015,2,3,?,abnormal,present,present,?,30,1.1,?,?,14.0,42,14900,?,no,no
,no,good,no,no,ckd
65,100,1.015,0,0,?,normal,notpresent,notpresent,90,98,2.5,?,?,9.1,28,5500,3.6
,yes,no,no,good,no,no,ckd
72,90,?,?,?,?,?,notpresent,notpresent,308,36,2.5,131,4.3,?,?,?,?,yes,yes,no,p
oor,no,no,ckd
70,90,1.015,0,0,?,normal,notpresent,notpresent,144,125,4.0,136,4.6,12.0,37,82
00,4.5,yes,yes,no,poor,yes,no,ckd
71,60,1.015,4,0,normal,normal,notpresent,notpresent,118,125,5.3,136,4.9,11.4,
35,15200,4.3,yes,yes,no,poor,yes,no,ckd
52,90,1.015,4,3,normal,abnormal,notpresent,notpresent,224,166,5.6,133,47,8.1,
23,5000,2.9,yes,yes,no,good,no,yes,ckd
75,70,1.025,1,0,?,normal,notpresent,notpresent,158,49,1.4,135,4.7,11.1,?,?,?,
yes,no,no,poor,yes,no,ckd
50,90,1.010,2,0,normal,abnormal,present,present,128,208,9.2,134,4.8,8.2,22,16
300,2.7,no,no,no,poor,yes,yes,ckd
5,50,1.010,0,0,?,normal,notpresent,notpresent,?,25,0.6,?,?,11.8,36,12400,?,no
,no,no,good,no,no,ckd
50,?,?,?,?,normal,?,notpresent,notpresent,219,176,13.8,136,4.5,8.6,24,13200,2
.7,yes,no,no,good,yes,yes,ckd
70,100,1.015,4,0,normal,normal,notpresent,notpresent,118,125,5.3,136,4.9,12.0
,37,    8400,8.0,yes,no,no,good,no,no,ckd
```

```
47,100,1.010,?,?,normal,?,notpresent,notpresent,122,?,16.9,138,5.2,10.8,33,10
200,3.8,no,yes,no,good,no,no,ckd
48,80,1.015,0,2,?,normal,notpresent,notpresent,214,24,1.3,140,4.0,13.2,39,?,?
,no,yes,no,poor,no,no,ckd
46,90,1.020,?,?,?,normal,notpresent,notpresent,213,68,2.8,146,6.3,9.3,?,?,?,y
es,yes,no,good,no,no,ckd
45,60,1.010,2,0,normal,abnormal,present,notpresent,268,86,4.0,134,5.1,10.0,29
,9200,?,yes,yes,no,good,no,no,ckd
73,?,1.010,1,0,?,?,notpresent,notpresent,95,51,1.6,142,3.5,?,?,?,?,no,
        no,no,good,no,no,ckd
41,70,1.015,2,0,?,abnormal,notpresent,present,?,68,2.8,132,4.1,11.1,33,?,?,ye
s,no,no,good,yes,yes,ckd
69,70,1.010,0,4,?,normal,notpresent,notpresent,256,40,1.2,142,5.6,?,?,?,?,no,
no,no,good,no,no,ckd
67,70,1.010,1,0,normal,normal,notpresent,notpresent,?,106,6.0,137,4.9,6.1,19,
6500,?,yes,no,no,good,no,yes,ckd
72,90,?,?,?,?,?,notpresent,notpresent,84,145,7.1,135,5.3,?,?,?,?,no,yes,no,go
od,no,no,ckd
41,80,1.015,1,4,abnormal,normal,notpresent,notpresent,210,165,18.0,135,4.7,?,
?,?,?,no,yes,no,good,no,no,ckd
60,90,1.010,2,0,abnormal,normal,notpresent,notpresent,105,53,2.3,136,5.2,11.1
,33,10500,4.1,no,no,no,good,no,no,ckd
57,90,1.015,5,0,abnormal,abnormal,notpresent,present,?,322,13.0,126,4.8,8.0,2
4,4200,3.3,yes,yes,yes,poor,yes,yes,ckd
53,100,1.010,1,3,abnormal,normal,notpresent,notpresent,213,23,1.0,139,4,?,?,?
,?,no,yes,no,good,no,no,ckd
60,60,1.010,3,1,normal,abnormal,present,notpresent,288,36,1.7,130,3.0,7.9,25,
15200,3.0,yes,no,no,poor,no,yes,ckd
69,60,?,?,?,?,?,notpresent,notpresent,171,26,48.1,?,?,?,?,?,?,yes,no,no,poor,
no,no,ckd
65,70,1.020,1,0,abnormal,abnormal,notpresent,notpresent,139,29,1.0,?,?,10.5,3
2,?,?,yes,no,no,good,yes,no,ckd
8,60,1.025,3,0,normal,normal,notpresent,notpresent,78,27,0.9,?,?,12.3,41,6700
,?,no,no,no,poor,yes,no,ckd
76,90,?,?,?,?,?,notpresent,notpresent,172,46,1.7,141,5.5,9.6,30,?,?,yes,yes,n
o,good,no,yes,ckd
39,70,1.010,0,0,?,normal,notpresent,notpresent,121,20,0.8,133,3.5,10.9,32,?,?
,no,yes,no,good,no,no,ckd
55,90,1.010,2,1,abnormal,abnormal,notpresent,notpresent,273,235,14.2,132,3.4,
8.3,22,14600,2.9,yes,yes,no,poor,yes,yes,ckd
56,90,1.005,4,3,abnormal,abnormal,notpresent,notpresent,242,132,16.4,140,4.2,
8.4,26,?,3,yes,yes,no,poor,yes,yes,ckd
50,70,1.020,3,0,abnormal,normal,present,present,123,40,1.8,?,?,11.1,36,4700,?
,no,no,no,good,no,no,ckd
66,90,1.015,2,0,?,normal,notpresent,present,153,76,3.3,?,?,?,?,?,?,no,no,no,p
oor,no,no,ckd
62,70,1.025,3,0,normal,abnormal,notpresent,notpresent,122,42,1.7,136,4.7,12.6
,39,7900,3.9,yes,yes,no,good,no,no,ckd
71,60,1.020,3,2,normal,normal,present,notpresent,424,48,1.5,132,4.0,10.9,31,?
,?,yes,yes,yes,good,no,no,ckd
59,80,1.010,1,0,abnormal,normal,notpresent,notpresent,303,35,1.3,122,3.5,10.4
,35,10900,4.3,no,yes,no,poor,no,no,ckd
81,60,?,?,?,?,?,notpresent,notpresent,148,39,2.1,147,4.2,10.9,35,9400,2.4,yes
,yes,yes,poor,yes,no,ckd
62,?,1.015,3,0,abnormal,?,notpresent,notpresent,?,?,?,?,?,14.3,42,10200,4.8,y
es,yes,no,good,no,no,ckd
59,70,?,?,?,?,?,notpresent,notpresent,204,34,1.5,124,4.1,9.8,37,6000,
        ?,no,yes,no,good,no,no,ckd
46,80,1.010,0,0,?,normal,notpresent,notpresent,160,40,2,140,4.1,9.0,27,8100,3
.2,yes,no,no,poor,no,yes,ckd
```

```
14,?,1.015,0,0,?,?,notpresent,notpresent,192,15,0.8,137,4.2,14.3,40,9500,5.4,
no,yes,no,poor,yes,no,ckd
60,80,1.020,0,2,?,?,notpresent,notpresent,?,?,?,?,?,?,?,?,?,no,yes,no,good,no
,no,ckd
27,60,?,?,?,?,?,notpresent,notpresent,76,44,3.9,127,4.3,?,?,?,?,no,no,no,poor
,yes,yes,ckd
34,70,1.020,0,0,abnormal,normal,notpresent,notpresent,139,19,0.9,?,?,12.7,42,
2200,?,no,no,no,poor,no,no,ckd
65,70,1.015,4,4,?,normal,present,notpresent,307,28,1.5,?,?,11.0,39,6700,?,yes
,yes,no,good,no,no,ckd
?,70,1.010,0,2,?,normal,notpresent,notpresent,220,68,2.8,?,?,8.7,27,?,?,yes,y
es,no,good,no,yes,ckd
66,70,1.015,2,5,?,normal,notpresent,notpresent,447,41,1.7,131,3.9,12.5,33,960
0,4.4,yes,yes,no,good,no,no,ckd
83,70,1.020,3,0,normal,normal,notpresent,notpresent,102,60,2.6,115,5.7,8.7,26
,12800,3.1,yes,no,no,poor,no,yes,ckd
62,80,1.010,1,2,?,?,notpresent,notpresent,309,113,2.9,130,2.5,10.6,34,12800,4
.9,no,no,no,good,no,no,ckd
17,70,1.015,1,0,abnormal,normal,notpresent,notpresent,22,1.5,7.3,145,2.8,13.1
,41,11200,?,no,no,no,good,no,no,ckd
54,70,?,?,?,?,?,notpresent,notpresent,111,146,7.5,141,4.7,11.0,35,8600,4.6,no
,no,no,good,no,no,ckd
60,50,1.010,0,0,?,normal,notpresent,notpresent,261,58,2.2,113,3.0,?,?,4200,3.
4,yes,no,no,good,no,no,ckd
21,90,1.010,4,0,normal,abnormal,present,present,107,40,1.7,125,3.5,8.3,23,124
00,3.9,no,no,no,good,no,yes,ckd
65,80,1.015,2,1,normal,normal,present,notpresent,215,133,2.5,?,?,13.2,41,?,?,
no,yes,no,good,no,no,ckd
42,90,1.020,2,0,abnormal,abnormal,present,notpresent,93,153,2.7,139,4.3,9.8,3
4,9800,?,no,no,no,poor,yes,yes,ckd
72,90,1.010,2,0,?,abnormal,present,notpresent,124,53,2.3,?,?,11.9,39,?,?,no,n
o,no,good,no,no,ckd
73,90,1.010,1,4,abnormal,abnormal,present,notpresent,234,56,1.9,?,?,10.3,28,?
,?,no,yes,no,good,no,no,ckd
45,70,1.025,2,0,normal,abnormal,present,notpresent,117,52,2.2,136,3.8,10.0,30
,19100,3.7,no,no,no,good,no,no,ckd
61,80,1.020,0,0,?,normal,notpresent,notpresent,131,23,0.8,140,4.1,11.3,35,?,?
,no,no,no,good,no,no,ckd
30,70,1.015,0,0,?,normal,notpresent,notpresent,101,106,6.5,135,4.3,?,?,?,?,no
,no,no,poor,no,no,ckd
54,60,1.015,3,2,?,abnormal,notpresent,notpresent,352,137,3.3,133,4.5,11.3,31,
5800,3.6,yes,yes,yes,poor,yes,no,ckd
4,?,1.020,1,0,?,normal,notpresent,notpresent,99,23,0.6,138,4.4,12,34,
        ?,?,no,no,no,good,no,no,ckd
8,50,1.020,4,0,normal,normal,notpresent,notpresent,?,46,1.0,135,3.8,?,?,?,?,n
o,no,no,good,yes,no,ckd
3,?,1.010,2,0,normal,normal,notpresent,notpresent,?,22,0.7,?,?,10.7,34,12300,
?,no,no,no,good,no,no,ckd
8,?,?,?,?,?,?,notpresent,notpresent,80,66,2.5,142,3.6,12.2,38,?,?,no,
        no,no,good,no,no,ckd
64,60,1.010,4,1,abnormal,abnormal,notpresent,present,239,58,4.3,137,5.4,9.5,2
9,7500,3.4,yes,yes,no,poor,yes,no,ckd
6,60,1.010,4,0,abnormal,abnormal,notpresent,present,94,67,1.0,135,4.9,9.9,30,
16700,4.8,no,no,no,poor,no,no,ckd
?,70,1.010,3,0,normal,normal,notpresent,notpresent,110,115,6.0,134,2.7,9.1,26
,9200,3.4,yes,yes,no,poor,no,no,ckd
46,110,1.015,0,0,?,normal,notpresent,notpresent,130,16,0.9,?,?,?,?,?,?,no,no,
no,good,no,no,ckd
32,90,1.025,1,0,abnormal,abnormal,notpresent,notpresent,?,223,18.1,113,6.5,5.
5,15,2600,2.8,yes,yes,no,poor,yes,yes,ckd
```

```
80,70,1.010,2,?,?,abnormal,notpresent,notpresent,?,49,1.2,?,?,?,?,?,?,yes,
        yes,no,good,no,no,ckd
70,90,1.020,2,1,abnormal,abnormal,notpresent,present,184,98.6,3.3,138,3.9,5.8
,?,?,?,yes,yes,yes,poor,no,no,ckd
49,100,1.010,3,0,abnormal,abnormal,notpresent,notpresent,129,158,11.8,122,3.2
,8.1,24,9600,3.5,yes,yes,no,poor,yes,yes,ckd
57,80,?,?,?,?,?,notpresent,notpresent,?,111,9.3,124,5.3,6.8,?,4300,3.0,yes,ye
s,no,good,no,yes,ckd
59,100,1.020,4,2,normal,normal,notpresent,notpresent,252,40,3.2,137,4.7,11.2,
30,26400,3.9,yes,yes,no,poor,yes,no,ckd
65,80,1.015,0,0,?,normal,notpresent,notpresent,92,37,1.5,140,5.2,8.8,25,10700
,3.2,yes,no,yes,good,yes,no,ckd
90,90,1.025,1,0,?,normal,notpresent,notpresent,139,89,3.0,140,4.1,12.0,37,790
0,3.9,yes,yes,no,good,no,no,ckd
64,70,?,?,?,?,?,notpresent,notpresent,113,94,7.3,137,4.3,7.9,21,?,?,yes,yes,y
es,good,yes,yes,ckd
78,60,?,?,?,?,?,notpresent,notpresent,114,74,2.9,135,5.9,8.0,24,?,?,no,yes,no
,good,no,yes,ckd
?,90,?,?,?,?,?,notpresent,notpresent,207,80,6.8,142,5.5,8.5,?,?,?,yes,yes,no,
good,no,yes,ckd
65,90,1.010,4,2,normal,normal,notpresent,notpresent,172,82,13.5,145,6.3,8.8,3
1,?,?,yes,yes,no,good,yes,yes,ckd
61,70,?,?,?,?,?,notpresent,notpresent,100,28,2.1,?,?,12.6,43,?,?,yes,yes,no,g
ood,no,no,ckd
60,70,1.010,1,0,?,normal,notpresent,notpresent,109,96,3.9,135,4.0,13.8,41,?,?
,yes,no,no,good,no,no,ckd
50,70,1.010,0,0,?,normal,notpresent,notpresent,230,50,2.2,?,?,12,41,10400,4.6
,yes,yes,no,good,no,no,ckd
67,80,?,?,?,?,?,notpresent,notpresent,341,37,1.5,?,?,12.3,41,6900,4.9,yes,yes
,no,good,no,yes,ckd
19,70,1.020,0,0,?,normal,notpresent,notpresent,?,?,?,?,?,11.5,?,6900,?,no,no,
no,good,no,no,ckd
59,100,1.015,4,2,normal,normal,notpresent,notpresent,255,132,12.8,135,5.7,7.3
,20,9800,3.9,yes,yes,yes,good,no,yes,ckd
54,120,1.015,0,0,?,normal,notpresent,notpresent,103,18,1.2,?,?,?,?,?,?,no,no,
no,good,no,no,ckd
40,70,1.015,3,4,normal,normal,notpresent,notpresent,253,150,11.9,132,5.6,10.9
,31,8800,3.4,yes,yes,no,poor,yes,no,ckd
55,80,1.010,3,1,normal,abnormal,present,present,214,73,3.9,137,4.9,10.9,34,74
00,3.7,yes,yes,no,good,yes,no,ckd
68,80,1.015,0,0,?,abnormal,notpresent,notpresent,171,30,1.0,?,?,13.7,
        43,4900,5.2,no,yes,no,good,no,no,ckd
2,?,1.010,3,0,normal,abnormal,notpresent,notpresent,?,?,?,?,?,?,?,?,?,no,no,n
o,good,yes,no,ckd
64,70,1.010,0,0,?,normal,notpresent,notpresent,107,15,?,?,?,12.8,38,?,?,no,no
,no,good,no,no,ckd
63,100,1.010,1,0,?,normal,notpresent,notpresent,78,61,1.8,141,4.4,12.2,36,105
00,4.3,no,yes,no,good,no,no,ckd
33,90,1.015,0,0,?,normal,notpresent,notpresent,92,19,0.8,?,?,11.8,34,7000,?,n
o,no,no,good,no,no,ckd
68,90,1.010,0,0,?,normal,notpresent,notpresent,238,57,2.5,?,?,9.8,28,8000,3.3
,yes,yes,no,poor,no,no,ckd
36,80,1.010,0,0,?,normal,notpresent,notpresent,103,?,?,?,?,11.9,36,8800,?,no,
no,no,good,no,no,ckd
66,70,1.020,1,0,normal,?,notpresent,notpresent,248,30,1.7,138,5.3,?,?,?,?,yes
,yes,no,good,no,no,ckd
74,60,?,?,?,?,?,notpresent,notpresent,108,68,1.8,?,?,?,?,?,?,yes,yes,no,good,
no,no,ckd
71,90,1.010,0,3,?,normal,notpresent,notpresent,303,30,1.3,136,4.1,13.0,38,920
0,4.6,yes,yes,no,good,no,no,ckd
```

```
34,60,1.020,0,0,?,normal,notpresent,notpresent,117,28,2.2,138,3.8,?,?,?,?,no,
no,no,good,yes,no,ckd
60,90,1.010,3,5,abnormal,normal,notpresent,present,490,95,2.7,131,3.8,11.5,35
,12000,4.5,yes,yes,no,good,no,no,ckd
64,100,1.015,4,2,abnormal,abnormal,notpresent,present,163,54,7.2,140,4.6,7.9,
26,7500,3.4,yes,yes,no,good,yes,no,ckd
57,80,1.015,0,0,?,normal,notpresent,notpresent,120,48,1.6,?,?,11.3,36,7200,3.
8,yes,yes,no,good,no,no,ckd
60,70,?,?,?,?,?,notpresent,notpresent,124,52,2.5,?,?,?,?,?,?,yes,no,no,good,n
o,no,ckd
59,50,1.010,3,0,normal,abnormal,notpresent,notpresent,241,191,12.0,114,2.9,9.
6,31,15700,3.8,no,yes,no,good,yes,no,ckd
65,60,1.010,2,0,normal,abnormal,present,notpresent,192,17,1.7,130,4.3,?,?,950
0,?,yes,yes,no,poor,no,no,ckd
60,90,?,?,?,?,?,notpresent,notpresent,269,51,2.8,138,3.7,11.5,35,?,?,yes,yes,
yes,good,yes,no,ckd
50,90,1.015,1,0,abnormal,abnormal,notpresent,notpresent,?,?,?,?,?,?,?,?,?,no,
no,no,good,yes,no,ckd
51,100,1.015,2,0,normal,normal,notpresent,present,93,20,1.6,146,4.5,?,?,?,?,n
o,no,no,poor,no,no,ckd
37,100,1.010,0,0,abnormal,normal,notpresent,notpresent,?,19,1.3,?,?,15.0,44,4
100,5.2,yes,no,no,good,no,no,ckd
45,70,1.010,2,0,?,normal,notpresent,notpresent,113,93,2.3,?,?,7.9,26,5700,?,n
o,no,yes,good,no,yes,ckd
65,80,?,?,?,?,?,notpresent,notpresent,74,66,2.0,136,5.4,9.1,25,?,?,yes,yes,ye
s,good,yes,no,ckd
80,70,1.015,2,2,?,normal,notpresent,notpresent,141,53,2.2,?,?,12.7,40,9600,?,
yes,yes,no,poor,yes,no,ckd
72,100,?,?,?,?,?,notpresent,notpresent,201,241,13.4,127,4.8,9.4,28,?,?,yes,ye
s,no,good,no,yes,ckd
34,90,1.015,2,0,normal,normal,notpresent,notpresent,104,50,1.6,137,4.1,11.9,3
9,?,?,no,no,no,good,no,no,ckd
65,70,1.015,1,0,?,normal,notpresent,notpresent,203,46,1.4,?,?,11.4,36,5000,4.
1,yes,yes,no,poor,yes,no,ckd
57,70,1.015,1,0,?,abnormal,notpresent,notpresent,165,45,1.5,140,3.3,10.4,31,4
200,3.9,no,no,no,good,no,no,ckd
69,70,1.010,4,3,normal,abnormal,present,present,214,96,6.3,120,3.9,9.4,28,115
00,3.3,yes,yes,yes,good,yes,yes,ckd
62,90,1.020,2,1,?,normal,notpresent,notpresent,169,48,2.4,138,2.9,13.4,47,110
00,6.1,yes,no,no,good,no,no,ckd
64,90,1.015,3,2,?,abnormal,present,notpresent,463,64,2.8,135,4.1,12.2,40,9800
,4.6,yes,yes,no,good,no,yes,ckd
48,100,?,?,?,?,?,notpresent,notpresent,103,79,5.3,135,6.3,6.3,19,7200,2.6,yes
,no,yes,poor,no,no,ckd
48,110,1.015,3,0,abnormal,normal,present,notpresent,106,215,15.2,120,5.7,8.6,
26,5000,2.5,yes,no,yes,good,no,yes,ckd
54,90,1.025,1,0,normal,abnormal,notpresent,notpresent,150,18,1.2,140,4.2,?,?,
?,?,no,no,no,poor,yes,yes,ckd
59,70,1.010,1,3,abnormal,abnormal,notpresent,notpresent,424,55,1.7,138,4.5,12
.6,37,10200,4.1,yes,yes,yes,good,no,no,ckd
56,90,1.010,4,1,normal,abnormal,present,notpresent,176,309,13.3,124,6.5,3.1,9
,5400,2.1,yes,yes,no,poor,yes,yes,ckd
40,80,1.025,0,0,normal,normal,notpresent,notpresent,140,10,1.2,135,5.0,15.0,4
8,10400,4.5,no,no,no,good,no,no,notckd
23,80,1.025,0,0,normal,normal,notpresent,notpresent,70,36,1.0,150,4.6,17.0,52
,9800,5.0,no,no,no,good,no,no,notckd
45,80,1.025,0,0,normal,normal,notpresent,notpresent,82,49,0.6,147,4.4,15.9,46
,9100,4.7,no,no,no,good,no,no,notckd
57,80,1.025,0,0,normal,normal,notpresent,notpresent,119,17,1.2,135,4.7,15.4,4
2,6200,6.2,no,no,no,good,no,no,notckd
```

```
51,60,1.025,0,0,normal,normal,notpresent,notpresent,99,38,0.8,135,3.7,13.0,49
,8300,5.2,no,no,no,good,no,no,notckd
34,80,1.025,0,0,normal,normal,notpresent,notpresent,121,27,1.2,144,3.9,13.6,5
2,9200,6.3,no,no,no,good,no,no,notckd
60,80,1.025,0,0,normal,normal,notpresent,notpresent,131,10,0.5,146,5.0,14.5,4
1,10700,5.1,no,no,no,good,no,no,notckd
38,60,1.020,0,0,normal,normal,notpresent,notpresent,91,36,0.7,135,3.7,14.0,46
,9100,5.8,no,no,no,good,no,no,notckd
42,80,1.020,0,0,normal,normal,notpresent,notpresent,98,20,0.5,140,3.5,13.9,44
,8400,5.5,no,no,no,good,no,no,notckd
35,80,1.020,0,0,normal,normal,notpresent,notpresent,104,31,1.2,135,5.0,16.1,4
5,4300,5.2,no,no,no,good,no,no,notckd
30,80,1.020,0,0,normal,normal,notpresent,notpresent,131,38,1.0,147,3.8,14.1,4
5,9400,5.3,no,no,no,good,no,no,notckd
49,80,1.020,0,0,normal,normal,notpresent,notpresent,122,32,1.2,139,3.9,17.0,4
1,5600,4.9,no,no,no,good,no,no,notckd
55,80,1.020,0,0,normal,normal,notpresent,notpresent,118,18,0.9,135,3.6,15.5,4
3,7200,5.4,no,no,no,good,no,no,notckd
45,80,1.020,0,0,normal,normal,notpresent,notpresent,117,46,1.2,137,5.0,16.2,4
5,8600,5.2,no,no,no,good,no,no,notckd
42,80,1.020,0,0,normal,normal,notpresent,notpresent,132,24,0.7,140,4.1,14.4,5
0,5000,4.5,no,no,no,good,no,no,notckd
50,80,1.020,0,0,normal,normal,notpresent,notpresent,97,40,0.6,150,4.5,14.2,48
,10500,5.0,no,no,no,good,no,no,notckd
55,80,1.020,0,0,normal,normal,notpresent,notpresent,133,17,1.2,135,4.8,13.2,4
1,6800,5.3,no,no,no,good,no,no,notckd
48,80,1.025,0,0,normal,normal,notpresent,notpresent,122,33,0.9,146,3.9,13.9,4
8,9500,4.8,no,no,no,good,no,no,notckd
?,80,?,?,?,?,?,notpresent,notpresent,100,49,1.0,140,5.0,16.3,53,8500,4.9,no,n
o,no,good,no,no,notckd
25,80,1.025,0,0,normal,normal,notpresent,notpresent,121,19,1.2,142,4.9,15.0,4
8,6900,5.3,no,no,no,good,no,no,notckd
23,80,1.025,0,0,normal,normal,notpresent,notpresent,111,34,1.1,145,4.0,14.3,4
1,7200,5.0,no,no,no,good,no,no,notckd
30,80,1.025,0,0,normal,normal,notpresent,notpresent,96,25,0.5,144,4.8,13.8,42
,9000,4.5,no,no,no,good,no,no,notckd
56,80,1.025,0,0,normal,normal,notpresent,notpresent,139,15,1.2,135,5.0,14.8,4
2,5600,5.5,no,no,no,good,no,no,notckd
47,80,1.020,0,0,normal,normal,notpresent,notpresent,95,35,0.9,140,4.1,?,?,?,?
,no,no,no,good,no,no,notckd
19,80,1.020,0,0,normal,normal,notpresent,notpresent,107,23,0.7,141,4.2,14.4,4
4,?,?,no,no,no,good,no,no,notckd
52,80,1.020,0,0,normal,normal,notpresent,notpresent,125,22,1.2,139,4.6,16.5,4
3,4700,4.6,no,no,no,good,no,no,notckd
20,60,1.025,0,0,normal,normal,notpresent,notpresent,?,?,?,137,4.7,14.0,41,450
0,5.5,no,no,no,good,no,no,notckd
46,60,1.025,0,0,normal,normal,notpresent,notpresent,123,46,1.0,135,5.0,15.7,5
0,6300,4.8,no,no,no,good,no,no,notckd
48,60,1.020,0,0,normal,normal,notpresent,notpresent,112,44,1.2,142,4.9,14.5,4
4,9400,6.4,no,no,no,good,no,no,notckd
24,70,1.025,0,0,normal,normal,notpresent,notpresent,140,23,0.6,140,4.7,16.3,4
8,5800,5.6,no,no,no,good,no,no,notckd
47,80,?,?,?,?,?,notpresent,notpresent,93,33,0.9,144,4.5,13.3,52,8100,5.2,no,n
o,no,good,no,no,notckd
55,80,1.025,0,0,normal,normal,notpresent,notpresent,130,50,1.2,147,5,15.5,41,
9100,6.0,no,no,no,good,no,no,notckd
20,70,1.020,0,0,normal,normal,notpresent,notpresent,123,44,1.0,135,3.8,14.6,4
4,5500,4.8,no,no,no,good,no,no,notckd
60,70,1.020,0,0,normal,normal,notpresent,notpresent,?,?,?,?,?,16.4,43,10800,5
.7,no,no,no,good,no,no,notckd
```

```
33,80,1.025,0,0,normal,normal,notpresent,notpresent,100,37,1.2,142,4.0,16.9,5
2,6700,6.0,no,no,no,good,no,no,notckd
66,70,1.020,0,0,normal,normal,notpresent,notpresent,94,19,0.7,135,3.9,16.0,41
,5300,5.9,no,no,no,good,no,no,notckd
71,70,1.020,0,0,normal,normal,notpresent,notpresent,81,18,0.8,145,5.0,14.7,44
,9800,6.0,no,no,no,good,no,no,notckd
39,70,1.025,0,0,normal,normal,notpresent,notpresent,124,22,0.6,137,3.8,13.4,4
3,?,?,no,no,no,good,no,no,notckd
56,70,1.025,0,0,normal,normal,notpresent,notpresent,70,46,1.2,135,4.9,15.9,50
,11000,5.1,?,?,?,good,no,no,notckd
42,70,1.020,0,0,normal,normal,notpresent,notpresent,93,32,0.9,143,4.7,16.6,43
,7100,5.3,no,no,no,good,no,no,notckd
54,70,1.020,0,0,?,?,?,?,76,28,0.6,146,3.5,14.8,52,8400,5.9,no,no,no,good,no,n
o,notckd
47,80,1.025,0,0,normal,normal,notpresent,notpresent,124,44,1.0,140,4.9,14.9,4
1,7000,5.7,no,no,no,good,no,no,notckd
30,80,1.020,0,0,normal,normal,notpresent,notpresent,89,42,0.5,139,5.0,16.7,52
,10200,5.0,no,no,no,good,no,no,notckd
50,?,1.020,0,0,normal,normal,notpresent,notpresent,92,19,1.2,150,4.8,14.9,48,
4700,5.4,no,no,no,good,no,no,notckd
75,60,1.020,0,0,normal,normal,notpresent,notpresent,110,50,0.7,135,5.0,14.3,4
0,8300,5.8,no,no,no,?,?,?,notckd
44,70,?,?,?,?,?,notpresent,notpresent,106,25,0.9,150,3.6,15.0,50,9600,6.5,no,
no,no,good,no,no,notckd
41,70,1.020,0,0,normal,normal,notpresent,notpresent,125,38,0.6,140,5.0,16.8,4
1,6300,5.9,no,no,no,good,no,no,notckd
53,60,1.025,0,0,normal,normal,notpresent,notpresent,116,26,1.0,146,4.9,15.8,4
5,7700,5.2,?,?,?,good,no,no,notckd
34,60,1.020,0,0,normal,normal,notpresent,notpresent,91,49,1.2,135,4.5,13.5,48
,8600,4.9,no,no,no,good,no,no,notckd
73,60,1.020,0,0,normal,normal,notpresent,notpresent,127,48,0.5,150,3.5,15.1,5
2,11000,4.7,no,no,no,good,no,no,notckd
45,60,1.020,0,0,normal,normal,?,?,114,26,0.7,141,4.2,15.0,43,9200,5.8,no,no,n
o,good,no,no,notckd
44,60,1.025,0,0,normal,normal,notpresent,notpresent,96,33,0.9,147,4.5,16.9,41
,7200,5.0,no,no,no,good,no,no,notckd
29,70,1.020,0,0,normal,normal,notpresent,notpresent,127,44,1.2,145,5.0,14.8,4
8,?,?,no,no,no,good,no,no,notckd
55,70,1.020,0,0,normal,normal,notpresent,notpresent,107,26,1.1,?,?,17.0,50,67
00,6.1,no,no,no,good,no,no,notckd
33,80,1.025,0,0,normal,normal,notpresent,notpresent,128,38,0.6,135,3.9,13.1,4
5,6200,4.5,no,no,no,good,no,no,notckd
41,80,1.020,0,0,normal,normal,notpresent,notpresent,122,25,0.8,138,5.0,17.1,4
1,9100,5.2,no,no,no,good,no,no,notckd
52,80,1.020,0,0,normal,normal,notpresent,notpresent,128,30,1.2,140,4.5,15.2,5
2,4300,5.7,no,no,no,good,no,no,notckd
47,60,1.020,0,0,normal,normal,notpresent,notpresent,137,17,0.5,150,3.5,13.6,4
4,7900,4.5,no,no,no,good,no,no,notckd
43,80,1.025,0,0,normal,normal,notpresent,notpresent,81,46,0.6,135,4.9,13.9,48
,6900,4.9,no,no,no,good,no,no,notckd
51,60,1.020,0,0,?,?,notpresent,notpresent,129,25,1.2,139,5.0,17.2,40,8100,5.9
,no,no,no,good,no,no,notckd
46,60,1.020,0,0,normal,normal,notpresent,notpresent,102,27,0.7,142,4.9,13.2,4
4,11000,5.4,no,no,no,good,no,no,notckd
56,60,1.025,0,0,normal,normal,notpresent,notpresent,132,18,1.1,147,4.7,13.7,4
5,7500,5.6,no,no,no,good,no,no,notckd
80,70,1.020,0,0,normal,normal,notpresent,notpresent,?,?,?,135,4.1,15.3,48,630
0,6.1,no,no,no,good,no,no,notckd
55,80,1.020,0,0,normal,normal,notpresent,notpresent,104,28,0.9,142,4.8,17.3,5
2,8200,4.8,no,no,no,good,no,no,notckd
```

```
39,70,1.025,0,0,normal,normal,notpresent,notpresent,131,46,0.6,145,5.0,15.6,4
1,9400,4.7,no,no,no,good,no,no,notckd
44,70,1.025,0,0,normal,normal,notpresent,notpresent,?,?,?,?,?,13.8,48,7800,4.
4,no,no,no,good,no,no,notckd
35,?,1.020,0,0,normal,normal,?,?,99,30,0.5,135,4.9,15.4,48,5000,5.2,no,no,no,
good,no,no,notckd
58,70,1.020,0,0,normal,normal,notpresent,notpresent,102,48,1.2,139,4.3,15.0,4
0,8100,4.9,no,no,no,good,no,no,notckd
61,70,1.025,0,0,normal,normal,notpresent,notpresent,120,29,0.7,137,3.5,17.4,5
2,7000,5.3,no,no,no,good,no,no,notckd
30,60,1.020,0,0,normal,normal,notpresent,notpresent,138,15,1.1,135,4.4,?,?,?,
?,no,no,no,good,no,no,notckd
57,60,1.020,0,0,normal,normal,notpresent,notpresent,105,49,1.2,150,4.7,15.7,4
4,10400,6.2,no,no,no,good,no,no,notckd
65,60,1.020,0,0,normal,normal,notpresent,notpresent,109,39,1.0,144,3.5,13.9,4
8,9600,4.8,no,no,no,good,no,no,notckd
70,60,?,?,?,?,?,notpresent,notpresent,120,40,0.5,140,4.6,16.0,43,4500,4.9,no,
no,no,good,no,no,notckd
43,80,1.025,0,0,normal,normal,notpresent,notpresent,130,30,1.1,143,5.0,15.9,4
5,7800,4.5,no,no,no,good,no,no,notckd
40,80,1.020,0,0,normal,normal,notpresent,notpresent,119,15,0.7,150,4.9,?,?,?,
?,no,no,no,good,no,no,notckd
58,80,1.020,0,0,normal,normal,notpresent,notpresent,100,50,1.2,140,3.5,14.0,5
0,6700,6.5,no,no,no,good,no,no,notckd
47,60,1.020,0,0,normal,normal,notpresent,notpresent,109,25,1.1,141,4.7,15.8,4
1,8300,5.2,no,no,no,good,no,no,notckd
30,60,1.025,0,0,normal,normal,notpresent,notpresent,120,31,0.8,150,4.6,13.4,4
4,10700,5.8,no,no,no,good,no,no,notckd
28,70,1.020,0,0,normal,normal,?,?,131,29,0.6,145,4.9,?,45,8600,6.5,no,no,no,g
ood,no,no,notckd
33,60,1.025,0,0,normal,normal,notpresent,notpresent,80,25,0.9,146,3.5,14.1,48
,7800,5.1,no,no,no,good,no,no,notckd
43,80,1.020,0,0,normal,normal,notpresent,notpresent,114,32,1.1,135,3.9,?,42,?
,?,no,no,no,good,no,no,notckd
59,70,1.025,0,0,normal,normal,notpresent,notpresent,130,39,0.7,147,4.7,13.5,4
6,6700,4.5,no,no,no,good,no,no,notckd
34,70,1.025,0,0,normal,normal,notpresent,notpresent,?,33,1,150,5.0,15.3,44,10
500,6.1,no,no,no,good,no,no,notckd
23,80,1.020,0,0,normal,normal,notpresent,notpresent,99,46,1.2,142,4.0,17.7,46
,4300,5.5,no,no,no,good,no,no,notckd
24,80,1.025,0,0,normal,normal,notpresent,notpresent,125,?,?,136,3.5,15.4,43,5
600,4.5,no,no,no,good,no,no,notckd
60,60,1.020,0,0,normal,normal,notpresent,notpresent,134,45,0.5,139,4.8,14.2,4
8,10700,5.6,no,no,no,good,no,no,notckd
25,60,1.020,0,0,normal,normal,notpresent,notpresent,119,27,0.5,?,?,15.2,40,92
00,5.2,no,no,no,good,no,no,notckd
44,70,1.025,0,0,normal,normal,notpresent,notpresent,92,40,0.9,141,4.9,14.0,52
,7500,6.2,no,no,no,good,no,no,notckd
62,80,1.020,0,0,normal,normal,notpresent,notpresent,132,34,0.8,147,3.5,17.8,4
4,4700,4.5,no,no,no,good,no,no,notckd
25,70,1.020,0,0,normal,normal,notpresent,notpresent,88,42,0.5,136,3.5,13.3,48
,7000,4.9,no,no,no,good,no,no,notckd
32,70,1.025,0,0,normal,normal,notpresent,notpresent,100,29,1.1,142,4.5,14.3,4
3,6700,5.9,no,no,no,good,no,no,notckd
63,70,1.025,0,0,normal,normal,notpresent,notpresent,130,37,0.9,150,5.0,13.4,4
1,7300,4.7,no,no,no,good,no,no,notckd
44,60,1.020,0,0,normal,normal,notpresent,notpresent,95,46,0.5,138,4.2,15.0,50
,7700,6.3,no,no,no,good,no,no,notckd
37,60,1.025,0,0,normal,normal,notpresent,notpresent,111,35,0.8,135,4.1,16.2,5
0,5500,5.7,no,no,no,good,no,no,notckd
```

```
64,60,1.020,0,0,normal,normal,notpresent,notpresent,106,27,0.7,150,3.3,14.4,4
2,8100,4.7,no,no,no,good,no,no,notckd
22,60,1.025,0,0,normal,normal,notpresent,notpresent,97,18,1.2,138,4.3,13.5,42
,7900,6.4,no,no,no,good,no,no,notckd
33,60,?,?,?,normal,normal,notpresent,notpresent,130,41,0.9,141,4.4,15.5,52,43
00,5.8,no,no,no,good,no,no,notckd
43,60,1.025,0,0,normal,normal,notpresent,notpresent,108,25,1.0,144,5.0,17.8,4
3,7200,5.5,no,no,no,good,no,no,notckd
38,80,1.020,0,0,normal,normal,notpresent,notpresent,99,19,0.5,147,3.5,13.6,44
,7300,6.4,no,no,no,good,no,no,notckd
35,70,1.025,0,0,?,?,notpresent,notpresent,82,36,1.1,150,3.5,14.5,52,9400,6.1,
no,no,no,good,no,no,notckd
65,70,1.025,0,0,?,?,notpresent,notpresent,85,20,1.0,142,4.8,16.1,43,9600,4.5,
no,no,no,good,no,no,notckd
29,80,1.020,0,0,normal,normal,notpresent,notpresent,83,49,0.9,139,3.3,17.5,40
,9900,4.7,no,no,no,good,no,no,notckd
37,60,1.020,0,0,normal,normal,notpresent,notpresent,109,47,1.1,141,4.9,15.0,4
8,7000,5.2,no,no,no,good,no,no,notckd
39,60,1.020,0,0,normal,normal,notpresent,notpresent,86,37,0.6,150,5.0,13.6,51
,5800,4.5,no,no,no,good,no,no,notckd
32,60,1.025,0,0,normal,normal,notpresent,notpresent,102,17,0.4,147,4.7,14.6,4
1,6800,5.1,no,no,no,good,no,no,notckd
23,60,1.020,0,0,normal,normal,notpresent,notpresent,95,24,0.8,145,5.0,15,52,6
300,4.6,no,no,no,good,no,no,notckd
34,70,1.025,0,0,normal,normal,notpresent,notpresent,87,38,0.5,144,4.8,17.1,47
,7400,6.1,no,no,no,good,no,no,notckd
66,70,1.025,0,0,normal,normal,notpresent,notpresent,107,16,1.1,140,3.6,13.6,4
2,11000,4.9,no,no,no,good,no,no,notckd
47,60,1.020,0,0,normal,normal,notpresent,notpresent,117,22,1.2,138,3.5,13,45,
5200,5.6,no,no,no,good,no,no,notckd
74,60,1.020,0,0,normal,normal,notpresent,notpresent,88,50,0.6,147,3.7,17.2,53
,6000,4.5,no,no,no,good,no,no,notckd
35,60,1.025,0,0,normal,normal,notpresent,notpresent,105,39,0.5,135,3.9,14.7,4
3,5800,6.2,no,no,no,good,no,no,notckd
29,80,1.020,0,0,normal,normal,notpresent,notpresent,70,16,0.7,138,3.5,13.7,54
,5400,5.8,no,no,no,good,no,no,notckd
33,80,1.025,0,0,normal,normal,notpresent,notpresent,89,19,1.1,144,5.0,15,40,1
0300,4.8,no,no,no,good,no,no,notckd
67,80,1.025,0,0,normal,normal,notpresent,notpresent,99,40,0.5,?,?,17.8,44,590
0,5.2,no,no,no,good,no,no,notckd
73,80,1.025,0,0,normal,normal,notpresent,notpresent,118,44,0.7,137,3.5,14.8,4
5,9300,4.7,no,no,no,good,no,no,notckd
24,80,1.020,0,0,normal,normal,notpresent,notpresent,93,46,1.0,145,3.5,?,?,107
00,6.3,no,no,no,good,no,no,notckd
60,80,1.025,0,0,normal,normal,notpresent,notpresent,81,15,0.5,141,3.6,15,46,1
0500,5.3,no,no,no,good,no,no,notckd
68,60,1.025,0,0,normal,normal,notpresent,notpresent,125,41,1.1,139,3.8,17.4,5
0,6700,6.1,no,no,no,good,no,no,notckd
30,80,1.025,0,0,normal,normal,notpresent,notpresent,82,42,0.7,146,5.0,14.9,45
,9400,5.9,no,no,no,good,no,no,notckd
75,70,1.020,0,0,normal,normal,notpresent,notpresent,107,48,0.8,144,3.5,13.6,4
6,10300,4.8,no,,no,no,good,no,no,notckd
69,70,1.020,0,0,normal,normal,notpresent,notpresent,83,42,1.2,139,3.7,16.2,50
,9300,5.4,no,no,no,good,no,no,notckd
28,60,1.025,0,0,normal,normal,notpresent,notpresent,79,50,0.5,145,5.0,17.6,51
,6500,5.0,no,no,no,good,no,no,notckd
72,60,1.020,0,0,normal,normal,notpresent,notpresent,109,26,0.9,150,4.9,15,52,
10500,5.5,no,no,no,good,no,no,notckd
61,70,1.025,0,0,normal,normal,notpresent,notpresent,133,38,1.0,142,3.6,13.7,4
7,9200,4.9,no,no,no,good,no,no,notckd
```

```
79,80,1.025,0,0,normal,normal,notpresent,notpresent,111,44,1.2,146,3.6,16.3,4
0,8000,6.4,no,no,no,good,no,no,notckd
70,80,1.020,0,0,normal,normal,notpresent,notpresent,74,41,0.5,143,4.5,15.1,48
,9700,5.6,no,no,no,good,no,no,notckd
58,70,1.025,0,0,normal,normal,notpresent,notpresent,88,16,1.1,147,3.5,16.4,53
,9100,5.2,no,no,no,good,no,no,notckd
64,70,1.020,0,0,normal,normal,notpresent,notpresent,97,27,0.7,145,4.8,13.8,49
,6400,4.8,no,no,no,good,no,no,notckd
71,60,1.025,0,0,normal,normal,notpresent,notpresent,?,?,0.9,140,4.8,15.2,42,7
700,5.5,no,no,no,good,no,no,notckd
62,80,1.025,0,0,normal,normal,notpresent,notpresent,78,45,0.6,138,3.5,16.1,50
,5400,5.7,no,no,no,good,no,no,notckd
59,60,1.020,0,0,normal,normal,notpresent,notpresent,113,23,1.1,139,3.5,15.3,5
4,6500,4.9,no,no,no,good,no,no,notckd
71,70,1.025,0,0,?,?,notpresent,notpresent,79,47,0.5,142,4.8,16.6,40,5800,5.9,
no,no,no,good,no,no,notckd
48,80,1.025,0,0,normal,normal,notpresent,notpresent,75,22,0.8,137,5.0,16.8,51
,6000,6.5,no,no,no,good,no,no,notckd
80,80,1.025,0,0,normal,normal,notpresent,notpresent,119,46,0.7,141,4.9,13.9,4
9,5100,5.0,no,no,no,good,no,no,notckd
57,60,1.020,0,0,normal,normal,notpresent,notpresent,132,18,1.1,150,4.7,15.4,4
2,11000,4.5,no,no,no,good,no,no,notckd
63,70,1.020,0,0,normal,normal,notpresent,notpresent,113,25,0.6,146,4.9,16.5,5
2,8000,5.1,no,no,no,good,no,no,notckd
46,70,1.025,0,0,normal,normal,notpresent,notpresent,100,47,0.5,142,3.5,16.4,4
3,5700,6.5,no,no,no,good,no,no,notckd
15,80,1.025,0,0,normal,normal,notpresent,notpresent,93,17,0.9,136,3.9,16.7,50
,6200,5.2,no,no,no,good,no,no,notckd
51,80,1.020,0,0,normal,normal,notpresent,notpresent,94,15,1.2,144,3.7,15.5,46
,9500,6.4,no,no,no,good,no,no,notckd
41,80,1.025,0,0,normal,normal,notpresent,notpresent,112,48,0.7,140,5.0,17.0,5
2,7200,5.8,no,no,no,good,no,no,notckd
52,80,1.025,0,0,normal,normal,notpresent,notpresent,99,25,0.8,135,3.7,15.0,52
,6300,5.3,no,no,no,good,no,no,notckd
36,80,1.025,0,0,normal,normal,notpresent,notpresent,85,16,1.1,142,4.1,15.6,44
,5800,6.3,no,no,no,good,no,no,notckd
57,80,1.020,0,0,normal,normal,notpresent,notpresent,133,48,1.2,147,4.3,14.8,4
6,6600,5.5,no,no,no,good,no,no,notckd
43,60,1.025,0,0,normal,normal,notpresent,notpresent,117,45,0.7,141,4.4,13.0,5
4,7400,5.4,no,no,no,good,no,no,notckd
50,80,1.020,0,0,normal,normal,notpresent,notpresent,137,46,0.8,139,5.0,14.1,4
5,9500,4.6,no,no,no,good,no,no,notckd
55,80,1.020,0,0,normal,normal,notpresent,notpresent,140,49,0.5,150,4.9,15.7,4
7,6700,4.9,no,no,no,good,no,no,notckd
42,70,1.025,0,0,normal,normal,notpresent,notpresent,75,31,1.2,141,3.5,16.5,54
,7800,6.2,no,no,no,good,no,no,notckd
12,80,1.020,0,0,normal,normal,notpresent,notpresent,100,26,0.6,137,4.4,15.8,4
9,6600,5.4,no,no,no,good,no,no,notckd
17,60,1.025,0,0,normal,normal,notpresent,notpresent,114,50,1.0,135,4.9,14.2,5
1,7200,5.9,no,no,no,good,no,no,notckd
58,80,1.025,0,0,normal,normal,notpresent,notpresent,131,18,1.1,141,3.5,15.8,5
3,6800,6.1,no,no,no,good,no,no,notckd
```

Lampiran 13. Surat Keputusan Penetapan Dosen Pembimbing

UNNES

KEPUTUSAN
DEKAN FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
Nomor: 992/UN37.1.4/PT/2020
Tentang
PENETAPAN DOSEN PEMBIMBING SKRIPSI/TUGAS AKHIR SEMESTER GASAL/GENAP
TAHUN AKADEMIK 2019/2020

Menimbang : Bahwa untuk memperlancar mahasiswa Jurusan/Prodi Ilmu Komputer/Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam membuat Skripsi/Tugas Akhir, maka perlu menetapkan Dosen-dosen Jurusan/Prodi Ilmu Komputer/Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam UNNES untuk menjadi pembimbing.

Mengingat :
1. Undang-undang No.20 Tahun 2003 tentang Sistem Pendidikan Nasional (Tambahan Lembaran Negara RI No.4301, penjelasan atas Lembaran Negara RI Tahun 2003, Nomor 78)
2. Peraturan Rektor No. 21 Tahun 2011 tentang Sistem Informasi Skripsi UNNES
3. SK. Rektor UNNES No. 164/O/2004 tentang Pedoman penyusunan Skripsi/Tugas Akhir Mahasiswa Strata Satu (S1) UNNES;
4. SK Rektor UNNES No.162/O/2004 tentang penyelenggaraan Pendidikan UNNES;

Menimbang : Usulan Ketua Jurusan/Prodi Ilmu Komputer/Teknik Informatika Tanggal 15 Oktober 2019

MEMUTUSKAN

Menetapkan :

PERTAMA : Menunjuk dan menugaskan kepada:

Nama	Budi Prasetiyo, S. Si., M. Kom
NIP	198805012014041001
Pangkat/Golongan	Penata Muda Tk. I - III/b
Jabatan Akademik	Asisten Ahli

Sebagai Pembimbing

Untuk membimbing mahasiswa penyusun skripsi/Tugas Akhir:

Nama	Hafid Alpin Al Gazni
NIM	4811415016
Jurusan/Prodi	Ilmu Komputer/Teknik Informatika
Topik	DATA MINING

KEDUA : Keputusan ini mulai berlaku sejak tanggal ditetapkan.

DITETAPKAN DI SEMARANG
PADA TANGGAL 20 Januari 2020
DEKAN

Dr. Sugianto, M.Si.
NIP 196102151993031001

Tembusan
1. Wakil Dekan Bidang Akademik
2. Ketua Jurusan
3. Petinggal

4811415016
D-AKD-24/Rev. 00