PENINGKATAN AKURASI ALGORITMA *K-NEAREST NEIGHBOR* MENGGUNAKAN *NORMALISASI Z-SCORE* DAN *PARTICLE SWARM OPTIMIZATION* UNTUK PREDIKSI *CUSTOMER CHURN*

Skripsi

disusun sebagai salah satu syarat

untuk memperoleh gelar Sarjana Komputer

Program Studi Teknik Informatika

oleh

Muhammad Ali Imron
4611415034

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2020

PERNYATAAN

Saya menyatakan dengan sebenar-benarnya bahwa skripsi saya yang berjudul "Peningkatan Akurasi Algoritma *K-Nearest Neighbor* (KNN) Menggunakan Normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk Prediksi *Customer Churn*" disusun atas dasar 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 di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi sesuai ketentuan peraturan perundang-undangan.

Semarang, 20 Maret 2020

Muhammad Ali Imron

4611415034

PERSETUJUAN PEMBIMBING

Nama : Muhammad Ali Imron

NIM : 4611415034

Program Studi : S-1 Teknik Informatika

Judul Skripsi : Peningkatan Akurasi Algoritma *K-Nearest Neighbor* (KNN) Menggunakan Normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk Prediksi *Customer Churn*

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

Semarang, 20 Maret 2020

Pembimbing

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

NIP. 198805012014041001

PENGESAHAN

Skripsi yang berjudul

Peningkatan Akurasi Algoritma *K-Nearest Neighbor* (KNN) Menggunakan Normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk Prediksi *Customer Churn*

Disusun oleh

Muhammad Ali Imron

4611415034

Telah dipertahankan di hadapan sidang panitia ujian skripsi FMIPA UNNES pada tanggal 20 Maret 2020

Panitia

Ketua

Dr. Sugianto, M.Si.

NIP 196102191993031001

Sekretaris

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

NIP 197405172006041001

Penguji 1

Riza Arifudin S.Pd., M.Cs.

NIP. 198005252005011001

Penguji 2

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

NIP. 197405172006041001

Anggota Penguji

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

NIP. 198805012014041001

MOTTO DAN PERSEMBAHAN

MOTTO

- Setiap orang pasti meiliki jalan hidupnya sendiri, percayalah Allah pasti akan memberikan jalan terbaik.
- Selalu berbuat baiklah pada siapapun, maka kebaikan itu akan kembali kepada diri sendiri.

PERSEMBAHAN

Skripsi ini ku persembahkan kepada:

- Kedua Orang Tua saya Bapak Khusairi dan Ibu Suwati yang selalu memberikan kasih sayang, doa, serta dukungan moril dan materiil.
- Adik saya Dwi Ayu Lestari yang telah memberikan dukungan untuk tetap belajar serta doa yang terus dipanjatkan.
- Teman-teman saya di jurusan Ilmu Komputer, Fakultas MIPA, serta teman-teman di Universitas Negeri Semarang.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu hingga terselesaikannya penulisan skripsi ini.
- Almamaterku, Universitas Negeri Semarang.

PRAKATA

Puji syukur penulis panjatkan kepada Allah *Subhanahu wa ta'ala* atas berkat rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi dalam yang berjudul "Peningkatan Akurasi Algoritma *K-Nearest Neighbor* (KNN) Menggunakan Normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk Prediksi *Customer Churn*".

Penulis menyadari bahwa penulisan skripsi ini tidak akan selesai tanpa adanya dukungan serta bantuan dari berbagai pihak, terutama pada saat masa pandemi COVID-19. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang.
2. Dr. Sugianto, M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. 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. 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. Riza Arifudin S.Pd., M.Cs., Dosen Penguji yang telah meluangkan waktu, membantu, membimbing, mengarahkan dan memberikan saran sehingga penulis dapat menyelesaikan skripsi ini.

6. Bapak dan Ibu Dosen Jurusan Ilmu Komputer yang telah memberikan bekal kepada penulis dalam penyusunan skripsi ini.
7. Kedua Orang Tua saya Bapak Khusairi dan Ibu Suwati yang telah mencurahkan keringatnya untuk membiayai pendidikan saya, yang selalu memberikan kasih sayang, doa, dan dukungannnya.
8. Adik saya saya Dwi Ayu Lestari dan Ayu Fitriani yang telah memberikan dukungan untuk tetap belajar serta doa yang terus dipanjatkan.
9. Keluarga besar kontrakan *Console House* (Akhsin, Alpin, Broto, Fachrizal, Farhan, Raka, Iqbal, Jefri, Khakim, Khamim, Salman, Warson) yang telah saling mendukung dalam menyelesaikan skripsi.
10. Teman-teman KKN (Mufti dan Mekar) yang selalu memberikan semangat.
11. Teman-teman saya di jurusan Ilmu Komputer, Fakultas MIPA, serta teman-teman di UNNES yang telah memberikan semangat dan dukungannya.
12. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu persatu, terimakasih atas bantuannya.

Semoga skripsi ini dapat memberikan manfaat bagi pembaca di masa yang akan datang.

Semarang, 20 Maret 2020

Penulis

Muhammad Ali Imron
4611415034

ABSTRAK

Muhammad Ali Imron 2020. Peningkatan Akurasi Algoritma *K-Nearest Neighbor* (KNN) Menggunakan Normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk Prediksi *Customer Churn* . Skripsi, Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Budi Prasetiyo, S.Si., M.Kom.

Kata Kunci : Algoritma *K-Nearest Neighbor, Z-Score, Particle Swarm Optimization, German Credit Dataset.*

Pesatnya perkembangan teknologi, sistem informasi, dan ilmu pengetahuan mengakibatkan persaingan dalam dunia bisnis menjadi semakin ketat. Dengan meningkatnya persaingan dalam dunia bisnis, banyak perusahaan memanfaatkan teknik *data mining* untuk mengetahui tingkat loyalitas pelanggan. Dalam usaha ini, *data mining* dapat digunakan untuk mengetahui tingkat loyalitas pelanggan. *Data mining* terdiri dari beberapa model penelitian, salah satunya adalah klasifikasi. Salah satu metode paling umum digunakan dalam klasifikasi adalah algoritma *K-Nearest Neighbor*. Dalam penelitian ini, data yang digunakan adalah *German Credit Datasets* yang diperoleh dari UCI *machine learning repository* yang terdiri dari 20 atribut dan 1 atribut *class. Dataset* ini memiliki struktur data yang tidak normal dimana nilai pada atribut tertentu memiliki *range* yang cukup jauh. Untuk mengatasi masalah data tersebut, Maka digunakan metode normalisasi *Z-Score*, dimana metode *Z-Score* digunakan untuk menormalkan stuktur data yang tidak normal menjadi normal. Kemudian untuk meningkatkan kinerja *K-Nearest Neighbor*, penentuan parameter nilai K pada penelitian ini menggunakan *Particle Swarm Optimization* sehingga meningkatkan akurasi pada klasifikasi *K-Nearest Neighbor*. Metode *Particle Swarm Optimization* menghasilkan parameter nilai K=10. Dari hasil penelitian ini, penerapan normalisasi *Z-Score* dan *Particle Swarm Optimization* dengan algoritma *K-Nearest Neighbor* berhasil meningkatkan akurasi sebesar 14%. Dengan akurasi awal 68,5%, setelah melakukan penerapan normalisasi *Z-Score* dan *Particle Swarm Optimization* menjadi 82,5%. Penelitian ini dapat dijadikan sebagai bahan acuan bagi peneliti selanjutnya agar melakukan penelitian pada tahap *preprocessing* dan optimasi karena pada tahap tersebut terbukti dapat memberikan peningkatan akurasi. *Particle Swarm Optimization* dalam penelitian kali ini kurang efisien dalam melakukan pencarian parameter nilai K, sehingga pada penelitian selanjutnya diharapkan dapat mengatasi masalah ini dengan menggunakan metode lain.

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

Lampiran Halaman

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Era data merupakan era dimana data ybab 1ang berkembang secara pesat, tersebar secara luas dan dengan kapasitas yang besar memerlukan metode pengolahan yang tepat serta terorganisasi agar dapat dimanfaatkan secara maksimal (Han, 2012: 1-2). Dari data yang tersedia akan digali informasinya dan diharapkan dapat diterapkan untuk data yang lebih besar yang belum pernah diketahui sebelumnya (Bramer, 2007: 11).

Dalam dunia bisnis, pelanggan adalah aset utama. Oleh karena itu, berbagai cara telah ditempuh perusahaan agar pelanggan tidak berhenti berlangganan (Wardani, 2018: 17). Untuk memperoleh pelanggan baru memerlukan biaya hingga 10 kali lipat lebih mahal dibandingkan biaya untuk mempertahankan pelanggan yang ada. Mahalnya biaya untuk memperoleh pelanggan baru tentunya perusahaan akan lebih memilih mempertahankan pelanggan. Berdasarkan fakta tersebut maka banyak perusahaan sekarang lebih beralih untuk mempertahankan pelanggan dan menghindari *churn* pelanggan (Arifin, 2015 : 1).

Dengan meningkatkan persaingan dan penawaran di pasar industri, banyak perusahaan yang memanfaatkan *data mining* untuk memprediksi (Brandusoiu &

Toderan, 2013: 19). *Data mining* adalah sebuah kegiatan digunakan untuk menemukan pola yang menarik untuk sejumlah besar data (Sugiharti, 2017: 903).

Dalam *data mining* terdapat beberapa tahapan, diantaranya tahapan *pre-processing,* tahap *processing,* dan tahap *post-processing.* Pada tahap *pre-processing* tediri dari beberapa tahapan diantaranya adalah *data cleaning, data integration, data reduction,* dan *data transformation* (Mabrur & Lubis, 2012: 54). Pada tahap *data transformation* terdapat beberapa metode untuk melakukan proses transformasi data diantaranya yaitu *smooting, generalization, normalization, aggregration* dan *attribute construction* (Junaidi *et al.*, 2011: 94). Menurut Junaidi, *et al.* (2011: 96) normalisasi merupakan proses dimana suatu atribut numerik dipetakan atau diskalakan dalam *range* tertentu. Normalisasi data berguna untuk meminimalisasi pembiasan data pada *data mining* karena nilai-nilai artibut dalam data biasanya memiliki rentang yang berbeda-beda (Suyatno, 2017: 111).

Terdapat beberapa teknik normalisasi yang sering di gunakan di antaranya adalah normalisasi *min-max,* normalisasi *Z-Score* dan normalisasi *decimal scaling* yang semuanya memliliki tujuan untuk memetakan data ke dalam skala tertentu (Saranya & Manikanda, 20013: 2701). Normalisasi *Z-Score* adalah normalisasi data yang digunakan untuk memberikan rentang data dengan menggunakan nilai *mean* dan standar deviasi (Goyal *et al.* 2014: 32).

Untuk menyelesaikan permasalahan optimasi, algoritma *Particle Swarm Optimization* (PSO) merupakan salah satu algoritma metaheuristik yang biasa digunakan (Ashari *et al.,* 2016: 149). Dalam beberapa kasus telah terbukti bahwa PSO lebih kompetitif (Muslim *et al.*, 2017: 2). Metode optimasi ini terbukti efektif

dan berhasil digunakan untuk memecahkan masalah optimasi multidimensi dan multiparameter pada pembelajaran *machine learning* seperti *neural network* dan algoritma teknik klasifikasi (Fei, et al., 2009: 1605).

Algoritma K-Nearest Neighbor (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Teknik ini sangat sederhana dan mudah diimplementasikan. Mirip dengan teknik *clustering*, yaitu mengelompokkan suatu data baru berdasarkan jarak data baru itu ke beberapa data/tetangga terdekat. Sebelum mencari jarak data ke tetangga adalah menentukan nilai K tetangga (*neighbor*) (Dzikrullah, 2017: 380).

Berdasarkan uraian permasalahan di atas, maka penelitian akan berfokus untuk meningkatan akurasi dalam klasifikasi *customer churn* dengan algoritma klasifikasi *K-Nearest Neighbor* menggunakan normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) yang berjudul "peningkatan akurasi algoritma *K-Nearest Neighbor* (KNN) menggunakan normalisasi *Z-Score* dan *Particle Swarm Optimization* (PSO) untuk prediksi *customer churn".*

1.2 Rumusan Masalah

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

1. Bagaimana cara kerja Algoritma normalisasi *Z-Score* dan PSO untuk meningkatkan akurasi Algoritma KNN pada prediksi *customer churn*?

2. Bagaimana hasil akurasi dari algoritma KNN setelah di normalisasi menggunakan normalisasi *Z-Score* dan PSO untuk prediksi *customer churn?*

1.3 Batasan Masalah

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

1. Algoritma klasifikasi yang digunakan adalah KNN.
2. Algoritma normalisasi yang digunakan dalam penelitian ini adalah normalisasi *Z-Score*.
3. Algoritma optimasi yang digunakan dalam penelitian ini adalah KNN untuk mencari optimasi nilai K.
4. Data *customer churn* yang digunakan dalam penelitian ini adalah *dataset* dari UCI *Machine Learning Repository,* yaitu *German Credit Data.* (*https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)*

1.4 Tujuan penelitian

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

1. Mengetahui cara kerja normalisasi *Z-Score* dan algoritma PSO untuk meningkatkan akurasi algoritma KNN pada prediksi *customer churn.*

2. Untuk menganalisis hasil dan meningkatlan akurasi algoritma KNN dengan penerapan Algoritma normalisasi *Z-Score* dan Algoritma PSO pada data *customer churn.*

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Menerapkan algoritma *Particle Swarm Optimization* dan normalisasi *Z-Score* untuk optimasi algoritma *K-Nearest Neighbor* dalam memprediksi *customer churn.*
2. Dalam lingkungan akademis diperoleh pengetahuan terhadap akurasi algoritma *K-Nearest Neighbor* yang dinormalisasi dengan algoritma *Z-Score* dan di optimalkan dengan *Particle Swarm Optimization.*

1.6 Sistematika Penulisan Skripsi

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 berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat penelitian serta sistematika penulisan skripsi.

2. BAB 2: TINJAUAN PUSTAKA

   Bab ini berisi penjelasan mengenai definisi maupun pemikiran-pemikiran yang dijadikan kerangka teoritis yang menyangkut dan mendasari pemecahan masalah dalam skripsi ini.

3. BAB 3: METODE PENELITIAN

   Bab ini berisi penjelasan mengenai studi pendahuluan, tahap pengumpulan data, dan tahap pengembangan sistem.

4. BAB 4: HASIL DAN PEMBAHASAN

   Bab ini berisi hasil penelitian berserta pembahasannya.

5. BAB 5: PENUTUP

   Bab ini berisi simpulan dari penulisan skripsi dan saran yang diberikan penulis untuk mengembangkan skripsi ini.

1.6.3 Bagian Akhir Skripsi

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

BAB 2

TINJAUAN PUSTAKA

2.1 Data Mining

Data mining merupakan langkah analisis terhadap proses penemuan pengetahuan di dalam basis data atau *knowledge discovery in database* yang sering dikenal dengan istilah KDD. *Data mining* ditunjukkan untuk mengambil pengetahuan dari sekumpulan data sehingga didapatkan pola yang dapat dimengerti manusia melalui tahapan *data mining* yang meliputi: seleksi, *preprocessing*, transformasi, *data mining*, dan interpretasi evaluasi. Dalam prosesnya *data mining* menggabungkan empat disiplin ilmu komputer, diantaranya *artificial intelligent, machine learning*, *statistics,* dan *database system.* Dari proses *data mining* dapat diperoleh karakteristik data (deskripstif) dan model pengetahuan (prediktif). Namun, secara fungsionalitas tugas *data mining* dapat dikelompokkan menjadi enam, yaitu: klasifikasi, klasterisasi, regresi, deteksi anomali, asosiasi, dan perangkuman (Suyanto, 2017: 1-3).

2.2 Klasifikasi

Salah satu metode *data mining* adalah klasifikasi, yang merupakan suatu teknik penambangan data yang dapat digunakan untuk memprediksi keanggotaan grup ke sejumlah data dengan menemukan model atau fungsi yang menggambarkan konsep atau kelas data (Muslim *et al.*, 2017: 1). Algoritma yang biasa digunakan

dalam klasikasi diantaranya *Decission Tree*, *Support Vector Machine, Naïve Bayes*, *K-Nearest Neighbor,* dll (Muslim *et al.,* 2018: 141).

Menurut Han *et al.* (2012: 328-220), Klasifikasi data juga merupakan sebuah pembelajaran terawasi *(supervised learning),* karena label/target kelas pada data sudah disediakan. Klasifikasi data adalah proses dua langkah, yang terdiri dari langkah pembelajaran (di mana model klasifikasi dibangun) dan langkah klasifikasi (di mana model digunakan untuk memprediksi label kelas untuk data yang diberikan). Langkah pertama dari proses klasifikasi biasanya merupakan pembelajaran pemetaan atau fungsi y = f(x), yang dapat memprediksi label kelas yang terkait dari tupel X yang diberikan untuk memisahkan kelas data. Untuk mengukur keakuratan *classifier* digunakan persentase *tuple set* uji benar dari semua set tes yang diberikan. Jika keakuratan *classifier* dapat diterima, maka *classifier* dapat digunakan untuk mengklasifikasi terhadap data yang label kelasnya tidak diketahui.

Dalam klasifikasi diberikan sejumlah *record* yang dinamakan *training set*, yang terdiri dari beberapa atribut, atribut dapat berupa kontinu ataupun kategoris, salah satu atribut menunjukkan kelas untuk *record*. Fungsi target disebut juga model klasifikasi seperti yang terlihat pada Gambar 2.1.

Gambar 2.1 Blok Diagram Model Klasifikasi

Fungsi target juga dikenal sebagai model klasifikasi. Menurut Hermawati (2013: 56) ada dua jenis model klasifikasi yaitu antara lain sebagai berikut:

1. Pemodelan Deskriptif (*Descriptive Modelling*) adalah model klasifikasi yang dapat berfungsi sebagai suatu alat penjelasan untuk membedakan objek-objek dalam kelas-kelas yang berbeda.
2. Pemodelan Prediktif (*Predictive Modelling*) adalah model klasifikasi yang dapat digunakan untuk memprediksi label kelas *record* yang tidak diketahui.

Teknik klasifikasi (*classifier*) merupakan suatu pendekatan sistematis untuk membangun model klasifikasi dari suatu himpunan data masukan. Tiap teknik menggunakan suatu algoritma pembelajaran (*learning algorithm*) untuk mendapatkan suatu model yang paling memenuhi hubungan antara himpunan atribut dan label kelas dalam data masukan. Tujuan dari algoritma pembelajaran adalah untuk membangun model yang secara umum berkemampuan baik, yaitu model yang dapat memprediksi label kelas dari *record* yang tidak diketahui kelas sebelumnya dengan lebih akurat (Hermawati, 2013: 56).

2.3 *K-Nearest Neighbor* (KNN)

K-Nearest Neighbor (KNN) termasuk kelompok *instance based learning*. Algoritma ini juga merupakan salah satu teknik *lazy learning*. *K-Nearest Neighbor* dilakukan dengan mencari kelompok n objek dalam data *training* yang paling dekat (mirip) dengan objek pada data baru atau data *testing*. Algoritma *K-Nearest Neighbor* adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru dan kasus lama yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang

ada (Kusrini, 2009: 93). *K-Nearest Neighbor* merupakan algoritma klasifikasi yang tetap konsisten dalam sejumlah besar data serta mengklasifikasikan berdasarkan jarak terpendek antara data yang dievaluasi oleh titik terdekat dalam data pelatihan. Algoritma KNN lebih fleksibel karena didasarkan pada kedekatan data pelatihan yang ada (Hidayah, 2017: 66). Objek diklasifikasikan menurut jarak dari tetangga terdekat, dengan objek yang ditugaskan ke kelas paling banyak di antara tetangga terdekatnya. Ini biasa menggunakan jarak *Euclidean* (Bouzalmat, 2013:171).

Langkah langkah dalam mengklasifikasi data dengan algoritma KNN yaitu (Karegowda et al., 2012: 148):

1. Mendefinisikan nilai K.
2. Melakukan perhitungan nilai jarak atau *Euclidean* antara data *testing* dan data *training*.
3. Mengelompokkan data berdasarkan perhitungan jarak atau *Euclidean*.
4. Mengelompokkan data berdasarkan nilai terkecil atau tetangga terdekat.
5. Memilih kelas yang paling banyak muncul dari sejumlah K yang dipilih untuk dijadikan sebagai hasil prediksi.

Data latih pada atribut ke 1 dapat dilihat pada Persamaan 1.

$$X_1 = (X_{11}, X_{12}, \ldots, X_{1n}) \quad (1)$$

Data latih pada atribut ke 2 dapat dilihat pada Persamaan 2.

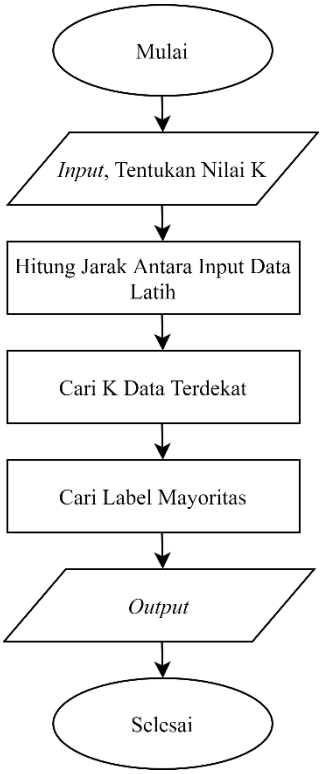
$$X_2 = (X_{21}, X_{22}, \ldots, X_{2n}) \quad (2)$$

Sementara untuk mencari jarak *Euclidean* dinyatakan dengan Persamaan 3.

$$d(X_1, X_2) = \sqrt{\sum_r^n} (a_r(X_1) - a_r(x_{12}))^2 \quad (3)$$

X_1 dan X_2 merupakan dua *record* dengan jumlah n atribut, perhitungan jarak X_1 dan X_2 digunakan untuk menentukan jarak nilai antar atribut pada *record* X_1 dan X_2. Dari keseluruhan hasil *Euclidean* akan dipilih sebanyak k dengan nilai terkecil atau terdekat untuk memprediksi dari data baru yang diklasifikasikan. Dari sejumlah kelas akan diambil kelas terbanyak atau mayoritas sebagai keputusan dari pengklasifikasian data tersebut.

Langkah-langkah dari proses KNN dapat di lihat pada Gambar 2.2.

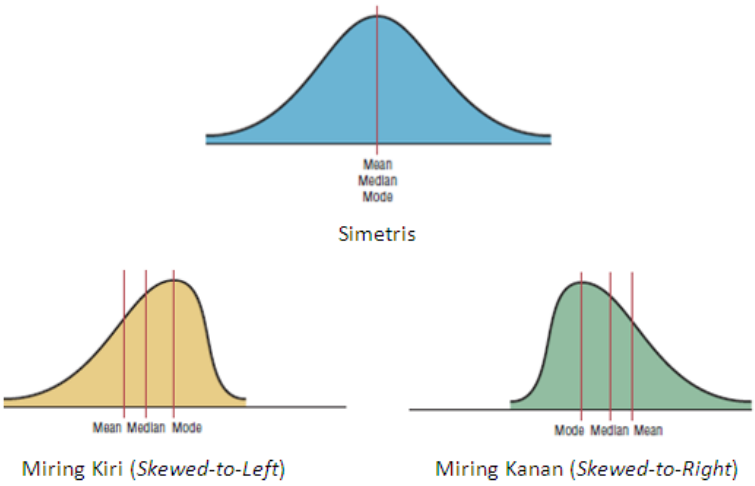


Gambar 2.2 *Flowchart* K-Nearst Neighbor

2.4 Distribusi Normal

Distribusi normal merupakan distribusi probabilitas kontinu yang paling penting dalam segala bidang Statistika. Distribusi ini memiliki karakteristik dari

fungsi kepadatan-nya yang berbentuk kurva simetris menyerupai suatu lonceng, sehingga kurva normal ini disebut sebagai kurva berbentuk lonceng (*bell-shaped curve*). Disamping itu, distribusi Normal juga disebut juga sebagai distribusi *gaussian* yang mana hal ini diberikan sebagai penghargaan untuk ahli matematika Jerman Karl Friedrich Gauss (1777 – 1855) dalam membentuk fungsi distribusi normal. Selain Kurva simetris yang menyerupai lonceng, kurva normal juga ada yang berbentuk miring ke kanan (*skewed to left*) dan ke kiri (*skewed to right*) (Bluman A.G., 2012:50-60). Berikut contoh distribusi normal yang akan di tunjukkan pada Gambar 2.3.



Gambar 2.3 Kurva Distribusi Normal

2.5 Normalisasi *Z-Score*

Pada tahap *preprocessing data mining*, terdapat tahapan yang disebut *data transformation.* Pada tahap ini dibagi menjadi sub tahapan lagi salah satunya adalah tahap normalisasi. Menurut Saranya & Manikandan (2013: 2701) normalisasi adalah suatu teknik yang memiliki tujuan untuk memetakan data ke dalam skala

tertentu dalam proses *data mining*. Sedangkan menurut Junaedi, et al., (2011: 96) Normalisasi adalah salah satu proses *data transformation* pada proses *data mining* dimana atribut numerik diskalakan dalam *range* yang lebih kecil. Nilai *Z-Score* berkisaran antara angka negatif dan positif yang tidak terbatas. Berbeda dengan seperti nilai yg di normalisasi, *Z-Score* tidak memiliki ketentuan nilai minimum dan maksimum (Panday & Jain., 2017: 39). Ada beberapa metode yang biasanya diterapkan dalam normalisasi data, diantaranya: *min-max normalization*, *Z-Score normalization* dan *normalization by decimal scaling. Z-Score normalization* adalah metode normalisasi data ketika rentang data tidak diketahui secara pasti sehingga diperlukan perhitungan rentang dengan menggunakan nilai *mean* dan standar deviasi data (Goyal et al., 2014: 32). Perhitungan normalisasi *Z-Score* dapat diselesaikan dengan Persamaan 4.

$$Z = \frac{X - \bar{X}}{SD_x} \tag{4}$$

Dimana:

Z = Hasil normalisasi

x = Nilai yang akan di normalisasi

$\bar{X}$ = Nilai Rata-Rata

SD_X = Standar Deviasi

Sedangkan untuk mecari nilai standar deviasi dapat di selesaikan dengan Persamaan 5.

$$SD_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{5}$$

Dimana:

SD_x = Standar Deviasi

n = Jumlah Sampel

x = Rata-Rata

x_i = Nilai x ke i

2.6 *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) merupakan salah satu teknik dasar dari *swarm intelligence system* untuk menyelesaikan masalah optimasi dalam pencarian ruang sebagai suatu solusi. PSO pertama kali diusulkan oleh James Kennedy dan Eberhart pada tahun 1995, dirancang untuk mensimulasikan burung dalam proses pencarian makanan. *Swarm intelligence system* melakukan penyebaran kecerdasan yang inovatif dalam menyelesaikan masalah optimasi dengan mengambil inspirasi dari contoh biologis, seperti fenomena kelompok (*swarm*) pada hewan, di mana setiap kelompok memiliki perilaku individu dalam melakukan tindakan bersama untuk mencapai tujuan yang sama.

Swarm biasa disebut agen, setiap agen bertindak berdasarkan aturan lokal meskipun tidak ada pusat kendali untuk setiap agen dalam melakukan reaksi, sehingga terbentuknya suatu kecerdasan (*intelligence*) global tanpa disadari oleh setiap agen dalam suatu kawanan untuk mencapai tujuan. Ada beberapa *properties*

yang harus diperhatikan dalam proses *swarm intelligence system* seperti kesatuan (*unity*), toleransi kesalahan (*fault tolerance*), perilaku berbasis aturan (*rule-based behavior*), otonomi (*autonomy*), skalabilitas (*scalability*), adaptasi (*adaptation*), kecepatan (*speed*), modularitas (*modularity*), dan memiliki sifat secara paralel (Sumathi & Surekha, 2010: 656).

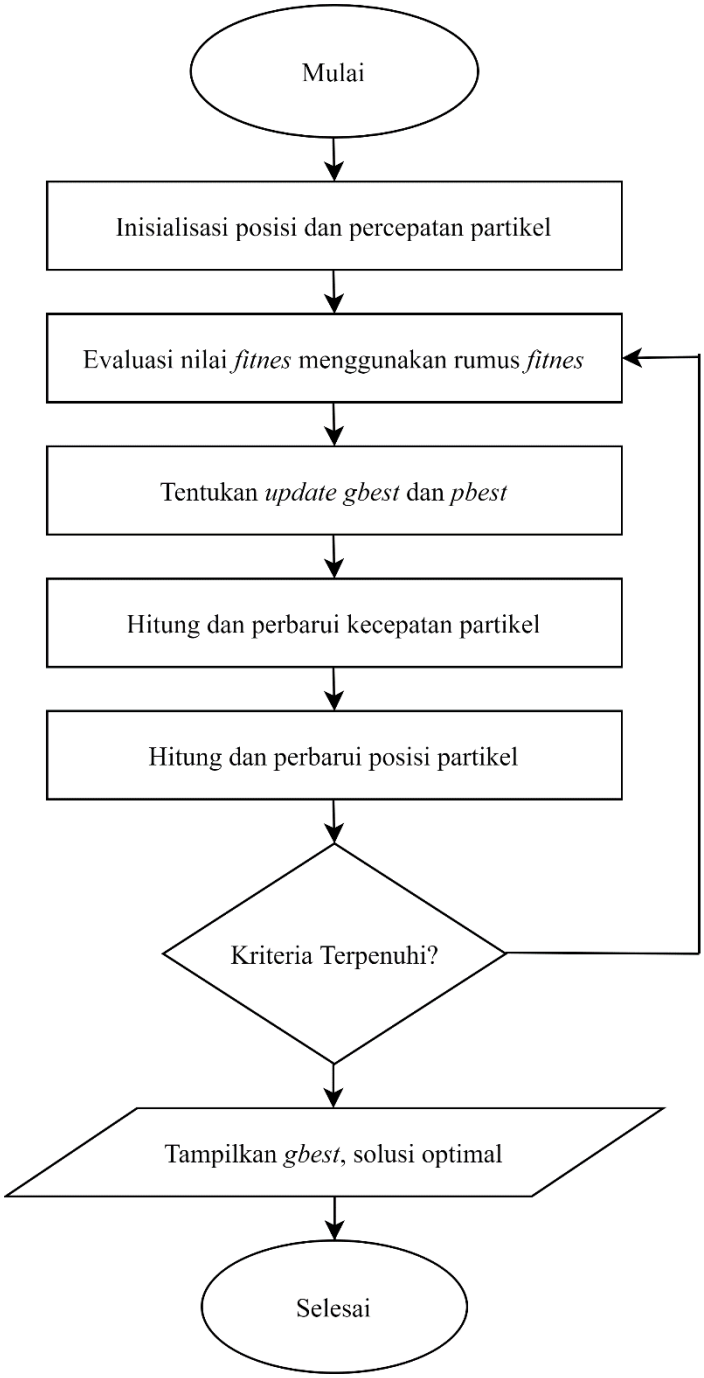
PSO merupakan salah satu metode metaheuristik yang pencarian solusinya berdasarkan populasi dari sekelompok burung atau ikan, di mana setiap populasi memiliki individu yang dapat mempengaruhi individu lainya. Individu tersebut disebut juga sebagai partikel yang diperlakukan seperti sebuah titik pada suatu dimensi ruang waktu tertentu (Santoso & Willy, 2011: 183). Pada algoritma PSO pencarian solusi dilakukan secara acak dari suatu populasi, dimana setiap partikel berkaitan dengan posisi dan kecepatan dalam melakukan pencarian baru secara dinamis berdasarkan perilaku mereka. Setiap partikel memiliki nilai *fitness* yang harus dievaluasi untuk setiap generasi berdasarkan *personal best* (*pbest*) dan *global best* (*gbest*) yang merupakan pengalaman dari setiap partikel dalam menghasilkan solusi terbaik. Pengalaman tersebut dapat digunakan sebagai parameter *weight inertia* dalam menentukan pengaruh kecepatan sebelumnya dengan kecepatan baru (Shiau, 2011: 238). Nilai *weight inertia* dapat memperbaiki kecepatan partikel dalam menyeimbangkan proses pencarian global dan lokal.

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

1. Jumlah partikel (*number of particles*) merupakan faktor yang dianggap sangat penting dalam melakukan penyelesaian masalah.

2. Bobot inersia (*inertia weight)*, memainkan peran yang sangat penting dalam kecepatan partikel dari algoritma PSO.
3. Faktor pembelajaran (*learning factors*). Parameter c_1 merupakan pengakuan koefisien, sedangkan c_2 adalah komponen sosial. Hal ini tidak terlalu penting untuk perilaku konvergensi dari algoritma PSO.
4. Rentang dan dimensi partikel (*range and dimension of particles*). Dimensi partikel dan rentang ditentukan berdasarkan masalah yang dioptimalkan.
5. Kecepatan (*velocity*) merupakan perubahan maksimum pada setiap partikel, dapat diambil selama iterasi yang didefinisikan sebagai kecepatan maksimum.
6. Menghentikan kondisi (*stopping condition*) merupakan salah satu cara apabila kriteria yang dicari sudah tercapai.

Langkah-langkah dari proses *Particle Swarm Optimization* dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Flowchart Particle Swarm Optimization*

2.7 Confussion Matrix

Confusion matrix (Kohavi & Provost, 1998) berisi informasi tentang klasifikasi aktual dan prediksi yang dilakukan oleh sistem klasifikasi. Tabel 2.1 *confusion matrix* digunakan untuk dua kelas klasifikasi. Akurasi klasifikasi, sensitivitas, spesifisitas, nilai prediktif positif, dan nilai prediktif negatif dapat didefinisikan dengan menggunakan elemen-elemen dari *confusion matrix* (Akay, 2008: 3243-3244).

Tabel 2.1 Representasi *Confusion matrix*

Actual	*Predicted*	
	Positive	*Negative*
Positive	*True Positive (TP)*	*False Negative (FN)*
Negative	*False Positive (FP)*	*True Negative (TN)*

Akurasi adalah persentase dari total data yang diidentifikasi dan dinilai. Berikut ini adalah aturan akurasi di tunjukkan pada Persamaan 6.

$$\text{Akurasi (\%)} = \frac{TP + TN}{TP + FP + FN + TN} \quad (6)$$

2.8 *Customer Churn*

Churn adalah pemutusan jasa suatu perusahaan oleh pelanggan karena pelanggan tersebut lebih memilih menggunakan layanan jasa perusahaan kompetitor (Hanifa *et al.*, 2017: 1).

Customer churn merupakan kecenderungan pelanggan untuk berhenti melakukan bisnis dengan sebuah perusahaan (Arifin, 2015: 1). *Customer Churn* didefinisikan sebagai kecenderungan pelanggan untuk berhenti melakukan bisnis dengan sebuah perusahaan. Hal ini menjadi masalah terbesar yang sedang dihadapi industri telekomunikasi saat ini. *Customer churn* dapat disebabkan oleh banyak hal, mulai dari tarif yang kompetitif antar operator, fitur dan fasilitas yang kompetitif, sampai bagaimana *provider* melayani, berinteraksi, dan mengelola hubungannya dengan pelanggan-pelanggannya.

Customer churn di industri telekomunikasi merupakan masalah yang sangat besar. Jika pada industri lain, misalnya industri kartu kredit, nilai *customer churn* yang terjadi sekitar 0.4% tiap bulan, di industri telekomunikasi dapat terjadi sampai 2,2% per bulan. Dapat dikatakan bahwa sekitar 27% pelanggan *provider churn* setiap tahunnya. Masalah c*ustomer churn* ini menjadi krusial, karena biaya yang dikeluarkan untuk mendapatkan pelanggan baru, untuk iklan, *marketing*, komisi, dan lain-lain akan jauh lebih besar dibandingkan biaya yang harus dikeluarkan untuk menjaga pelanggan yang sudah ada. Ditambah lagi belum kebanyakan pelanggan baru cenderung tidak lebih menghasilkan keuntungan dibandingkan pelanggan yang sudah lama dan bertahan. (Herawati *et al.*, 2016:27-28).

Suatu perusahaan telekomunikasi seluler dengan jumlah pelanggan yang mencapai 100 juta orang perlu menentukan pelanggan mana yang loyal dan tidak loyal sehingga bisa menentukan promosi yang tepat sasaran kepada setiap kategori pelanggan. Misalkan, jika terdapat 60% pelanggan yang termasuk

kategori tidak loyal dan mungkin berpindah (*churn*) ke operator lain, maka perusahaan ini bisa membuat promosi yang lebih menarik dan tepat supaya tidak kehilangan sangat banyak pelanggan. Promosi bisa difokuskan hanya untuk kategori pelanggan tertentu saja tanpa perlu promosi untuk kategori pelanggan yang loyal. Dengan demikian, biaya promosi bisa ditekan. Hal yang sama juga bisa dilakukan untuk perusahaan lain yang bergerak dalam bisnis apa pun. Lebih luas lagi, masalah ini digunakan untuk membangun sistem optimasi untuk CRM yang lebih lengkap dan menyeluruh (Suyanto, 2017: 5-6).

2.9 Penelitian Terkait

Penelitian ini dikembangkan dari beberapa referensi yang mempunyai keterkaitan dengan metode dan objek penelitian. Penggunaan referensi ini ditujukan untuk memberikan batasan terhadap metode dan sistem yang nantinya akan dikembangkan lebih lanjut. Berikut merupakan beberapa penelitian sebelumnya.

Dalam penelitianmya Islam *et al.*, (2007) yang berjudul "*Investigating the Performance of Naive- Bayes Classifiers and K- Nearest Neighbor Classifiers"* Meneliti tentang performa dari *Naive- Bayes Classifiers* dan *K- Nearest Neighbor*. Hasil dari penelitian tersebut bahwa *K- Nearest Neighbor* mempunyai tingkat kesalahan lebih sedikit dibandingkan dari *Naïve Bayes* dalam *klasifikasi*. *K- Nearest Neighbor* mempunyai nilai kesalahan 9.45%, sedangkan *Naïve Bayes* mempunyai nilai kesalahan 12.43%.

Dalam Penelitiannya Nanni, L, & Lumini, A.(2007) "*Particle swarm optimization for ensembling generation for evidential k-nearest-neighbour*

classifier" Meneliti tentang Penerapan *Particle Swarm Optimization* pada KNN dengan menggunakan beberapa *dataset* salah satunya adalah *German Credit Datasets* untuk optimasi Nilai Parameter K. Pada penelitian tersebut dihasilkan adalah sebesar 73,7% pada *dataset* tersebut.

Dalam penelitiannya Safitri, A, R, & Muslim, M, A. (2019) yang berjudul "*Improved Accuracy of Naive Bayes Classifier for Determination of Customer Churn Uses SMOTE and Genetic Algorithms*" Meneliti tentang peningkatan akurasi metode *Naive Bayes* dengan menggunakan Metode *SMOTE* dan Algoritma Genetika. Pada penelitian tersebut menggunakan data *German Credit dataset* yang menghasilkan akurasi sebesar 77,536%.

Dalam penelitiannya Sobran *et al*., (2013) yang berjudul "*Classification of Imbalanced Dataset using Conventional Naïve Bayes Classifier"* meneliti tentang penerapan metode *Smote* pada metode klasifikasi *Naïve Bayes* menggunakan tiga dataset yaitu *Herbaman's Survival, German Credit Dataset, Pima Indian Dataset.* Masing-masing menghasilkan akurasi sebesar 64,46% pada dataset *Herbaman's Survival,* kemuadian pada *German Credit Dataset* menghasilkan akurasi 65,31%, yang terakhir menggunakan dataset *Pima Indian Dataset* menghasilkan akurasi sebesar 67,15%.

Dalam penelitiannya Jeatrakul *et al.,* (2010) yang berjudul "*Classification of Imbalanced Data by Combining the Complementary Neural Network and SMOTE Algorithm"* penelitian tersebut membahas tentang Penerapan Metode *Smote* pada tiga metode yaitu Artificial Neural Network (ANN), *K-Nearest Neighbor* (KNN*) dan Support Vector Machine* (SVM).

Dalam penelitiannya Prihandityia *et al.*, (2018: 1-8) yang berjudul "*The Implementation of Z-Score Normalization and Boosting Techniques to Increase Accuracy of C4.5 Algorithm in Diagnosing Chronic Kidney Disease*" meneliti tentang implementasi normalisasi *Z-Score* dan *Boosting* untuk meningkatkan akurasi pada algoritma C4.5. Hasil dari implementasi tersebut dapat meningkatkan akurasi yang lebih baik. Dengan akurasi awal sebesar 96%,setelah mengimplementasikan normalisasi *Z-Score* menjadi 96,75%.

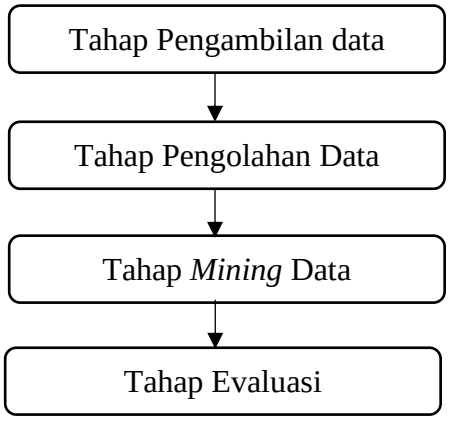
Dalam penelitiannya Alaa Tharwat *et al.*, (2018) yang berjudul *"Recognizing human activity in mobile crowdsensing environment using optimized K-NN algorithm"* meneliti tentang penggunaan metode PSO untuk mencari nilai K pada metod K-NN. Pada penelitian tersebut membuktikan bahwa PSO-KNN dapat menemukan nilai optimal dibandingkan dengan Algoritma Genetika (GA) dan *Ant Bee Colony* (ABCO).

BAB 3

METODE PENELITIAN

3.1 Studi Pendahuluan

Pada tahap ini, dikumpulkan bahan, informasi, keterangan dan teori dalam buku dan narasumber serta rujukan dari artikel, jurnal, dan karya ilmiah lainnya yang berhubungan dengan objek penelitian dan algoritma yang digunakan dalam penelitian. Bahan referensi yang digunakan mencakup *K-Nearest Neighbors, Z-Score Normalization, Particle Swarm Optimization,* serta analisis seputar peningkatan performa akurasi *K-Nearest Neighbors* dengan normalisasi *Z-Score* dan *Particle Swarm Optimization.* Dalam peningkatan akurasi algoritma *K-Nearest Neighbors* dengan menerapkan *Z-Score* dan *Particle Swarm Optimization* untuk prediksi *customer churn* memiliki tiga tahap penelitian. Ketiga tahapan tersebut antara lain tahap pengambilan data, tahap pengolahan data dan tahap *mining* data seperti ditunjukan pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.2 Pengambilan Data

Tahap pengambilan data, penulis melakukan usaha-usaha untuk mendapatkan data yang dapat dipercaya kebenarannya sehingga informasi yang didapat dapat dipertanggung jawabkan kebenarannya. Jenis data yang digunakan dalam penyusunan skripsi ini adalah data sekunder. Data sekunder merupakan sumber data yang penelitiannya diperoleh secara tidak langsung melainkan melalui perantara (diperoleh dan dicatat oleh pihak lain). Data yang digunakan dalam penelitian ini adalah *German Credit Datasets* yang diperoleh dari UCI *machine learning repository* yang terdiri dari 20 atribut dan 1 atribut *class* dengan total 1.000 data. Tabel 3.1 menunjukkan deskripsi dari *German Credit Datasets*.

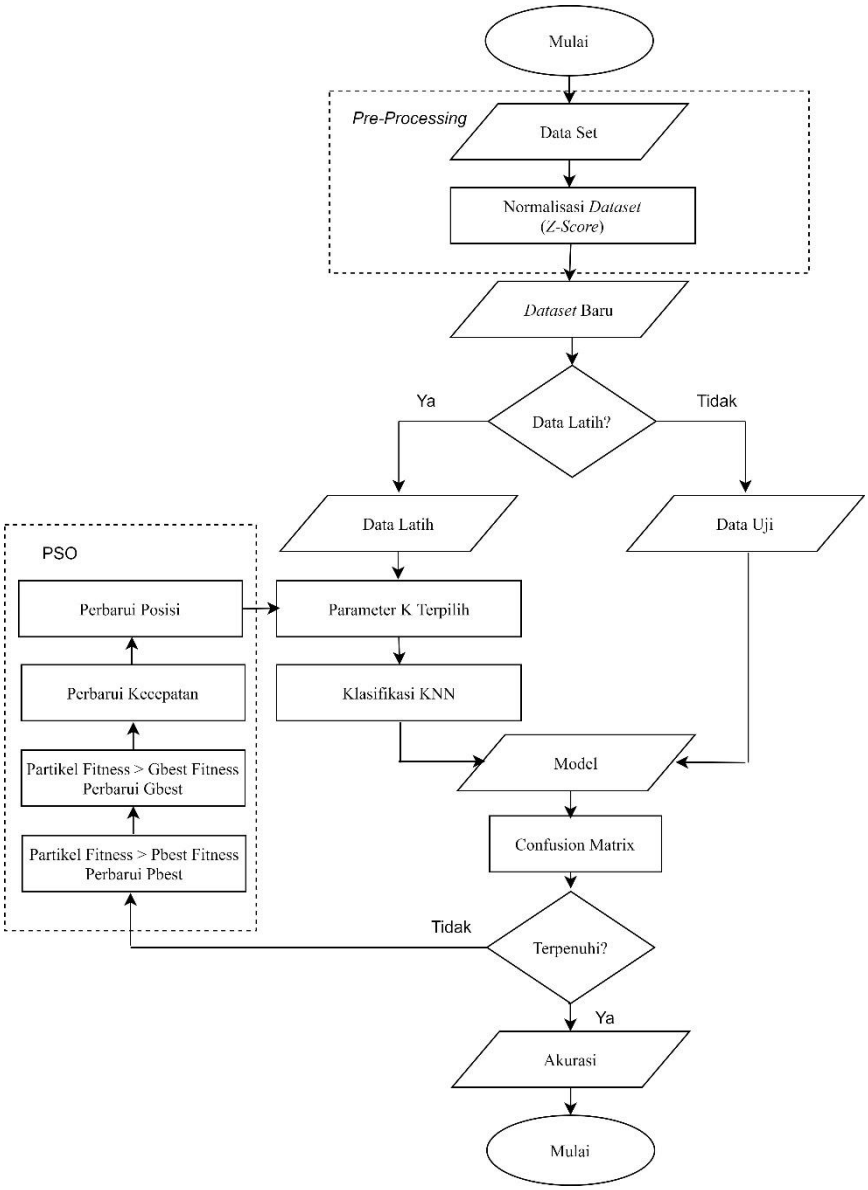
Tabel 3.1. Fitur *German Credit Datasets*

No	Fitur	Deskripsi	Tipe Atribut
1	*Status of existing checking account*	Status rekening giro/simpanan yang dimiliki oleh debitur	Numerik
2	*Duration in month*	Durasi kredit dalam bulan	Numerik
3	*Credit history*	Sejarah kredit yang pernah dimiliki	Numerik
4	*Purpose*	Tujuan mengajukan kredit	Numerik
5	*Credit Amount*	Jumlah uang yang dikredit	Numerik
6	*Saving account/bonds*	Rekening tabungan yang dimiliki	Numerik
7	*Present employment since*	Rentang lama kerjanya debitur	Numerik
8	*Installment rate in percentage of disposable income*	Tingkat angsuran dalam persentase pemakaian sekali pakai	Numerik
9	*Personal status and sex*	Status pribadi dan jenis kelamin	Numerik
10	*Other debtors/ guarantors*	Debitur/penjamin lainnya	Numerik

11	*Present residence since*	Lamanya tinggal dalam *residence*	Numerik
12	*Property*	Properti kepemilikan	Numerik
13	*Age in years*	Umur dalam tahun	Numerik
14	*Other installment plans*	Rencana angsuran lainnya	Numerik
15	*Housing*	Status kediaman yang ditinggali	Numerik
16	*Number of existing credits at this bank*	Jumlah kredit yang ada di bank ini	Numerik
17	*Job*	Pekerjaan	Numerik
18	*Number of people being liable to provide maintenance for*	Jumlah orang yang bertanggung jawab untuk menyediakan pemeliharaan	Numerik
19	*Telephone*	Kepemilikan *telephone*	Numerik
20	*Foreign worker*	Status pekerja asing	Numerik
21	*Class*	Kelas	Numerik

3.3 Tahap Pengolahan Data

Tahap pengolahan data dalam penelitian ini dilakukan dalam beberapa tahapan, dimulai dari tahapan mengkonversikan data dari ekstensi *.data* menjadi *.csv*, data yang diperoleh dari UCI *machine learning repository* sudah tersedia data numerik, sehingga tidak diperlukan transformasi data dari nominal ke numerik. Tahapan selanjutnya adalah normalisasi *Z-Score* dan tahapan *Mining data*. Untuk lebih detail mengenai metode yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.2.



Gambar3.2 *Flowchart Z-Score*, PSO, dan KNN

3.3.1 Tahapan Normalisasi Z-Score

Adapun penerapan tahapan normalisasi *Z-Score* dalam proses *preprocessing* data mining adalah sebagai berikut:

1. Cari nilai rata-rata dari setiap atribut numerik.

2. Setelah diketahui nilai *mean* atau rata-rata dari setiap atribut, langkah selanjutnya adalah mencari varians data dari atribut numerik. Varians digunakan untuk mengetahui seberapa jauh data tersebar dari nilai *mean*. Varians yang rendah menunjukkan data berkelompok sangat dekat disekitar *mean* dan sebaliknya. Dalam perhitungan varians data, setiap *value* awal dari data di atribut numerik dikurangi dengan nilai *mean* setiap atribut. Setelah diketahui nilai pengurangan tadi, langkah selanjutnya yaitu mengkuadratkan hasil pengurangan tadi kemudian menjumlahkan hasil pengkuadratan untuk setiap atributnya. Setelah diketahui hasil penjumlahan dari setiap atribut, langkah berikutnya yaitu membagi nilai penjumlahan tadi dengan (n), untuk n adalah jumlah *record* data dan akan menghasilkan nilai varians data setiap atribut numerik.
3. Setelah diketahui nilai varians datanya, langkah selanjutnya adalah mencari nilai standar deviasi. Nilai dari standar deviasi dapat diketahui dengan melakukan perhitungan akar kuadrat dari nilai varians data setiap atribut numerik. Untuk mencari nilai standar deviasi dapat di selesaikan dengan Persamaan 7.

$$SD_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad (7)$$

Dimana:

SD_x = Standar Deviasi

n = Jumlah Sampel

x = Rata-Rata

x_i = Nilai x ke i

4. Langkah berikutnya adalah menghitung nilai normalisasi *Z-Score* dengan Persamaan 8.

$$Z = \frac{X - \bar{X}}{SD_x} \quad (8)$$

Dimana:

Z = Hasil normalisasi

x = Nilai yang akan di normalisasi

$\bar{X}$ = Nilai Rata-Rata

SD_X = Standar Deviasi

3.3.2 Tahapan Pembagian Data Training dan Data Testing

Pembagian *data training* dan *data testing* dilakukan dengan pemisahan data otomatis menggunakan *metode splitter*, dengan perbandingan *training set* dan *testing set* sebesar 80:20. Pembagian ini bersifat acak dengan konsistensi pengacakan (*random state*) tertentu, sehingga tiap eksekusi nilainya tetap.

3.3.3 Tahapan *Particle Swarm Optimization*

Menurut Chen (2013: 230) untuk memulai algoritma PSO, kecepatan awal (*velocity*) dan posisi awal (*position*) ditentukan secara *random*. Kemudian proses pengembangannya adalah sebagai berikut:

1. Asumsikan bahwa ukuran kelompok atau kawanan (jumlah partikel) adalah N. Inisialisasi kecepatan dan posisi awal pada tiap partikel dalam N dimensi ditentukan secara *random* (acak). Semua partikel bergerak menuju titik

optimal dengan suatu kecepatan. Kecepatan awal semua partikel diasumsikan sama dengan nol, set iterasi i = 1.

2. Hitung *fitness* dari partikel menggunakan rumus fungsi *fitness* pada Persamaan 9.

$$Fitnes = Error\ rate = \frac{\text{jumlah } instance \text{ yang diklasifikasikan salah}}{\text{jumlah } instance} \quad (9)$$

3. Perbarui *pbest* dan *gbest* berdasarkan fungsi *fitness* dengan Persamaan 10 dan 11.

$$If\ (pos > pbest): pbest = pos \quad (10)$$

$$\text{If } (pos > gbest): gbest = pos \quad (11)$$

Nilai *fitness* partikel dibandingkan dengan *gbest*. Jika *gbest* yang terbaik maka *gbest* yang di-*update*.

4. Perbarui kecepatan dengan Persamaan 12.

$$v_{id}^{new} = w.v_{id}^{old} + c_1.r_1\left(pb_{id}^{old} - x_{id}^{old}\right) + c_2 + r_2(gb_{id}^{old} - x_{id}^{old}) \quad (12)$$

Dimana:

w = Parameter *inertia weight*, digunakan untuk mengontrol perilaku konvergensi PSO

x_{id}^{old} = Posisi awal partikel i pada d dimensi

pb_{id}^{old} = *pbest* (*personal best*) partikel i pada d dimensi

gb_{id}^{old} = *gbest* (*global best*) pada d dimensi

r_1 dan r_2 = Parameter acak (*random*) antara 0 sampai 1

c_1 dan c_2 = Konstanta akselerasi (*learning rate*), menunjukkan gerakan iterasi partikel kontrol

5. Perbarui posisi dengan Persamaan 13.

$$x_{id}^{new} = x_{id}^{old} + v_{id}^{new} \quad (13)$$

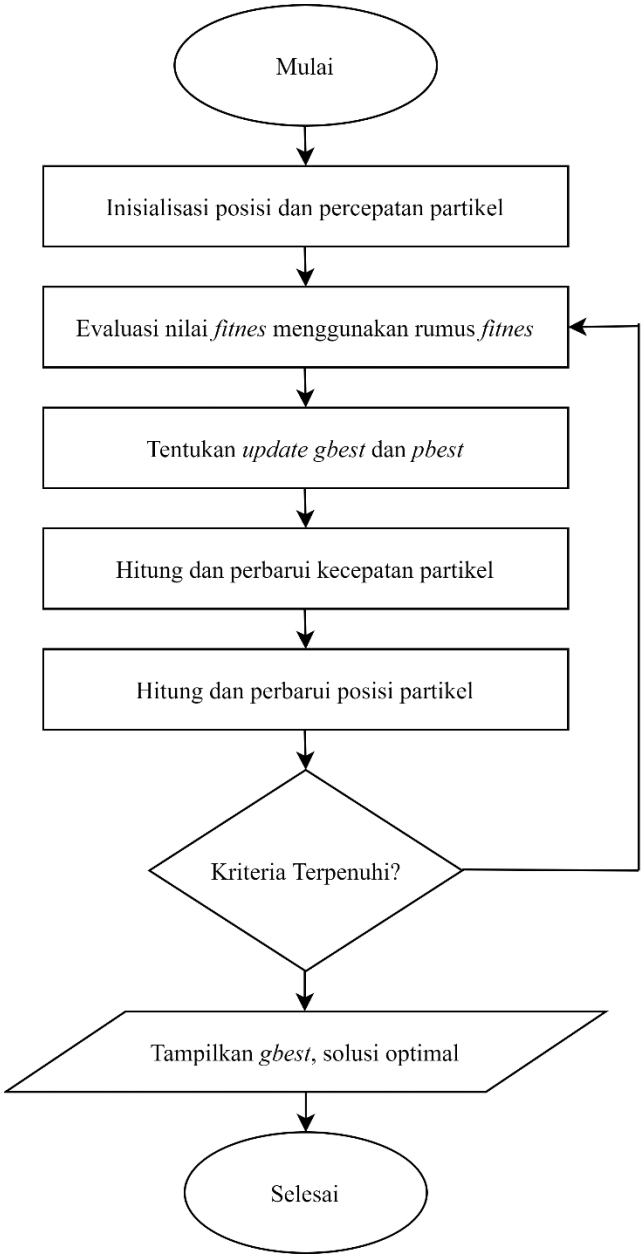
Dimana:

x_{id}^{old} = Posisi awal partikel i pada d dimensi

v_{id}^{new} = Kecepatan baru partikel i pada d dimensi

6. Cek apakah solusi yang sekarang sudah konvergen. Jika posisi semua partikel menuju ke satu nilai yang sama, maka ini disebut konvergen. Jika belum konvergen maka langkah 2 diulang dengan memperbarui iterasi i = i + 1, dengan cara menghitung nilai baru dari *pbest* dan *gbest*. Proses iterasi ini dilanjutkan sampai semua partikel menuju ke satu titik solusi yang sama. Biasanya akan ditentukan dengan kriteria penghentian (*stopping criteria*), misalnya jumlah selisih solusi sekarang dengan solusi sebelumnya sudah sangat kecil.
7. Menurut vandenBergh & Engelbrecht (2004: 232) ada 2 aspek penting dalam memilih kondisi berhenti yaitu:
    a. Kondisi berhenti tidak menyebabkan PSO *convergent premature* (memusat sebelum waktunya) di mana solusi tidak optimal yang didapat.
    b. Kondisi berhenti harus melindungi dari kondisi *oversampling* pada nilainya, jika kondisi berhenti memerlukan perhitungan yang terus menerus maka kerumitan dari proses pencarian akan meningkat.

Tahapan algoritma *Particle Swarm Optimization* dapat dilihat pada Gambar 3.3.



Gambar 3.3 *Flowchart Particle Swarm Optimization*

3.3.4 Tahapan *K-Nearest Neighbor*

Langkah langkah dalam mengklasifikasi data dengan algoritma KNN yaitu (Karegowda et al., 2012: 148):

1. Mendefinisikan nilai K.
2. Melakukan perhitungan nilai jarak atau *Euclidean* antara data testing dan data training.
3. Mengelompokkan data berdasarkan perhitungan jarak atau *Euclidean*.
4. Mengelompokkan data berdasarkan nilai terkecil atau tetangga terdekat.
5. Memilih kelas yang paling banyak muncul dari sejumlah K yang dipilih untuk dijadikan sebagai hasil prediksi.

Data latih pada atribut ke 1 dapat dilihat pada Persamaan 14.

$$X_1 = (X_{11}, X_{12}, \ldots, X_{1n}) \quad (14)$$

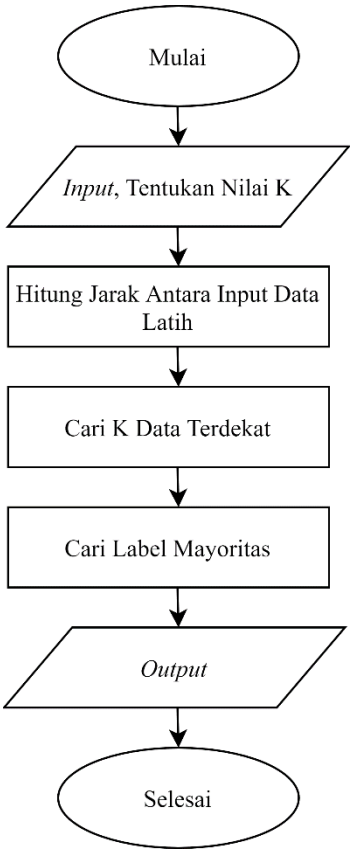
Data latih pada atribut ke 2 dapat dilihat pada Persamaan 15.

$$X_2 = (X_{21}, X_{22}, \ldots, X_{2n}) \quad (15)$$

Sementara untuk mencari jarak *Euclidean* dinyatakan dengan Persamaan 16.

$$d(X_1, X_2) = \sqrt{\sum_r^n} (a_r(X_1) - a_r(x_{12}))^2 \quad (16)$$

Tahapan algoritma *K-Nearest Neighbor* dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Flowchart K-Nearest Neighbor*

3.3.5 Tahapan Evaluasi dengan *Confusion matrix*

Tahapan evaluasi dilakukan diakhir proses penelitian. Tahap ini berguna untuk melakukan pengujian model dan menghitung akurasi yang dihasilkan. Dalam penelitian ini evaluasi dilakukan dengan *confusion matrix.* Adapun langka-langkah sebagai berikut.

1. Masukkan hasil pengujian pada Tabel *confusion matrix* seperti pada Tabel 3.2.

Tabel 3.2 Pengujian *Confusion matrix*

Actual	*Predicted*	
	Positive	*Negative*
Positive	*True Positive (TP)*	*False Negative (FN)*
Negative	*False Positive (FP)*	*True Negative (TN)*

2. Hitung nilai akurasi, tentukan akurasi tertinggi dengan Persamaan 17.

$$Akurasi = \frac{TP+TN}{P+N} x100\% \quad (17)$$

3. Kesimpulan dari hasil akurasi yang diperoleh.

3.4 Tahapan *Mining* Data

Dalam tahapan ini dilakukan proses klasifikasi KNN menggunakan proses normalisasi *Z-Score* dan proses optimasi PSO serta tanpa proses normalisasi *Z-Score* dan proses optimasi PSO untuk mengetahui adanya peningkatan akurasi yang dihasilkan. *Confusion matrix* digunakan untuk perhitungan akurasi sekaligus menjadi evaluator dalam proses klasifikasi.

3.5 Penarikan Kesimpulan

Kesimpulan yang akan diperoleh dalam penelitian ini adalah tentang bagaimana cara kerja algoritma normalisasi *Z-Score* dan *Particle Swarm Optimization* untuk meningkatkan akurasi algoritma *K-Nearest Neighbor* pada prediksi *customer churn*, serta hasil peningkatan akurasi dari optimasi algoritma *K-*

Nearest Neighbor menggunakan normalisasi *Z-Score* dan *Particle Swarm Optimization* untuk prediksi *Customer Churn.*

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil penelitian

Dalam peningkatan akurasi algoritma *K-Nearest Neighbor* dengan menerapkan Z-*Score* dan *Particle Swarm Optimization* untuk prediksi *customer churn* memiliki tiga tahap penelitian. Ketiga tahapan tersebut antara lain tahap pengambilan data, tahap pengolahan data dan tahap *mining* data. Adapun penjelasan yang lebih lengkap terkait tahapan penelitian tersebut akan diuraikan sebagai berikut.

4.1.1 Hasil Pengambilan Data

Data yang digunakan pada peneliian ini adalah *German Credit Datasets* yang diambil dari UCI *machine learning repository.* Data ini memiliki 20 fitur dan 1.000 *instance. Dataset* ini memiliki 1 fitur kelas yang terdiri dari *good and bad* atau *loyal and churn.*

4.1.2 Hasil pengolahan data

Dataset yang diperoleh 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).* Selain itu, format *file .csv* memiliki tingkat kompabilitas yang cukup tinggi, karena hampir semua program pengolahan data

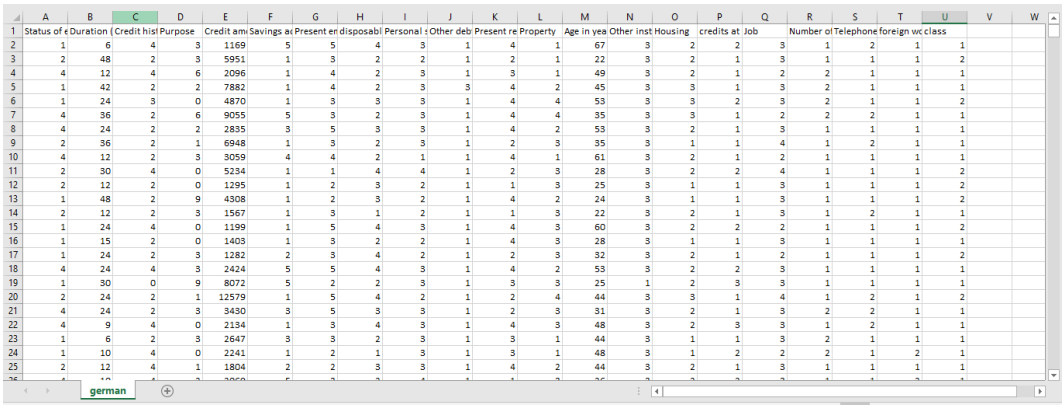
sudah mendukung format *.csv*, seperti *Microsoft Office*, *Notepad*, *MySQL, Oracle*, dll. Oleh karena itu, format *.csv* dijadikan sebagai standar dalam pengolahan data.

```
@attribute Statusofexistingcheckingaccount numeric
@attribute Duration(month) numeric
@attribute Credithistory numeric
@attribute Purpose numeric
@attribute Creditamount numeric
@attribute Savingsaccount/bonds numeric
@attribute Presentemploymentsince numeric
@attribute disposableincome numeric
@attribute Personalstatus numeric
@attribute Otherdebtors numeric
@attribute Presentresidencesince numeric
@attribute Property numeric
@attribute Ageinyears numeric
@attribute Otherinstallmentplans numeric
@attribute Housing numeric
@attribute creditsatthisbank numeric
@attribute Job numeric
@attribute Numberofpeoplebeingliabletoprovidemaintenancefor numeric
@attribute Telephone numeric
@attribute foreignworker numeric
@attribute category {1,2}

@data
1,6,4,3,1169,5,5,4,3,1,4,1,67,3,2,2,3,1,2,1,1
2,48,2,3,5951,1,3,2,2,1,2,1,22,3,2,1,3,1,1,1,2
4,12,4,6,2096,1,4,2,3,1,3,1,49,3,2,1,2,2,1,1,1
1,42,2,2,7882,1,4,2,3,3,4,2,45,3,3,1,3,2,1,1,1
1,24,3,0,4870,1,3,3,3,1,4,4,53,3,3,2,3,2,1,1,2
4,36,2,6,9055,5,3,2,3,1,4,4,35,3,3,1,2,2,2,1,1
4,24,2,2,2835,3,5,3,3,1,4,2,53,3,2,1,3,1,1,1,1
2,36,2,1,6948,1,3,2,3,1,2,3,35,3,1,1,4,1,2,1,1
4,12,2,3,3059,4,4,2,1,1,4,1,61,3,2,1,2,1,1,1,1
```

Gambar 4.1 *German Credit Datasets Dataset* dengan format *.arff*

Melakukan konversi 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 Gambar 4.2.



Status of e	Duration (	Credit hist	Purpose	Credit amo	Savings ac	Present en	disposabl	Personal s	Other debt	Present re	Property	Age in yea	Other inst	Housing	credits at	Job	Number of	Telephone	foreign wo	class
1	6	4	3	1169	5	5	4	3	1	4	1	67	3	2	2	3	1	2	1	1
2	48	2	3	5951	1	3	2	2	1	2	1	22	3	2	1	3	1	1	1	2
4	12	4	6	2096	1	4	2	3	1	3	1	49	3	2	1	2	2	1	1	1
1	42	2	2	7882	1	4	2	3	3	4	2	45	3	3	1	3	2	1	1	1
1	24	3	0	4870	1	3	3	3	1	4	4	53	3	3	2	3	2	1	1	2
4	36	2	6	9055	5	3	2	3	1	4	4	35	3	3	1	2	2	2	1	1
4	24	2	2	2835	3	5	3	3	1	4	2	53	3	2	1	3	1	1	1	1
2	36	2	1	6948	1	3	2	3	1	2	3	35	3	1	1	4	1	2	1	1
4	12	2	3	3059	4	4	2	1	1	4	1	61	3	2	1	2	1	1	1	1
2	30	4	0	5234	1	1	4	4	1	2	3	28	3	2	2	4	1	1	1	2
2	12	2	0	1295	1	2	3	2	1	1	3	25	3	1	1	3	1	1	1	2
1	48	2	9	4308	1	2	3	2	1	4	2	24	3	1	1	3	1	1	1	2
2	12	2	3	1567	1	3	1	2	1	1	3	22	3	2	1	3	1	2	1	1
1	24	4	0	1199	1	5	4	3	1	4	3	60	3	2	2	2	1	1	1	2
1	15	2	0	1403	1	3	2	2	1	4	3	28	3	1	1	3	1	1	1	1
1	24	2	3	1282	2	3	4	2	1	2	3	32	3	2	1	2	1	1	1	2
4	24	4	3	2424	5	5	4	3	1	4	2	53	3	2	2	3	1	1	1	1
1	30	0	9	8072	5	2	2	3	1	3	3	25	1	2	3	3	1	1	1	1
2	24	2	1	12579	1	5	4	2	1	2	4	44	3	3	1	4	1	2	1	2
4	24	2	3	3430	3	5	3	3	1	2	3	31	3	2	1	3	2	2	1	1
4	9	4	0	2134	1	3	4	3	1	4	3	48	3	2	3	3	1	2	1	1
1	6	2	3	2647	3	3	2	3	1	3	1	44	3	1	1	3	2	1	1	1
1	10	4	0	2241	1	2	1	3	1	3	1	48	3	1	2	2	2	1	2	1
2	12	4	1	1804	2	2	3	3	1	4	2	44	3	2	1	3	1	1	1	1

Gambar 4.2 *German Credit Datasets* dengan format *.csv*

4.1.2.1 Hasil Z-Score

Adapun penerapan normalisasi *Z-Score* dalam proses *pre-processing data mining* adalah sebagai berikut.

1. Melakukan analisis atribut dari *German Credit Datasets.*
2. Kemudian lakukan perhitungan untuk mencari nilai rata-rata dari setiap atribut numerik. Sebagai contoh perhitungan menggunakan beberapa data pada *German Credit Datasets*, dapat dilihat pada Tabel 4.1.

Tabel 4.1 Contoh data yang akan di normalisasi

existing checking account	*Duration*	*Credit history*	*...*	*Credit amount*	*tlp*	*foreign worker*	*class*
1	6	4	…	1169	2	50	1
2	48	2	…	5951	1	31	2
4	12	4	…	2096	1	32	1
1	42	2	…	7882	1	21	1
1	24	3	…	4870	1	33	2
4	36	2	…	9055	2	30	1
4	24	2	…	2835	1	26	1
2	36	2	…	6948	2	29	1
4	12	2	…	3059	1	53	1

Nilai rata-rata untuk *duration* =

$$\tilde{x}(duration) = \frac{\text{Jumlah nilai } (duration)}{\text{Banyaknya data}} = \frac{20903}{1000} = 20{,}9$$

Nilai rata-rata untuk atribut *Credit amount* =

$$\tilde{x}(credit\ amount) = \frac{Jumlah\ nilai\ (credit\ amount)}{\text{Banyaknya data}} = \frac{32712583}{1000} = 3.271{,}26$$

3. Setelah diketahui nilai rata-rata dari setiap atribut numerik, langkah selanjutnya adalah mencari nilai varians data dari atribut numerik. Varians data digunakan untuk mengetahui seberapa jauh data tersebar dari nilai rata-rata. Varians yang rendah menunjukkan data berkelompok sangat dekat disekitar nilai rata-rata dan sebaliknya. Dalam perhitungan varians data, setiap *value* awal dari data di atribut numerik dikurangi dengan nilai rata-rata setiap atribut, untuk hasil pengurangan *value* data dengan nilai rata-rata dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil pengurangan *value* awal data dengan nilai rata-rata untuk setiap atribut

existing checking account	*Duration*	*Credit history*	...	*Credit amount*	*tlp*	*foreign worker*	*class*
1	-14,9	4		-2.102,3	2	50	1
2	27,1	2		2.679,7	1	31	2
4	-8,9	4		-1.175,3	1	32	1
1	21,1	2		4.610,7	1	21	1
1	3,1	3		1.598,7	1	33	2
4	15,1	2		5.783,7	2	30	1
4	3,1	2		-432,3	1	26	1
2	15,1	2		3.676,7	2	29	1
4	-8,9	2		-212,3	1	53	1

Setelah diketahui nilai pengurangan tadi, langkah selanjutnya yaitu mengkuadratkan hasil pengurangan tadi kemudian menjumlahkan hasil

pengkuadratan untuk setiap atributnya. Berikut ini merupakan hasil perhitungan kuadrat data dapat dilihat pada Tabel 4.3

Tabel 4.3 Hasil kuadrat data setelah proses pengurangan dengan nilai rata-rata

existing checking account	*Duration*	*Credit history*	*Credit amount*	*tlp*	*foreign worker*	*class*
1	222.01	4	4.419.665,29	2	50	1
2	734,41	2	7.180.792,09	1	31	2
4	79,21	4	1.381.330,09	1	32	1
1	445,21	2	21.258.554,5	1	21	1
1	9,61	3	2.555.841,69	1	33	2
4	228,01	2	33.451.185.7	2	30	1
4	9,61	2	186.883,29	1	26	1
2	228,01	2	13.518,122.9	2	29	1
4	79,21	2	45.071,29	1	53	1

Kemudian jumlahkan setiap atribut setelah proses pengkuadratan.

Jumlah nilai pada atribut *duration* = 1.813,28

Jumlah nilai pada atribut *credit amount* = 37.028.138,24

4. Setelah diketahui hasil penjumlahan dari setiap atribut, langkah berikutnya yaitu membagi nilai penjumlahan tadi dengan (*n*), untuk *n* adalah jumlah *record* data dan akan menghasilkan nilai varians data setiap atribut numerik.

   Diketahui :

   n= 1000

   Nilai varians atribut *duration* = 1.813,28/1.000 = 1,81

   Nilai varians atribut *credit amount* = 37.028.138,24/1.000 = 37.028,14

5. Setelah diketahui nilai varians datanya, langkah selanjutnya adalah mencari nilai standar deviasi. Nilai dari standar deviasi dapat diketahui dengan melakukan perhitungan akar kuadrat dari nilai varians data setiap atribut numerik.

   Standar deviasi atribut *duration* =$\sqrt{1{,}81} = 1{,}34$

   Standar deviasi atribut *credit amount* =$\sqrt{37.028{,}14} = 192{,}43$

6. Kemudian selanjutnya adalah menghitung nilai normalisasi *Z-Score* dengan Persamaan 18.

$$Z = \frac{X - \bar{X}}{SD_x} \quad (18)$$

Dimana:

Z = Hasil normalisasi

x = Nilai yang akan di normalisasi

$\bar{X}$ = Nilai Rata-Rata

SD_X = Standar Deviasi

Berikut ini merupakan hasil perhitungan normalisasi *Z-Score* dapat dilihat pada Tabel 4.4.

Tabel 4.4 hasil normalisasi *Z-Score*

existing checking account	*Duration*	*Credit history*	...	*Credit amount*	*tlp*	*foreign worker*	*class*
1	-1,24	4		0,75	2	50	1
2	1,75	2		1,63	1	31	2
4	0,26	4		0,57	1	32	1
1	2,25	2		0,37	1	21	1
1	0,26	3		-0,73	1	33	2
4	-0,49	2		-0,66	2	30	1
4	0,26	2		-0,71	1	26	1

2	0,75	2	1,7	2	29	1
4	-1,24	2	-0,22	1	53	1

4.1.2.2 Hasil Particle Swarm Optimization

Adapun tahapan PSO dalam melakukan pemilihan nilai K terbaik adalah sebagai berikut.

1. *Input* data yang akan di optimasi seperti Tabel 4.5.

Tabel 4.5 Data sampel yang akan di optimasi.

checking account	*Duration*	*Credit history*	*purpose*	*Credit amount*	*job*	*Number of people*	*tlp*	*foreign worker*
-1.25	-1,24	1.34	0,06	0,75	0.15	-43	1.21	-0,2
-1.25	1,75	1.75	-0,3	1,63	0.15	2.33	-0.82	-0.2
-1.25	0,26	0.26	-1,03	0,57	0.15	2.33	-0.82	-0.2
-1.25	2,25	2.25	2,25	0,37	0.15	-0.43	-0.82	-0.2
-1.25	0,26	0.26	-1,03	-0,73	-1.38	-0.43	-0.82	-0.2
-1.25	-0,49	-0.49	-1,03	-0,66	0.15	-0.43	-0.82	-0.2
-1.25	0,26	0.26	0,06	-0,71	-1.38	-0.43	-0.82	-0.2
-1.25	0,75	0.75	2,25	1,7	0.15	-0.43	-0.82	-0.2
-1.25	-1,24	-1.24	0,06	-0,22	0.15	2.33	-0.82	-0.2
-1.25	-0.9	-0.9	-1,03	-0.37	-1.38	2,33	-0.82	5.1

2. Jumlah partikel ditetapkan sebanyak 60 partikel dan iterasi sebanyak 50.
3. Misal terdapat sampel sebanyak 3 partikel. Inisialisasi posisi tiap partikel (*xt*), nilai parameter *w*, c_1 dan c_2 mengacu pada penelitian sebelumnya dimana menurut Indraswari & Arifin (2017) ketika nilai koefisien akselerasi 1 (c_1) = 0,7, dan koefisien akselerasi 2 (c_2) = 0,7 merupakan nilai yang paling optimal untuk optimasi. Setelah melakukan percobaan dengan rentang 0,01 sampai dengan 1, maka nilai parameter *w* ditetapkan sebesar 0,72 dimana nilai tersebut merupakan salah satu nilai paling optimal untuk

menghasilkan akurasi tertinggi dalam proses klasifikasi. Pada tahap awal kecepatan partikel (*vt*) = 0. Posisi tiap partikel di inisialisai seperti berikut.

Partikel 1 = [100101101]

Partikel 2 = [011100110]

Partikel 3 = [100111100]

4. Hitung dan evaluasi nilai *fitness* masing-masing partikel menggunakan algoritma KNN.

   Nilai *fitness* tiap partikel:

   Partikel 1 = [100101101] = 0,5

   Partikel 2 = [011100110] = 0

   Partikel 3 = [100111100] = 0,5

5. Tentukan nilai *pbest* masing-masing partikel berdasarkan nilai akurasi yang dihasilkan oleh KNN. Tentukan *gbest* berdasarkan nilai *pbest* tertinggi.

   Pbest Partikel 1 = [100101101] = 0,5

   Pbest Partikel 2 = [011100110] = 0

   pbest Partikel 3 = [100111100] = 0,5

   gbest = 0,5

6. Hitung kecepatan dan posisi partikel menggunakan Persamaan 19 dan 20.

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times r_{1i} \times (p_{id} - x_{id}^{t}) + c_2 \times r_{2i} \times \left(p_{gd} - x_{id}^{t}\right) \quad (19)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (20)$$

Dimana r1 dan r2 merupakan nilai *random* antara 0-1.

vpartikel 1 = (0,72 *0)+0,7*0,5*(0,5-0,5)+0,7*0,5*(0,5-0,5)

= 0

vpartikel 2 = (0,72*0)+0,7*0,5*(0-0)+0,7*0,5*(0,5-0)

= 0,175

vpartikel 3 = (0,72*0)+0,7*0,5*(0,5-0,5)+0,7*0,5*(0,5-0,5)

=0

Selanjutnya perbarui posisi.

xpartikel 1 = 0,5+0

= 0,5

xpartikel 2 = 0,125+0

= 0,175

xpartikel 2 = 0,5+0

= 0,5

Maka didapatkan hasil dari perhitugan diatas yang merupakan iterasi pertama maka di peroleh nilai K terbaik sebesar 0,5.

4.1.2.3 Hasil K-Nearest Neighbor

Berikut adalah tahapan KNN dan contoh kasus dalam klasifikasi KNN.

1. *Input* data yang akan di klasifikasi menggunakan KNN diambil 9 atribut dan 9 *record* dari 20 atribut dan 1000 *record*. Seperti contoh data pada Tabel 4.6 dan Tabel 4.7.

Tabel 4.6 Data latih sampel yang akan diklasifikasi

checking account	*Duration*	*Credit history*	*purpose*	*Credit amount*	*job*	*Number of people*	*tlp*	*foreign worker*	*Class*
-1.25	-1,24	1.34	0,06	0,75	0.15	-43	1.21	-0,2	1
-1.25	1,75	1.75	-0,3	1,63	0.15	2.33	-0.82	-0.2	1
-1.25	0,26	0.26	-1,03	0,57	0.15	2.33	-0.82	-0.2	2
-1.25	2,25	2.25	2,25	0,37	0.15	-0.43	-0.82	-0.2	2
-1.25	0,26	0.26	-1,03	-0,73	-1.38	-0.43	-0.82	-0.2	2

-1.25	-0,49	-0.49	-1,03	-0,66	0.15	-0.43	-0.82	-0.2	1
-1.25	0,26	0.26	0,06	-0,71	-1.38	-0.43	-0.82	-0.2	2
-1.25	0,75	0.75	2,25	1,7	0.15	-0.43	-0.82	-0.2	1
-1.25	-1,24	-1.24	0,06	-0,22	0.15	2.33	-0.82	-0.2	1

Tabel 4.7 Data uji sampel yang akan diklasifikasi

checking account	*Duration*	*Credit history*	*purpose*	*Credit amount*	*job*	*Number of people*	*tlp*	*foreign worker*	*Class*
-1.25	-0.9	-0.9	-1,03	-0.37	-1.38	2,33	-0.82	5.1	?

2. Tentukan nilai K. sample nilai K = 3.
3. Hitung jarak tiap titik menggunakan rumus *euclidien* seperti pada Persamaan 21.

$$d(X_1,X_2) = \sqrt{\sum_r^n}(a_r(X_1) - a_r(x_{12}))^2 \quad (21)$$

a. Jarak data latih 1 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (-1{,}24 - (-90))^2 + (-1{,}34 - (-0{,}9))^2}$$
$$+ (0{,}06 - (-1{,}03))^2 + (0{,}75 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2$$
$$+ (-4{,}3 - (2{,}33))^2 + (-1{,}21 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

= 45,78

b. Jarak data latih 2 ke data uji.

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (1{,}75 - (-90))^2 + (1{,}75 - (-0{,}9))^2}$$
$$+ (-0{,}3 - (-1{,}03))^2 + (1{,}63 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2$$
$$+ (2{,}33 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

= 7,00

c. Jarak data latih 3 ke daa uji.

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (0{,}26 - (-90))^2 + (0{,}26 - (-0{,}9))^2}$$
$$+ (-1{,}03 - (-1{,}03))^2 + (0{,}57 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2$$

$$+ (2{,}33 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

$= 5{,}83$

d. Jarak data latih 4 ke data uji.

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (2{,}25 - (-90))^2 + (2{,}25 - (-0{,}9))^2}$$
$$+ (2{,}25 - (-1{,}03))^2 + (0{,}37 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2$$
$$+ (-0{,}43 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

$= 8{,}32$

e. Jarak data latih 5 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (0{,}26 - (-90))^2 + (0{,}26 - (-0{,}9))^2}$$
$$+ (-1{,}03 - (-1{,}03))^2 + (-0{,}73 - (-0{,}37))^2$$
$$+\left(-1{,}38 - (-1{,}38)\right)^2 \left(-0{,}43 - (2{,}33)\right)^2 + \left(-0{,}82 - (-0{,}82)\right)^2$$
$$+(-0{,}2 - (5{,}1))^2$$

$= 6{,}21$

f. Jarak data latih 6 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (-0{,}49 - (-90))^2 + (-0{,}49 - (-0{,}9))^2}$$
$$+ (-1{,}03 - (-1{,}03))^2 + (-0{,}66 - (-0{,}37))^2$$
$$+(0{,}15 - (-1{,}38))^2 + \ (-0{,}43 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2$$
$$+(-0{,}2 - (5{,}1))^2$$

$= 6{,}20$

g. Jarak data latih 7 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (-0{,}26 - (-90))^2 + (-0{,}26 - (-0{,}9))^2}$$
$$+ (0{,}06 - (-1{,}03))^2 + (-0{,}71 - (-0{,}37))^2 + (-1{,}38 - (-1{,}38))^2$$
$$+ (-0{,}43 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

$= 6{,}30$

h. Jarak data latih 8 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (0{,}75 - (-90))^2 + (0{,}75 - (-0{,}9))^2} + (2{,}25 - (-1{,}03))^2 + (1{,}7 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2 + (-0{,}43 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

$$= 7{,}65$$

i. Jarak data latih 9 ke data uji

$$d = \sqrt{(-1{,}25 - (-1{,}25))^2 + (-1{,}24 - (-90))^2 + (-1{,}24 - (-0{,}9))^2} + (0{,}06 - (-1{,}03))^2 + (0{,}22 - (-0{,}37))^2 + (0{,}15 - (-1{,}38))^2 + (2{,}33 - (2{,}33))^2 + (-0{,}82 - (-0{,}82))^2 + (-0{,}2 - (5{,}1))^2$$

$$= 5{,}65$$

4. Tentukan tentangga terdekat sesuai K terpilih. Dikarenakan K yang ditentukan adalah 3 maka tetangga terdekat sebagai berikut.

   Data ke-1 dengan jarak = 5,65

   Data ke-2 dengan jarak = 5,83

   Data ke-3 dengan jarak = 6,20

5. Pilih kategori data prediksi berdasarkan mayoritas kategori tetangga terdekat. Menurut sampel perhitungan diatas dengan memilih nilai K 3 label maka terpilih adalah 1.

4.1.2.4 Hasil Evaluasi dengan Confusion Matrix

Evaluasi menggunakan *confusion marix* dilakukan untk mengetahui tingkat akurasi dari perhitungan yang sudah dilakukan dengan penelitian. Adapun langkahnya sebagai berikut.

1. Masukkan hasil pengujian pada tabel *confusion matix* pada Tabel 4.8.

Tabel 4.8 Pengujian model menggunakan *confusion matrix*

Actual	*Predicted*	
	Positive	*Negative*
Positive	143	29
Negative	6	22

2. Berdasarkan Tabel 4.8, tentukan akurasi dari model yang sudah dihasilkan dengan Persamaan 17.

$$Akurasi = \frac{143+22}{200} x100\% = 82,5\%$$

3. Nyatakan kesimpulan dari hasil akurasi yang diperoleh.

4.1.2.5 Hasil Pembagian Data

Tahap pembagian datsa adalah proses membagi data menjadi 2, yakni *data training* dan *data testing* yang besarnya masing-masing adalah 80:20. Penelitian ini menggunakan metode *splitter* yang terdapat pada *library sklearn*. Pembagian dilakukan secara acak dengan menentukan konsistensi pengacakan (*random state*) dengan jumlah tertentu

4.1.3 Hasil *Mining* Data

Pada tahap ini adalah melakukan proses *mining* data sebanyak 4 kali. Pertama merupakan proses klasifikasi dengan KNN tanpa menggunakan *Z-Score* dan PSO. Kedua, proses klasifikasi dengan KNN dengan *Z-Score*. Ketiga, proses klasifikasi dengan KNN dengan PSO. Keempat, proses klasifikasi dengan KNN menggunakan *Z-Score* dan PSO. Kemudian keempat hasil klasifikasi tersebut diuji

menggunakan *confusion matrix* untuk mengetahui tingkat akurasi yang dihasilkan dari tiap proses yang telah dilakukan.

4.1.3.1 Penerapan Algoritma KNN

Penerapan yang pertama adalah penerapan algoritma KNN tanpa *Z-Score* dan PSO dengan pembagian data sebesar 80:20. Jadi besar *data training* dan *data testing* masing-masing adalah 800 dan 200 data. Hasil akurasi yang didapat dari penerapan KNN dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil akurasi algoritma KNN

Algoritma	Hasil Akurasi
KNN	68,5%

4.1.3.2 Penerapan Z-Score pada KNN

Penerapan klasifiksi dilakukan dengan menerapkan algoritma KNN dan metode normalisasi *Z-Score* pada proses *pre-processing*nya. Berikut ini merupakan hasil pengujian algoritma KNN pada *German Credit Datasets* dan normalisasi *Z-Score* sebagai proses *pre-processing* disajikan pada Tabel 4.10.

Tabel 4.10 Hasil akurasi KNN dan *Z-Score*

Algoritma	Hasil Akurasi
KNN+*Z-Score*	80,5%

4.1.3.3 Penerapan Hasil PSO pada KNN

Penerapan PSO pada KNN adalah untuk optimasi parameter K pada algoritma KNN. Setiap iterasi akan didapat posisi terbaik dengan nilai *cost* terendah

yang disebut *best cost*. Setelah melakukan seluruh iterasi, nilai *cost* dari setiap iterasi dibandingkan untuk mendapatkan *final best cost. Final best cost* inilah yang akan digunakan sebagai rekomendasi dari proses optimasi. Rekomendasi ini berupa nilai *cost* terendah dan nilai K terpilih yang dapat menghasilkan akurasi tertinggi. Hasil penerapan PSO pada KNN dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil penerapan PSO pada KNN

Algoritma	Hasil
PSO	K = 10
KNN+PSO	73,5%

4.1.3.4 Penerapan Z-Score dan PSO pada KNN

Fitur yang terpilih akan dilakukan *splitting* membagi *data training* dan data *testing* yang masing-masing besarannya adalah 800 dan 200. Kemudian pada *data training* akan dilakukan pemilihan parameter nilai K menggunakan PSO, sehinga didapatkan parameter nilai K terbaik yang akan dilanjutkan kedalam proses klasifikasi dengan menggunakan algoritma KNN. Setelah diklasifikasi maka data akan diuji dengan *data testing* dan dilakukan pengecekan menggunakan *confusion matrix* untuk mengetahui akurasi yang dihasilkan oleh algoritma KNN dengan menggunakan *Z-Score* dan PSO. Hasil akurasi dari algoritma KNN dengan menggunakan *Z-Score* dan PSO dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil akurasi algoritma KNN dengan *Z-Score* dan PSO

Algoritma	Hasil Akurasi
KNN + *Z-Score* + PSO	82,5%

Hasil akurasi yang dihasilkan algoritma KNN tanpa menggunakan *Z-Score* dan PSO adalah 68%. Sementara akurasi yang dihasilkan algoritma KNN degan menggunakan *Z-Score* dan PSO adalah 82%. Sehingga ada peningkatan akurasi sebesar 14% antara sebelum menggunakan *Z-Score* dan PSO pada algoritma KNN dalam melakukan klasifikasi *German Credit Dataset*.

4.2 Implementasi Sistem

4.2.1 Tahapan Perancangan Sistem

Tahap perencanan sistem dilakukan untuk membuat sistem bekerja sesuai kebutuhan yang diperlukan dalam penelitian. Sistem digunakan sebagai alat uji sekaligus efisiensi perhitungan dari metode.sistem yang dibuat berbasis *web* dengan Bahasa HTML (*Hyper Text Markup Language)* dan CSS (*Cascading Style Sheet*) untuk bagian *user interface*. Sementara untuk bagian perhitungan menggunakan bahasa *python* dengan *framework flask.*

4.2.2 Implementasi Algoritma

Implementasi algoritma dalam penelitian ini menggunakan bahasa pemrogaman *python* dengan *framework flask, scikit-learn library,* dan *psywarms documentation.*

4.2.2.1 Implementasi Algoritma KNN

Implementasi ini dilakukan dengan menerapkan algoritma KNN pada *German Credit Dataset*. Sebelum dilakukan pengklasifikasian oleh algoritma KNN, *German Credit Dataset* akan dilakukan pembagian data menjadi data *training* dan data *testing* menggunakan metode *splitter* dengan proporsi 80:20. *Source code* algoritma KNN ditampilkan pada Gambar 4.3.

```
#import modul dari library
import pandas as pd
import numpy as np
#mengambil modul dari library
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn import metrics

#Membaca dataset
df = pd.read_csv("dataset\german.txt")
print(df)

#Membuat array, pemisah atribut dan kelas
X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

#Membagi data training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=7, shuffle=True)

#proses klasifikasi KNN

clfKnn = neighbors.KNeighborsClassifier(weights='distance')
clfKnn.fit(X_train, y_train)
y_predKnn = clfKnn.predict(X_test)
hasilKnn = metrics.accuracy_score(y_test, y_predKnn)
```

Gambar 4.3 *Source Code* Algoritma Proses KNN

4.2.2.2 Implementasi Algoritma Z-Score pada KNN

Implementasi yang dilakukan pada tahap ini yaitu dengan menerapkan algoritma *Z-Score* pada proses *pre-processing* untuk normalisasi *German Credit Dataset*. *Source code* algoritma *Z-Score* dan KNN ditampilkan pada Gambar 4.4.

```
#import modul dari library
import pandas as pd
import numpy as np

#mengambil modul dari library
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn import metrics

#Membaca dataset
df = pd.read_csv("dataset\german.txt")
print(df)

#Membuat array, pemisah atribut dan kelas
X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

#Proses Z-Score
z = stats.zscore(X)

#Membagi data training dan testing
X_train, X_test, y_train, y_test = train_test_split(z, y,
test_size=0.20, random_state=7, shuffle=True)

#proses klasifikasi KNN
pos_lagi =2
clf = neighbors.KNeighborsClassifier(n_neighbors=pos_lagi,
weights='distance')


clfKnn = neighbors.KNeighborsClassifier(weights='distance')
clfKnn.fit(X_train, y_train)
y_predKnn = clfKnn.predict(X_test)
hasilKnn = metrics.accuracy_score(y_test, y_predKnn)

#Hasil Klasifikasi
print("Hasil KNN = ", hasilKnn)
```

Gambar 4.4 *Source Code* Algoritma *Z-Score* dan Proses KNN

4.2.2.3 Implementasi Algoritma PSO pada KNN

Pada tahap ini adalah menerapkan metode PSO pada KNN kedalam sistem. Adapun *Source code* algoritma *Z-Score* dan KNN ditampilkan pada Gambar 4.5.

```
#import modul dari library
import pandas as pd
import numpy as np
#mengambil modul dari library
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
import pyswarms as ps
from pyswarms.utils.functions import single_obj as fx

#Membaca dataset
df = pd.read_csv("dataset\german.txt")

#Memisahkan atribut kelas dan membentuk arrey
X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

#Pembagian data split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=7, shuffle=True)

#Proses optiasi PSO
def f_per_particle(x):
    n = x[0]*10
    n_new= round(n, None)
    n_lagi = int(n_new)

    clfpso = neighbors.KNeighborsClassifier(n_neighbors=n_lagi,
weights='distance')
    clfpso.fit(X_train, y_train)
    hasil = clfpso.score(X_test,y_test)
    return 1-hasil

def f(x):
    # print(x.shape)
    n_particle = 60
    j = [f_per_particle(x[i]) for i in range(n_particle)]
    return np.array(j)

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

max_bound = 3*np.ones(2)
min_bound = max_bound-2.9
bounds = (min_bound, max_bound)
print(bounds)

optimizer = ps.single.GlobalBestPSO(n_particles=60,
dimensions=2, options=options, bounds=bounds)

# Perform optimization
cost, pos = optimizer.optimize(f, iters=50)
print(cost)
posnew = round(pos[0]*10, None)
pos_lagi = int(posnew)
print(round(pos[0]*10, None))
```

```
#classifier yang digunakan di PSO
clf = neighbors.KNeighborsClassifier(n_neighbors=pos_lagi,
weights='distance')
clf.fit(X_train, y_train)

#menghitung akurasi hasil PSO
y_pred = clf.predict(X_test)

#akurasi yang dihasilkan
hasil = metrics.accuracy_score(y_test, y_pred)

#Klasifikasi KNN
clfKnn = neighbors.KNeighborsClassifier(weights='distance')
clfKnn.fit(X_train, y_train)
y_predKnn = clfKnn.predict(X_test)
hasilKnn = metrics.accuracy_score(y_test, y_predKnn)
```

Gambar 4.5 *Source Code* Algoritma PSO dan Proses KNN

4.2.2.4 Implementasi Z-Score dan PSO pada KNN

Implementasi yang dilakukan pada tahap ini yaitu dengan menerapkan algoritma *Z-Score* pada proses *pre-processing* untuk normalisasi *German Credit Dataset.* Kemudian dilaukan proses *splitting data* dengan pembagian 800:200. untuk optimasi parameter K pada algoritma KNN. Setiap iterasi akan didapat posisi terbaik dengan nilai *cost* terendah yang disebut *best cost*. Setelah melakukan seluruh iterasi, nilai *cost* dari setiap iterasi dibandingkan untuk mendapatkan *final best cost. Final best cost* inilah yang akan digunakan sebagai rekomendasi dari proses optimasi. Kemudian *dataset* akan di klasifikasi menggunkan algoritma KNN yang kemudian menghasilkan model. Model tersebut akan diuji menggunakan *data testing* untuk meghasilkan akurasi. Implementasi pada tahap ini dapat dilihat pada Gambar 4.6.

```
#import modul
import pandas as pd
import numpy as np
import pyswarms as ps

#mengambil modul tertentu dari library
from sklearn import neighbors
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from scipy import stats
from pyswarms.utils.functions import single_obj as fx

#membaca dataset
df = pd.read_csv("D:\imron\kuliah\skripsi\data\german.txt")

#memisahkan atribut kelas dan membentuk arrey
X = np.array(df.drop(['class'], 1))
y = np.array(df['class'])

#Proses Z-Score
z = stats.zscore(X)

#Pembagian data split
X_train, X_test, y_train, y_test = train_test_split(z, y,
test_size=0.20, random_state=7, shuffle=True)

#Proses optiasi PSO

def f_per_particle(x):
    n = x[0]*10
    n_new= round(n, None)
    n_lagi = int(n_new)
    clfpso = neighbors.KNeighborsClassifier(n_neighbors=n_lagi,
weights='distance')
    clfpso.fit(X_train, y_train)
    hasil = clfpso.score(X_test,y_test)
    return 1-hasil

def f(x):
    # print(x.shape)
    n_particle = 60
    j = [f_per_particle(x[i]) for i in range(n_particle)]
    return np.array(j)

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

max_bound = 3*np.ones(2)
min_bound = max_bound-2.9
bounds = (min_bound, max_bound)
print(bounds)
optimizer = ps.single.GlobalBestPSO(n_particles=60,
dimensions=2, options=options, bounds=bounds)
```

```
# Perform optimization
cost, pos = optimizer.optimize(f, iters=50)

print("**************")
print(cost)
posnew = round(pos[0]*10, None)
pos_lagi = int(posnew)
print(round(pos[0]*10, None))

#classifier yang digunakan di PSO
clf = neighbors.KNeighborsClassifier(n_neighbors=pos_lagi,
weights='distance')
clf.fit(X_train, y_train)

#menghitung akurasi hasil PSO
y_pred = clf.predict(X_test)

#akurasi yang dihasilkan
hasil = metrics.accuracy_score(y_test, y_pred)

#Klasifikasi KNN dengan Z-Score
clfKnn = neighbors.KNeighborsClassifier(weights='distance')
clfKnn.fit(X_train, y_train)
y_predKnn = clfKnn.predict(X_test)
hasilKnn = metrics.accuracy_score(y_test, y_predKnn)

#Klasifikasi KNN
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=7, shuffle=True)
classifier = neighbors.KNeighborsClassifier(weights='distance')
classifier.fit(X_train, y_train)
y_predik = classifier.predict(X_test)
hasilknnori = metrics.accuracy_score(y_test, y_predik)
```

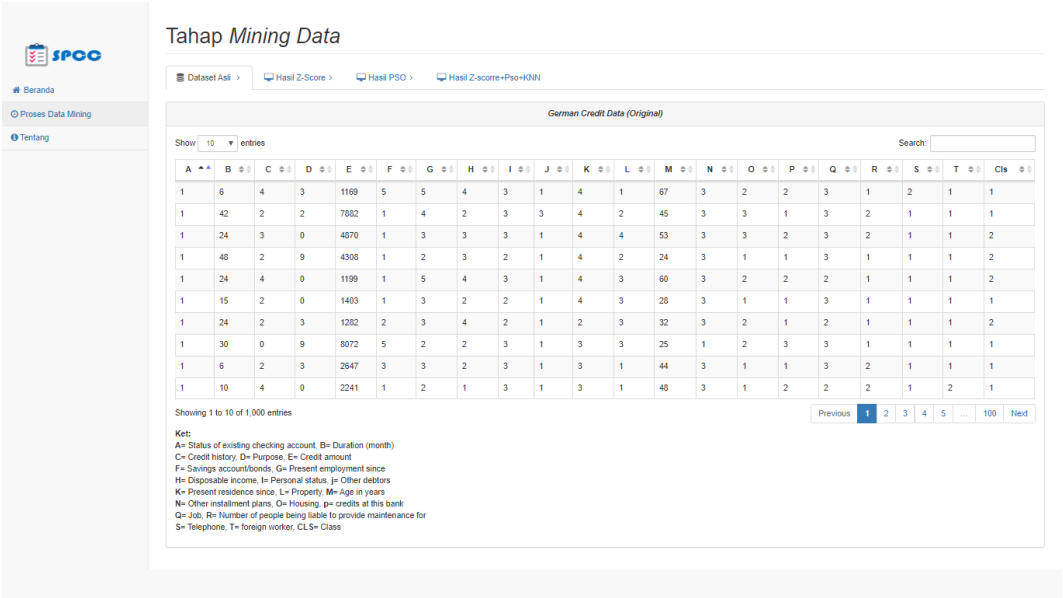
Gambar 4.6 *Source Code* Algoritma PSO dan Proses KNN

4.2.3 Implementasi *User Interface*

User interface dibuat berbasis web dengan menggunakan bahasa *HTML, CSS* dan juga *Javascript.* Dalam system ini terdapat 3 *menu* utama yaitu Beranda, Proses *data Mining* dan Tentang Aplikasi. Pada *menu* utama terdapat logo dan biodata peneliti serta nama instansi terkait seperti pada Gambar 4.7.

Gambar 4.7 Tampiln *Menu* Beranda

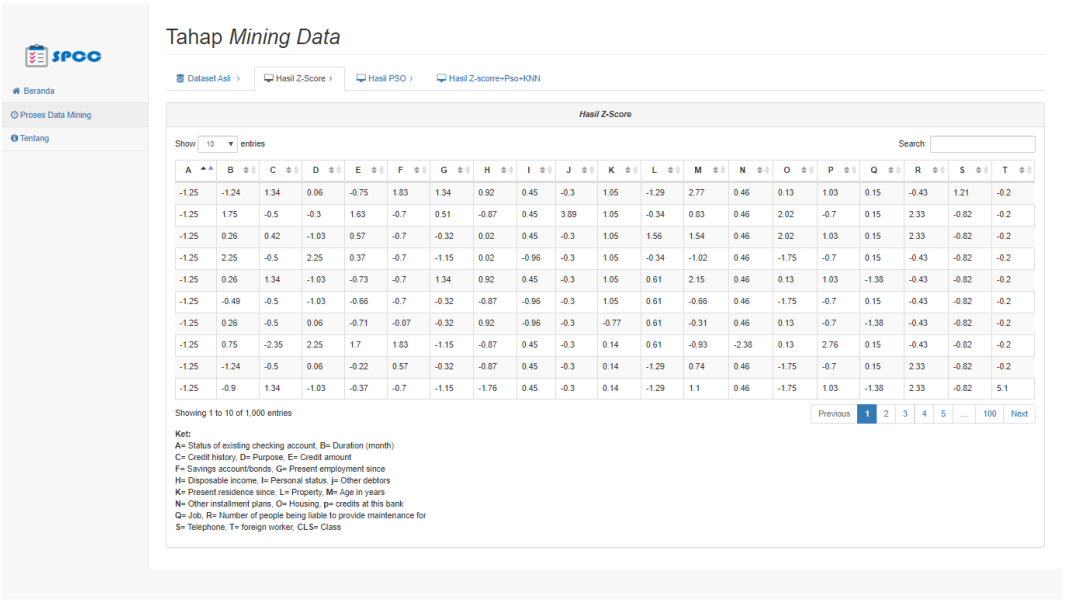
Pada menu Proses *Data Mining* terdapat beberapa bagian yang menggambarkan tahapan-tahapan penelitian yang dimulai dari data awal yang di peroleh dari *UCI Machine learning Repository* seperti pada Gambar 4.8.



Gambar 4.8 Tampilan *Dataset* Asli

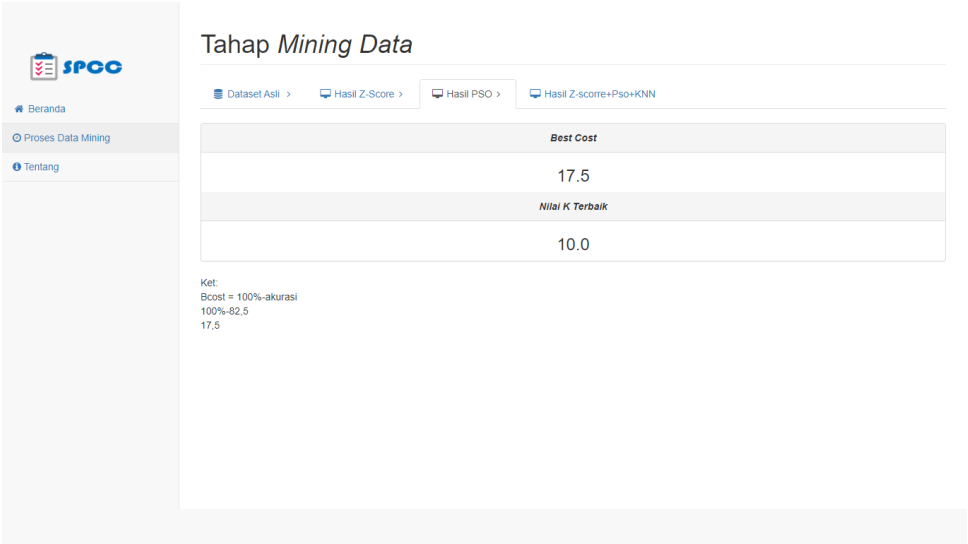
Data yang diperoleh mengalami persebaran data yang tidak normal, oleh karena itu hal tersebut diatasi dengan algoritma normalisasi *Z-Score* dengan

merepresentasikan data asli kedalam data baru dengan rentang atau *range* nilai yang hampir seragam dan rentang nilai yang sempit, dapat dilihat pada Gambar 4.9.



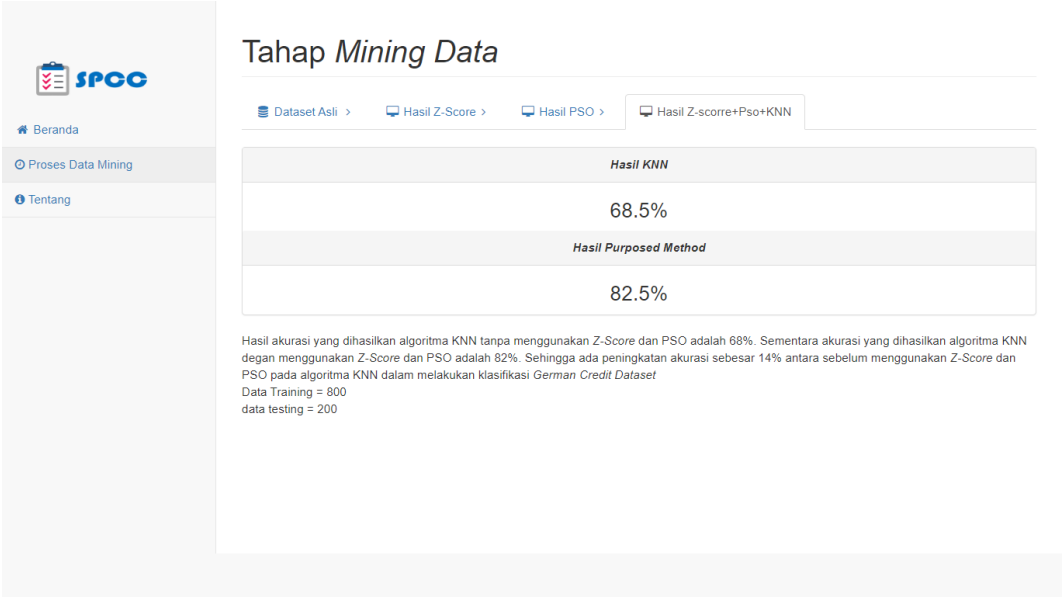
Gambar 4.9 Tampilan *Dataset* Setelah di Normalisasi Menggunakan *Z-Score*

Tab Selanjutnya dalam Proses *Data Mining* adalah Hasil PSO. Pada bagian ini sistem menampilkan *Best Cost* dan Parameter nilai K yang didapatkan Setelah melewati proses PSO. Tampilan sistem dapat dilihat pada Gambar 4.10.



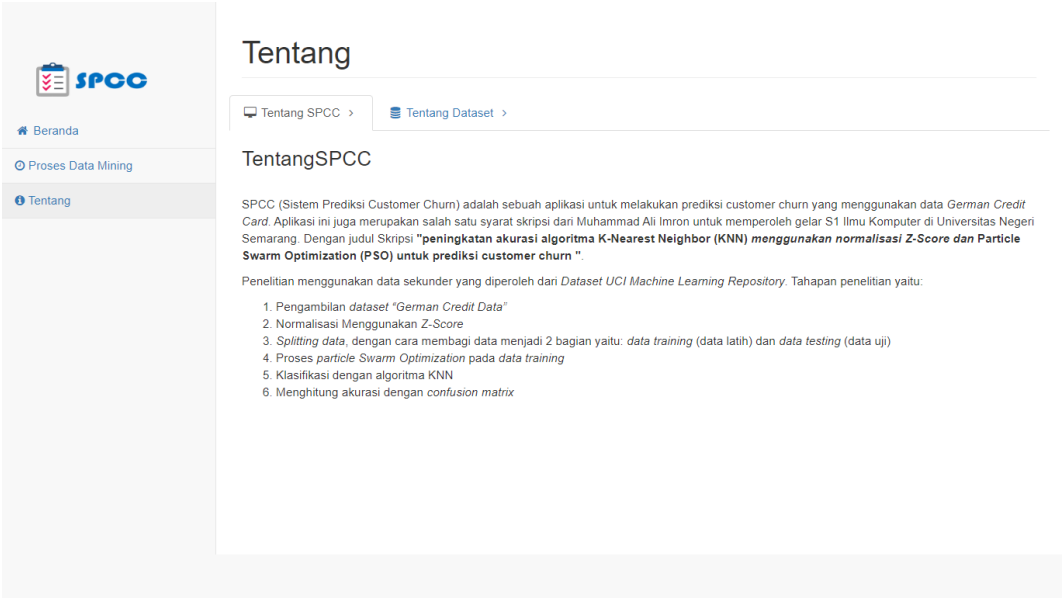
Gambar 4.10 Tampilan Hasil PSO

Pada *tab* akhir pada *menu* Proses *Data Mining* sistem menampilkan akurasi yang dihasilkan algoritma KNN tanpa menggunakan normalisasi *Z-Score* dan PSO serta hasil dari KNN dengan menggunakan *Z-Score* dan PSO. Terdapat juga keterangan yang menunjukkan adanya peningkatan akurasi yang terjadi dalam proses penelitian ini dapat dilihat pada Gambar 4.11.

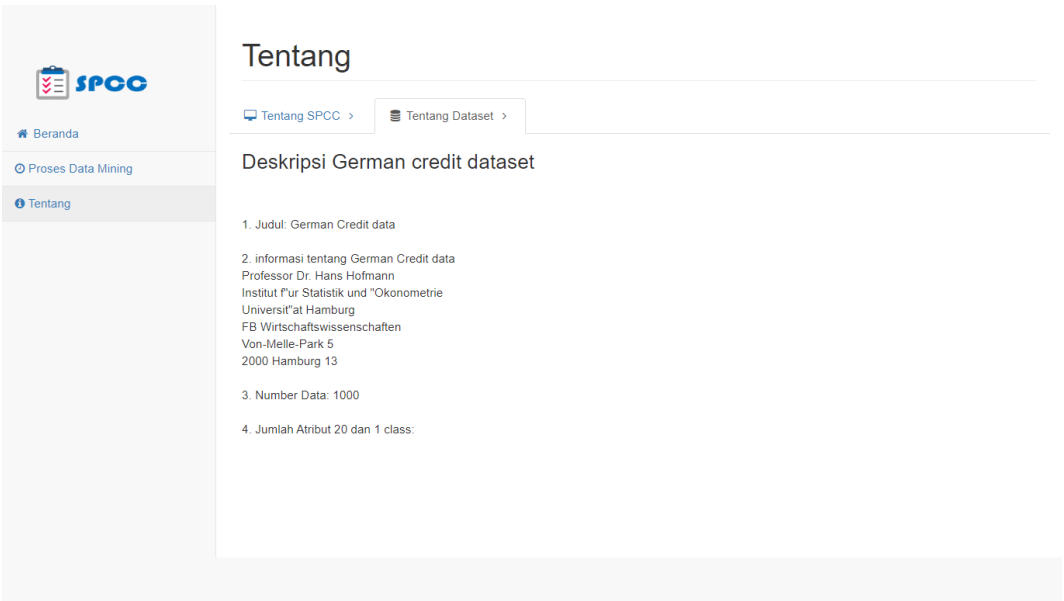


Gambar 4.11 Tampilan Hasil Akhir Proses Data Mining

Menu terakhir yaitu Tentang, didalamnya berisi tentang informasi aplikasi dan *dataset* yang di gunakan pada penelitian kali ini dapat dilihat pada Gambar 4.12 dan 4.13.



Gambar 4.12 Tampilan Tentang Aplikasi



Gambar 4.13 Tampilan Tentang *Dataset*

4.3 Pembahasan

Penelitian ini menerapkan normalisasi *Z-Score* dan PSO untuk mengoptimalkan kinerja algoritma KNN sehingga menghasilkan akurasi yang lebih baik. *Dataset* yang digunakan pada penelitian kali ini adalah *German Credit Data*

yang diperoleh dari *UCI Machine Learning Repository*. *Dataset* ini bersifat publik sehingga dapat digunakan oleh siapa saja. *German Credit data* terdiri dari 20 atribut dan 1 *class*.

Pada penelitian ini dilakukan perbandingan akurasi yang dihasilkan oleh algoritma KNN tanpa menggunakan normalisasi *Z-Score* dan PSO dengan algoritma KNN yang menggunakan normalisasi *Z-Score* dan PSO untuk mengetahui pengaruh normalisasi *Z-Score* dan PSO dalam melakukan peningkatan akurasi pada algoritma KNN.

Metode normalisasi Z-*Score* dapat memberikan peningkatan akurasi dengan merepresentasikan data asli kedalam data baru dengan rentang atau *range* nilai yang hampir seragam dan rentang nilai yang sempit. Hal tersebut dapat menyederhanakan data sehingga proses *mining* data dapat lebih optimal dan memberikan peningkatan akurasi untuk algoritma KNN.

Tahapan PSO digunakan untuk optimasi parameter K pada algoritma KNN. Setiap iterasi akan didapat posisi terbaik dengan nilai *cost* terendah yang disebut *best cost*. Setelah melakukan seluruh iterasi, nilai *cost* dari setiap iterasi dibandingkan untuk mendapatkan *final best cost. Final best cost* inilah yang akan digunakan sebagai rekomendasi dari proses optimasi. Rekomendasi ini berupa nilai *cost* terendah dan nilai K terpilih yang dapat menghasilkan akurasi tertinggi.

Penelitian ini mencatat tiap akurasi yang dihasilkan dari proses *mining* data yang telah dilakukan. Adapun hasil tersebut dapat dilihat pada Tabel 4.13.

Tabel 4.13 Hasil setiap metode yang digunakan

Algoritma	Hasil Akurasi
KNN	68,5%
KNN+*Z-Score*	80,5%
KNN+PSO	73,5
KNN + *Z-Score* + PSO	82,5%

Berdasarkan Tabel 4.13 diketahuai peningkatan tiap metode yang digunakan. Algoritma KNN tanpa menggunakan normalisasi *Z-Score* dan PSO menghasilkan sebesar 68,5%. Algoritma KNN dengan normalisasi *Z-Score* tanpa menggunakan PSO menghasilkan akurasi sebesar 80,5%. Algoritma KNN dengan PSO tanpa menggunakan normalisasi *Z-Score* menghasilkan akurasi sebesar 73,5%. *Purposed method* yang dalam hal ini adalah algoritma KNN dengan menggunakan normalisasi *Z-Score* dan PSO menghasilkan kurasi sebesar 82,5%. Sehingga dapat disimpulkan bahwa terjadi peningkatan akurasi sebesar 14% dengan membandingkan algoritma KNN tanpa menggunakan normalisasi *Z-Score* dan PSO dengan *Purposed method* dalam penelitian ini.

Tabel 4.14 Hasil Akurasi Penelitian

Metode	Akurasi
Sobran *et al*	65,3%
Safitri & Muslim	74,9%
Jeatraku *et al*	77,9%
Purposed Method	82,5%

Kelebihan dari penelitian ini yaitu dengan menerapkan *Z-Score* untuk menormalisasi data dan PSO sebagai optimasi parameter dalam proses KNN dapat meningkatkan hasil akurasi pada prediksi *customer churn* dengan menggunakan *German Credit Data*, sehingga dapat digunakan oleh peneliti selanjutnya sebagai referensi dalam melakukan penelitian klasifikasi penentuan *customer churn.* Namun penelitian ini juga memiliki kekurangan, hal ini di sebabkan oleh PSO dalam melakukan optimasi parameter nilai K. Sehingga waktu yang diperlukan cukup lama sehingga kurang efisien dalam melakukan klasifikasi *customer churn*.

BAB 5

PENUTUP

5.1 Kesimpulan

Dari hasil penelitian dan pembahasan terkait implementasi normalisasi *Z-Score* dan PSO guna meningkatkan akurasi pada algoritma KNN menggunakan *German Credit Datasets* yang diperoleh dari *UCI Machine Learning Repository* dapat ditarik kesimpulan sebagai berikut:

1. Penerapan *normalization Z-Score* dapat memberikan *range* dari setiap nilai dalam atribut pada *German Credit Datasets* sehingga meningkatkan akurasi algoritma KNN. Kemudian Penerapan PSO pada *German Credit Datasets* digunakan untuk mencari nilai parameter K trbaik, setelah didapatkan hasil optimasi nilai parameter K terbaik maka akan dilakukan klasifikasi menggunakan algoritma KNN.
2. Hasil akurasi yang diperoleh pada penerapan algoritma KNN menggunakan *normalization Z-Score* dan PSO sebesar 82,5%. Peningkatan akurasi tersebut mengalami kenaikan 14% dari pada penerapan algoritma KNN yang hanya memiliki akurasi 68,5% untuk objek *German Credit Datasets* yang berasal dari *UCI repository of machine learning*.

5.2 Saran

Untuk peneliti yang ingin mengembangkan lebih lanjut, saran yang diberikan sebagai berikut.

1. Untuk penelitian selanjutnya diharapkan dapat mengembangkan penelitian ini dengan menggunakan algoritma lain untuk meningkatkan akurasi pada klasifikasi *customer churn*.
2. Untuk melakukan pencarian parameter nilai K dapat menggunakan metode lain yang lebih efisien dibandingkan dengan PSO
3. Melakukan pengembangan metode normalisasi *Z-Scor*e, PSO, atau KNN untuk mendapatkan hasil akurasi yang lebih baik.

DAFTAR PUSTAKA

Akay, M.F. (2009). Support Vector Machines Combined with Feature Selection for Breast Cancer Diagnosis. *Expert Systems with Applications*, 36(2): 3240-3247.

Arifin, M. (2015). IG-KNN untuk Prediksi Customer Churn Telekomunikasi. *Jurnal Simetris*, 6(1): 1–10.

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.

Bai, Q. (2010). Analysis of Particle Swarm Optimization Algorithm. *Computer and Information Scinece*, 3(1): 180-184.

Bluman, A.G., (2012). *Elementary Statistics: A Step By Step Approach*. New York: McGraw-Hill.

Brandusoiu, I, & Toderean, G. (2013). Churn Prediction in the Telecommunications Sector using Support Vector Machines. *Annals of the Oradea University*, 22(1): 19–22.

Chen, R. M., & Shih, H. F. (2013). Solving University Course Timetabling Problemsusing Constriction Particle Swarm Optimization with Local Search. *Article Algorithms 2013*, *6*: 227-244.

Fei, S. W., Wang, M. J., Miao, Y. B., Tu, J., & Liu, C. L. (2009). Particle Swarm Optimization-based Support Vector Machine for Forecasting Dissolved Gases Content in Power Transformer Oil. *Energy Conversion and Management*, 50(6): 1604-1609.

Goyal, H., Sandeep, Venu, Pokuri, R., Kathula, S., & Battula, N. (2014). Normalization of Data in Data Mining. *International Journal of Software and Web Science (IJSWS)*, 10(1): 32-33.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques Third Edition.* USA: Elsivier.

Hanifa, T. T., Adiwijaya., & Al-faraby, S. 2017. Analisis Churn Prediction pada Data Pelanggan PT. Telekomunikasi dengan Logistic Regression dan Underbagging. *e-Proceeding of Engineering*. 4(2): 3210-3225.

Herawati, M., Mukhlash, I., & Wibowo, I. L. 2016. Prediksi Customer Churn Menggunakan Algoritma Fuzzy Iterative Dichotomiser. *Journal of Mathematics and Its Applications*. 13(1): 23-36.

Hermawati, F. A. (2013). *Data Mining.* Yogyakarta: CV. Andi Offset.

Hidayah, M. R., Akhlis, I., & Sugiharti, E. (2017). Recognition Number of The Vehicle Plate Using Otsu Method and K-Nearest Neighbour Classification. *Scientific Journal of Informatics (SJI)*, 4(1): 66-74.

Indraswari, R., Arifin, A. Z., & Herumurti, D. (2017). RBF Kernel Optimization Method with Particle Swarm Optimization on SVM Using the Analysis of Input Data's Movement. *Jurnal Ilmu Komputer dan Informasi*, 10(1): 36-42

Islam, M. J., Wu, Q. M. J., Ahmadi, M, & Sid-Ahmed, M. A. (2007). Investigating the Performance of Naive-Bayes Classifiers and K- Nearest Neighbor Classifiers. *International Conference on Convergence Information Technology*. South Korea: Gyeongju.

Jeatrakul, P., Wong, K. W., & Fung, C. C. (2010). Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm. In *International Conference on Neural Information Processing*. 152-159.

Junaedi, H, Budianti, H, Maryati, I., & Melani, Y. (2011). Data Transformation pada Data Mining. *IDeaTech*, 93-99.

Karegowda, A. G., Jarayam, M. A., & Majunath, A. S. (2012). Cascading K-means Clustering and K-Nearest Neighbor Classifier for Categorization of Diabetic Patients. *International Journal of Engineering and Advanced Technology (IJEAT)*, 1(3): 147–151.

Nanni, L, & Lumini, A. (2007). Particle Swarm Optimization for Ensembling Generation for Evidential K-Nearest Neighbour Classifier. *Neural Comput & Applic.* 18:105-108.

Mabrur, A.G., Lubis, R. (2012). Penerapan Data Mining Untuk Memprediksi Kriteria Nasabah Kredit. *Journal Komputer dan Informatika (KOMPUTA)*, 1(1): 53-57.

Muslim, M.A., Rukmana, S.H., Sugiharti, E., Prasetiyo, B., & Alimah, S. (2017). Optimization of C4.5 Algorithm-based Particle Swarm Optimization for Breast Cancer Diagnosis. *Journal of Physic*, 983(1): 1-5.

Pandey, A, & Jain, A. (2017) Comparative Analysis of KNN Algorithm using Various Normalization Techniques. *International Journal of Computer Network and Information Security (IJCNS).* 11(04): 36-42.

Prihanditya, H. A., Muslim, M. A., & Alamsyah. (2018). The Implementation of Normalisasi Z-Score Normalization and Boosting Techniques to Increase Accuracy of C4.5 Algorithm in Diagnosing Chronic Kidney Disease. *Scientific Journal of Informatics (SJI)*, 5(1): 1-8.

Safitri, A, R., & Muslim, M. A. (2019) Improved Accuracy of Naive Bayes Classifier for Determination of Customer Churn Uses SMOTE and Genetic Algorithms. *Scientific Journal of Informatics (SJI).* 6(1): 1-9.

Saranya, C., Munikandan. (2013). A Study on Normalization Techniques for Privacy Preserving Data Mining. *International Journal of Engineering and Technology*, 5(3): 2701.

Sasongko, T.B. (2016). Komparasi dan Analisis Kinerja Model Algoritma SVM dan PSO-SVM (Studi Kasus Klasifikasi Jalur Minat SMA). *Jurnal Teknik Infromatika dan Sistem Informasi,* 2(2): 244-253.

Siringoringo, R. (2018). Klasifikasi Data Tidak Seimbang Menggunakan Algoritma Smote dan K-Nearets Neighbor. *Journal Information System Development* (*ISD*), 1(3): 44-49.

Shiau, F.D., (2011). A Hybrid Particle Swarm Optimization for A University Course Scheduling Problem with Flexible Preferences. *Expert Systems with Applications*, 38(1): 235–248.

Sobran, N. M. M., Ahmad, A., & Ibrahim, Z. (2013). Classification of imbalanced dataset using conventional naive bayes classifier. *In International Conference on Artificial Intelligence in Computer Science and ICT*. 35-42.

Sugiharti, E., Firmansyah, S., & Devi, F.R. (2017). Predictive Evaluation of Performance of Computer Science Students of Unnes Using Data Mining Based on Naïve Bayes Classifier (NBC) Algorithm. *Journal of Theoretical and Applied Information Technology*. 95(4): 902–911.

Sumathi, S., & Surekha, P. (2010). *Computational Intelligence Paradigms: Theory and Applications Using Matlab (1st ed.).* Boca Raton: CRC Press.

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

Syulistyo, A.R., Purnomo, D.M., Rachmadi, M.F, & Wibowo, A. (2016). Particle Swarm Optimization (PSO) for Training Optimization on Convolutional Neural Network (CNN). *Jurnal Ilmu Komputer dan Informasi,* 9(1): 52-58.

Vijaya, J., & Sivasankar, E. (2017). An Efficient System for Customer Churn Prediction Through Particle Swarm Optimization Based Feature Selection Model with Simulated Annealing. *Cluster Computing,* 20: 1-13.

Tharwat, A., Mahdi, H., Elhoseny, M., & Hassanien, A.L. (2018). Recognizing Human Activity in Mobile Crowdsensing Environment using Optimized K-NN Algorithm. *Expert Systems With Applications*, 107:32-44.

Wardani, N.W., Dantes, G.R, & Indrawan, G. (2018). Prediksi Customer Churn dengan Algoritma Decision Tree C4.5 Berdasarkan Segmentasi Pelanggan pada Perusahaan Retail. *Jurnal Resistor.* 1(1):17.

Zhang, S., Deng, Z,. Cheng, D., Zong, M, & Zhu, X. (2015). Efficient KNN Classification Algorithm for Big Data. *Neurocomputing,* 195:143-148.

LAMPIRAN

Lampiran 1

Source Code Sistem

Source Code app.py

```
from flask import Flask, url_for, render_template,
send_from_directory

app = Flask(__name__)
@app.route('/')
@app.route('/index')
def index():
    return render_template('index.html')

@app.route('/tentang')
def tentang():
    return render_template('tentang.html')

@app.route('/<path:resource>')
def serveStaticResource(resource):
      return send_from_directory('static/', resource)

@app.route('/mining')
def mining():
      #Import modul yang digunakan
      import pandas as pd
      import numpy as np
      import pyswarms as ps

      #Mengambil modul tertentu dari library
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import confusion_matrix
      from sklearn.model_selection import train_test_split
      from sklearn.ensemble import BaggingClassifier
      from sklearn import metrics

      #Membaca dataset
      df1 = pd.read_csv('dataset/wbc.csv')
      df2   = pd.read_csv('dataset/wbcnew.csv')
      df3 = pd.read_csv('dataset/databagging.csv')

      #Menghitung jumlah instance pada dataset
      n1=len(df1)
      n2=len(df2)
      n3=len(df3)

      #Memisahkan atribut dengan kelas dan membentuk array
      a = np.array(df1.drop(['Class','id'],1)
      b = np.array(df1['Class']
      X = np.array(df2.drop(['Class','id'], 1))
      y = np.array(df2['Class'])
      d = np.array(df3.drop(['0','10'],1))
      e = np.array(df3['10'])
```

```
from flask import Flask, url_for, render_template,
send_from_directory

app = Flask(__name__)
@app.route('/')
@app.route('/index')
def index():
    return render_template('index.html')

@app.route('/tentang')
def tentang():
    return render_template('tentang.html')

@app.route('/<path:resource>')
def serveStaticResource(resource):
      return send_from_directory('static/', resource)

@app.route('/proses')
def proses():
      import pandas as pd
      import numpy as np

      from sklearn import neighbors
      from sklearn.model_selection import train_test_split
      from sklearn import metrics
      import pyswarms as ps
      from scipy import stats
      from pyswarms.utils.functions import single_obj as fx


      df = pd.read_csv("dataset/german.txt")

      X = np.array(df.drop(['class'], 1))
      y = np.array(df['class'])
      n1 =len (df)
      z = stats.zscore(X)
      a = np.around(z,decimals =2)
      hasilscore = pd.DataFrame(a)
      n2=len (hasilscore)


      X_train, X_test, y_train, y_test = train_test_split(z, y,
test_size=0.20, random_state=7, shuffle=True)

      def f_per_particle(x):
            n = x[0]*10
            n_new= round(n, None)
            n_lagi = int(n_new)

neighbors.KNeighborsClassifier(n_neighbors=n_lagi,
weights='distance')
            clfpso.fit(X_train, y_train)
            hasil = clfpso.score(X_test,y_test)
            return 1-hasil

      def f(x):
```

```
          # print(x.shape)
            n_particle = 60
            j = [f_per_particle(x[i]) for i in
range(n_particle)]
            return np.array(j)

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

      max_bound = 3*np.ones(2)
      min_bound = max_bound-2.9
      bounds = (min_bound, max_bound)
      print(bounds)

      optimizer = ps.single.GlobalBestPSO(n_particles=60,
dimensions=2, options=options, bounds=bounds)

            # Perform optimization
      cost, pos = optimizer.optimize(f, iters=50)


      print("*************")
      print(cost)
      posnew = round(pos[0]*10, None)
      pos_lagi = int(posnew)
      print(round(pos[0]*10, None))

      clf = neighbors.KNeighborsClassifier(n_neighbors=pos_lagi,
weights='distance')

      clf.fit(X_train, y_train)
      y_pred = clf.predict(X_test)
      hasil = metrics.accuracy_score(y_test, y_pred)

      clfKnn =
neighbors.KNeighborsClassifier(weights='distance')
      clfKnn.fit(X_train, y_train)
      y_predKnn = clfKnn.predict(X_test)
      hasilKnn = metrics.accuracy_score(y_test, y_predKnn)

      X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=7, shuffle=True)
      classifier =
neighbors.KNeighborsClassifier(weights='distance')
      classifier.fit(X_train, y_train)
      y_predik = classifier.predict(X_test)
      hasilknnori = metrics.accuracy_score(y_test, y_predik)

      print("Hasil KNN = ", hasilKnn)
      print("Hasil = ", hasil)
      print("hasil KNN Murni = ", hasilknnori)

      KNN = np.around(hasilknnori*100,decimals = 2)
      purpos= np.around(hasil*100,decimals = 2)
      bcost = np.around(cost*100,decimals= 2)
```

```
      return render_template('proses.html',
hasilscore=hasilscore, df=df, n1=n1, z=z, a=a, n2=n2, X=X, y=y,
KNN=KNN, purpos=purpos, bcost=bcost, posnew=posnew)

if __name__ == '__main__':
    app.run(debug=True)
```

Source Code base_layout.html

```
      <!DOCTYPE html>
      <html>

      <head>
          {% block head %}
          <meta charset="utf-8">
          <meta http-equiv="X-UA-Compatible" content="IE=edge">
          <meta name="viewport" content="width=device-width,
      initial-scale=1">
          <meta name="description" content="">
          <meta name="author" content="Allenh">

         <script language='JavaScript'>
              var txt="SPCC - Sistem Prediksi customer churn ";
              var speed=500;
              var refresh=null;
              function action() { document.title=txt;
              txt=txt.substring(1,txt.length)+txt.charAt(0);
              refresh=setTimeout("action()",speed);}action();
          </script>
          <link rel="shortcut icon"
      href="{{url_for('static',filename='images/logo.png')}}">
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource',
      resource='css/bootstrap.min.css') }}">
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource',
      resource='css/metisMenu.min.css') }}">
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource', resource='css/sb-admin-
      2.css') }}">
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource', resource='css/font-
      awesome.min.css') }}">
          {% block title %}{% endblock %}

          <!-- HTML5 shim and Respond.js for IE8 support of
      HTML5 elements and media queries -->
          <!-- WARNING: Respond.js doesn't work if you view the
      page via file:// -->
          <!--[if lt IE 9]>
          <script src="js/html5shiv.min.js"></script>
          <script src="js/respond.min.js"></script>
          <![endif]-->
          {% endblock %}
```

```
    </head>

    <body>

                </div>
                <!-- /.navbar-header -->

                <!-- /.navbar-top-links -->
                <div class="navbar-default sidebar"
    role="navigation">
                    <div class="sidebar-nav navbar-collapse">
                        <ul class="nav" id="side-menu">
                                <br>
                            <ul>
                                <a href='index'
    align='center'><img
    src={{url_for('static',filename='images/spcc.png')}}
    height=40 width="130"></a>
                            </ul>
                                <br>
                            <li>
                                <a href="index"><i class="fa
    fa-home fa-fw"></i> Beranda</a>
                            </li>

                            <li>
                                <a href="proses"><i class="fa
    fa-clock-o"></i> Proses Data Mining</a>
                            </li>
                            <li>
                                <a href="tentang"><i class="fa
    fa-info-circle"></i> Tentang</a>
                            </li>
                        </ul>
                    </div>
                    <!-- /.sidebar-collapse -->
                </div>
                <!-- /.navbar-static-side -->
            </nav>
            <div id="page-wrapper">
                {% with messages = get_flashed_messages() %}
    {% if messages %}
                <div class="container">>
                    <ul class="flashes">
                        {% for message in messages %}
                        <li>{{ message }}</li>
                        {% endfor %}
                    </ul>
                </div>
                {% endif %} {% endwith %} {% block content %}
    {% endblock %}
            </div>
        </div>
        {% block js %}
        <script src="{{ url_for('serveStaticResource',
    resource='js/jquery.min.js')}}"></script>
```

```
        <script src="{{ url_for('serveStaticResource',
    resource='js/bootstrap.min.js')}}"></script>
        <script src="{{ url_for('serveStaticResource',
    resource='js/metisMenu.min.js')}}"></script>
        <script src="{{ url_for('serveStaticResource',
    resource='js/sb-admin-2.js')}}"></script>
        {% endblock %}
    </body>

</html>
```

Source Code index.html

```
    {% extends "base_layout.html" %} {% block title %}{%
    endblock %}
    {% block head %}
        {{ super() }}
        <link rel="stylesheet" href="{{
    url_for('serveStaticResource',
    resource='css/timeline.css') }}">
        <link rel="stylesheet" href="{{
    url_for('serveStaticResource', resource='css/morris.css')
    }}">
    {% endblock %}
    {% block content %}

    <!-- /.row -->


    <div class="row">
        <h3 align="center">
            <img
    src={{url_for('static',filename='images/unnes.png')}}
    height="210" width="140">
            <br>
            <br>
            <strong>PENINGKATAN AKURASI ALGORITMA
            <i>K-NEAREST NEIGHBOR</i> MENGGUNAKAN NORMALISASI
    <i>Z-SCORE</i> DAN <i>PARTICLE SWARM OPTIMIZATION</i>
    UNTUK PREDIKSI <i>CUSTOMER CHURN</i>
            </strong>
        </h3>
        <br>
        <h4 align="center">
        <br>
            <strong>oleh<br>
            Muhammad ali Imron<br>
            4611415034
            </strong>
        </h4>
        <br>
        <h3 align="center">
            <strong>
                Program Studi Teknik Informatika<br>
                Jurusan Ilmu Komputer<br>
```

```
                    Fakultas Matematika dan Ilmu Pengetahuan
      Alam<br>
                    Universitas Negeri Semarang<br>
                    2020
                </strong>
            </h3>
      </div>

      <!-- /.row -->{% endblock %}
      {% block js %}
          {{ super() }}
          <script src="{{ url_for('serveStaticResource',
      resource='js/raphael-min.js')}}"></script>
          <script src="{{ url_for('serveStaticResource',
      resource='js/morris.min.js')}}"></script>
          <script src="{{ url_for('serveStaticResource',
      resource='js/morris-data.js')}}"></script>
{% endblock %}
```

Source Code mining.html

```
      % extends "base_layout.html" %}
      {% block head %}
          {{ super() }}
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource',
      resource='css/responsive.dataTables.min.css') }}">
          <link rel="stylesheet" href="{{
      url_for('serveStaticResource',
      resource='css/dataTables.bootstrap.css') }}">
      {% endblock %}
      {% block content %}
      <div class="row">
          <div class="col-lg-12">
              <h1 class="page-header">Tahap <i>Mining
      Data</i></h1>
          </div>
      </div>
      <ul class="nav nav-tabs">
          <li><a data-toggle="tab" href="#dataset"><i class="fa
      fa-database fa-fw"></i> Dataset Asli <i class="fa fa-
      angle-right fa-fw"></i></a></li>
          <li><a data-toggle="tab" href="#ZScore"><i class="fa
      fa-desktop fa-fw"></i> Hasil Z-Score<i class="fa fa-angle-
      right fa-fw"></i></a></li>
          <li><a data-toggle="tab" href="#hasilpso"><i class="fa
      fa-desktop fa-fw"></i> Hasil PSO<i class="fa fa-angle-
      right fa-fw"></i></a></li>
          <li><a data-toggle="tab" href="#hasil"><i class="fa
      fa-desktop fa-fw"></i> Hasil Z-scorre+Pso+KNN</a></li>
      </ul>

      <div class="tab-content">
      <!--- Dataset Asli-->
          <div id="dataset" class="tab-pane fade">
```

```
          <h3><strong></strong></h3>
              <!-- row -->
              <div class="row">
              <!-- col-lg-12 -->
                  <div class="col-lg-12">
                  <!-- panel -->
                      <div class="panel panel-default">
                      <!-- panel-heading -->
                          <div class="panel-heading">
                          <strong><center>
                              <i>German Credit Data
      (Original)</i>
                          </center></strong>
                          </div>
                          <!-- /.panel-heading -->
                          <!-- panel-body -->
                          <div class="panel-body">
                              <div class="row">
                              <div class="col-lg-12">
                              <!-- table-responsive -->
                              <table width="100%" class="table
      table-striped table-bordered table-hover table-responsive"
      id="dataTables-dataset">
                                      <thead>
                                          <tr>
                                              <th>A</th>
                                              <th>B</th>
                                              <th>C</th>
                                              <th>D</th>
                                              <th>E</th>
                                              <th>F</th>
                                              <th>G</th>
                                              <th>H</th>
                                              <th>I</th>
                                              <th>J</th>
                                              <th>K</th>
                                              <th>L</th>
                                              <th>M</th>
                                              <th>N</th>
                                              <th>O</th>
                                              <th>P</th>
                                              <th>Q</th>
                                              <th>R</th>
                                              <th>S</th>
                                              <th>T</th>
                                              <th>Cls</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>


                                            </tr>
                                        {% endfor %}
                                    </tbody>
                                </table>
                                <!-- /.table-responsive -->
                                <p>
                                    <Strong> Ket:</Strong> <br>
                                    <strong>A</strong>= Status of
existing checking account, <strong>B</strong>= Duration
(month)<br>
                                    <strong>C</strong>= Credit
history, <strong>D</strong>= Purpose, <strong>E</strong>=
Credit amount <br>
                                    <strong>F</strong>= Savings
account/bonds, <strong>G</strong>= Present employment
since  <br>
```

```
                                <strong>H</strong>= Disposable
income, <strong>I</strong>= Personal status,
<strong>j</strong>= Other debtors <br>
                                <strong>K</strong>= Present
residence since, <strong>L</strong>= Property,
<strong>M</strong>= Age in years <br>
                                <strong>N</strong>= Other
installment plans, <strong>O</strong>= Housing,
<strong>p</strong>= credits at this bank <br>
                                <strong>Q</strong>= Job,
<strong>R</strong>= Number of people being liable to
provide maintenance for <br>
                                <strong>S</strong>= Telephone,
<strong>T</strong>= foreign worker, <strong>CLS</strong>=
Class
                            </p>
                        </div>
                    </div>
                    <!-- /.panel-body -->
                </div>
                    </div>
                <!-- /.panel -->
                </div>
            <!-- /.col-lg-12 -->
            </div>
        <!-- /.row -->
        </div>

    <div id="ZScore" class="tab-pane fade">

        <h3><strong></strong></h3>
            <!-- row -->
            <div class="row">
            <!-- col-lg-12 -->
                <div class="col-lg-12">
                <!-- panel -->
                    <div class="panel panel-default">
                    <!-- panel-heading -->
                        <div class="panel-heading">
                        <strong><center>
                            <i>Hasil Z-Score</i>
                        </center></strong>
                        </div>
                        <!-- /.panel-heading -->
                        <!-- panel-body -->
                        <div class="panel-body">
                            <!-- table-responsive -->
                            <table width="100%" class="table
table-striped table-bordered table-hover" id="dataTables-
ZScore">
                                <thead>
                                    <tr>
                                        <th>A</th>
                                        <th>B</th>
                                        <th>C</th>
                                        <th>D</th>
```

```
                                        <th>E</th>
                                        <th>F</th>
                                        <th>G</th>
                                        <th>H</th>
                                        <th>I</th>
                                        <th>J</th>
                                        <th>K</th>
                                        <th>L</th>
                                        <th>M</th>
                                        <th>N</th>
                                        <th>O</th>
                                        <th>P</th>
                                        <th>Q</th>
                                        <th>R</th>
                                        <th>S</th>
                                        <th>T</th>

                                    </tr>
                                </thead>
                                <tbody>
                                    {% for i in range(n2): %}
                                        <tr>
                                            <td>{{ a[i,0]
    }}</td>
                                            <td>{{ a[i,1]
    }}</td>
                                            <td>{{ a[i,2]
    }}</td>
                                            <td>{{ a[i,3]
    }}</td>
                                            <td>{{ a[i,4]
    }}</td>
                                            <td>{{ a[i,5]
    }}</td>
                                            <td>{{ a[i,6]
    }}</td>
                                            <td>{{ a[i,7]
    }}</td>
                                            <td>{{ a[i,8]
    }}</td>
                                            <td>{{ a[i,9]
    }}</td>
                                            <td>{{ a[i,10]
    }}</td>
                                            <td>{{ a[i,11]
    }}</td>
                                            <td>{{ a[i,12]
    }}</td>
                                            <td>{{ a[i,13]
    }}</td>
                                            <td>{{ a[i,14]
    }}</td>
                                            <td>{{ a[i,15]
    }}</td>
                                            <td>{{ a[i,16]
    }}</td>
```

```
                                                <td>{{ a[i,17]
}}</td>
                                                <td>{{ a[i,18]
}}</td>
                                                <td>{{ a[i,19]
}}</td>

                                            </tr>
                                        {% endfor %}
                                    </tbody>
                                </table>
                                <!-- /.table-responsive -->
                                <p>
                                    <Strong> Ket:</Strong> <br>
                                    <strong>A</strong>= Status of
existing checking account, <strong>B</strong>= Duration
(month)<br>
                                    <strong>C</strong>= Credit
history, <strong>D</strong>= Purpose, <strong>E</strong>=
Credit amount <br>
                                    <strong>F</strong>= Savings
account/bonds, <strong>G</strong>= Present employment
since  <br>
                                    <strong>H</strong>= Disposable
income, <strong>I</strong>= Personal status,
<strong>j</strong>= Other debtors <br>
                                    <strong>K</strong>= Present
residence since, <strong>L</strong>= Property,
<strong>M</strong>= Age in years <br>
                                    <strong>N</strong>= Other
installment plans, <strong>O</strong>= Housing,
<strong>p</strong>= credits at this bank <br>
                                    <strong>Q</strong>= Job,
<strong>R</strong>= Number of people being liable to
provide maintenance for <br>
                                    <strong>S</strong>= Telephone,
<strong>T</strong>= foreign worker, <strong>CLS</strong>=
Class
                                </p>
                            </div>
                        <!-- /.panel-body -->
                        </div>
                    <!-- /.panel -->
                    </div>
                <!-- /.col-lg-12 -->
                </div>
            <!-- /.row -->
            </div>
        <div id="hasil" class="tab-pane fade">

            <h3><strong></strong></h3>
                <!-- row -->
                <div class="row">
                <!-- col-lg-12 -->
                    <div class="col-lg-12">
                    <!-- panel -->
```

```
                    <div class="panel panel-default">
                    <!-- panel-heading -->
                        <div class="panel-heading">
                        <strong><center>
                            <i>Hasil KNN</i>
                        </center></strong>
                        </div>
                        <h3 align="center">{{KNN}}%</h3>

                        <div class="panel-heading">
                        <strong><center>
                            <i>Hasil Purposed Method</i>
                        </center></strong>
                        </div>
                        <h3 align="center">{{purpos}}%</h3>

                    </div>
                        <p>
                                Hasil akurasi yang dihasilkan
algoritma KNN tanpa menggunakan <I>Z-Score</I> dan PSO
adalah 68%. Sementara akurasi yang dihasilkan algoritma
KNN degan menggunakan <i>Z-Score</i> dan PSO adalah 82%.
Sehingga ada peningkatan akurasi sebesar 14% antara
sebelum menggunakan <i>Z-Score</i> dan PSO pada algoritma
KNN dalam melakukan klasifikasi <i>German Credit
Dataset</i><br>
                                Data Training = 800<br>
                                data testing  = 200
                        </p>
                <!-- /.panel -->
                </div>
            <!-- /.col-lg-12 -->
            </div>
        <!-- /.row -->
        </div>

        <div id="hasilpso" class="tab-pane fade">

        <h3><strong></strong></h3>
            <!-- row -->
            <div class="row">
            <!-- col-lg-12 -->
                <div class="col-lg-12">
                <!-- panel -->
                    <div class="panel panel-default">
                    <!-- panel-heading -->
                        <div class="panel-heading">
                        <strong><center>
                            <i>Best Cost</i>
                        </center></strong>
                        </div>
                        <h3 align="center">{{bcost}}</h3>

                        <div class="panel-heading">
                        <strong><center>
                            <i>Nilai K Terbaik</i>
```

```
                            </center></strong>
                            </div>
                            <h3 align="center">{{posnew}}</h3>

                        </div>
                            <p>
                                    Ket:<br>
                                    Bcost = 100%-akurasi<br>
                                            100%-82,5<br>
                                            17,5
                            </p>
                    <!-- /.panel -->
                    </div>
                <!-- /.col-lg-12 -->
                </div>
            <!-- /.row -->
            </div>

        </div>

        <!-- /.row -->
        {% endblock %}
        {% block js %}
        {{ super() }}
        <script src="{{ url_for('serveStaticResource',
        resource='js/jquery.dataTables.min.js')}}"></script>
        <script src="{{ url_for('serveStaticResource',
        resource='js/dataTables.bootstrap.min.js')}}"></script>
        <script>
        $(document).ready(function() {
            $('#dataTables-dataset').DataTable({
                responsive: true
            });
        });
        $(document).ready(function() {
            $('#dataTables-ZScore').DataTable({
                responsive: true
            });
        });
        $(document).ready(function() {
            $('#dataTables-hasil').DataTable({
                responsive: true
            });
        });
        </script>
{% endblock %}
```

Source Code tentang.html

```
        {% extends "base_layout.html" %}
        {% block content %}
        <body>
        <div class="row">
            <div class="col-lg-12">
                <h1 class="page-header">Tentang</h1>
```

```
            </div>
                <ul class="nav nav-tabs">
                    <li><a data-toggle="tab"
href="#tentangspcc"><i class="fa fa-desktop fa-fw"></i>
Tentang SPCC <i class="fa fa-angle-right fa-
fw"></i></a></li>
                    <li><a data-toggle="tab"
href="#tentangdataset"><i class="fa fa-database fa-
fw"></i> Tentang Dataset <i class="fa fa-angle-right fa-
fw"></i></a></li>
                </ul>
        <!-- /.col-lg-12 -->
</div>
        <div class="tab-content">
            <!--- Dataset Asli-->
        <div id="tentangspcc" class="tab-pane fade">
            <p align="justify">
                <h3>TentangSPCC</h3><br>
                SPCC (Sistem Prediksi Customer Churn) adalah
sebuah aplikasi untuk melakukan prediksi customer churn
yang menggunakan data  <i>German Credit Card</i>. Aplikasi
ini juga merupakan salah satu syarat skripsi dari Muhammad
Ali Imron untuk memperoleh gelar S1 Ilmu Komputer di
Universitas Negeri Semarang. Dengan judul Skripsi
<strong>"peningkatan akurasi algoritma K-Nearest Neighbor
(KNN) <i>menggunakan normalisasi Z-Score dan </i>Particle
Swarm Optimization (PSO) </i> untuk prediksi customer
churn "</strong>.
            </p>
            <p align="justify">
                Penelitian menggunakan data sekunder yang
diperoleh dari <i>Dataset UCI Machine Learning
Repository</i>. Tahapan penelitian yaitu:
                <ol>
                    <li>Pengambilan <i>dataset “German Credit
Data”</i></li>
                    <li>Normalisasi Menggunakan <i>Z-
Score</i></li>
                    <li><i>Splitting data</i>, dengan cara
membagi data menjadi 2 bagian yaitu: <i>data training</i>
(data latih) dan <i>data testing</i> (data uji)</li>
                    <li>Proses <i>particle Swarm
Optimization</i> pada <i>data training</i></li>
                    <li>Klasifikasi dengan algoritma KNN</li>
                    <li>Menghitung akurasi dengan <i>confusion
matrix</i></li>
                </ol>
            </p>
        </div>
        <div id="tentangdataset" class="tab-pane fade">
            <p align='justify'>
                <h3> Deskripsi German credit dataset</h3><br>
                <br>
                1. Judul: German Credit data<br>
                <br>
                2. informasi tentang German Credit data<br>
```

```
                    Professor Dr. Hans Hofmann<br>
                    Institut f"ur Statistik und "Okonometrie<br>
                    Universit"at Hamburg<br>
                    FB Wirtschaftswissenschaften<br>
                    Von-Melle-Park 5<br>
                    2000 Hamburg 13<br>
                    <br>
                    3. Number Data:  1000<br>
                        <br>
                        4. Jumlah Atribut 20 dan 1 class:<br>

      </p>
      </div>
      </div>
      </body>
{% endblock %}
```

Lampiran 2

German Credit Dataset

```
tatus of existing checking account,Duration (month),Credit
history,Purpose,Credit amount,Savings account/bonds,Present
employment since,disposable income,Personal status,Other
debtors,Present residence since,Property,Age in years,Other
installment plans,Housing,credits at this bank,Job,Number of
people being liable to provide maintenance for,Telephone,foreign
worker,class
1,6,4,3,1169,5,5,4,3,1,4,1,67,3,2,2,3,1,2,1,1
2,48,2,3,5951,1,3,2,2,1,2,1,22,3,2,1,3,1,1,1,2
4,12,4,6,2096,1,4,2,3,1,3,1,49,3,2,1,2,2,1,1,1
1,42,2,2,7882,1,4,2,3,3,4,2,45,3,3,1,3,2,1,1,1
1,24,3,0,4870,1,3,3,3,1,4,4,53,3,3,2,3,2,1,1,2
4,36,2,6,9055,5,3,2,3,1,4,4,35,3,3,1,2,2,2,1,1
4,24,2,2,2835,3,5,3,3,1,4,2,53,3,2,1,3,1,1,1,1
2,36,2,1,6948,1,3,2,3,1,2,3,35,3,1,1,4,1,2,1,1
4,12,2,3,3059,4,4,2,1,1,4,1,61,3,2,1,2,1,1,1,1
2,30,4,0,5234,1,1,4,4,1,2,3,28,3,2,2,4,1,1,1,2
2,12,2,0,1295,1,2,3,2,1,1,3,25,3,1,1,3,1,1,1,2
1,48,2,9,4308,1,2,3,2,1,4,2,24,3,1,1,3,1,1,1,2
2,12,2,3,1567,1,3,1,2,1,1,3,22,3,2,1,3,1,2,1,1
1,24,4,0,1199,1,5,4,3,1,4,3,60,3,2,2,2,1,1,1,2
1,15,2,0,1403,1,3,2,2,1,4,3,28,3,1,1,3,1,1,1,1
1,24,2,3,1282,2,3,4,2,1,2,3,32,3,2,1,2,1,1,1,2
4,24,4,3,2424,5,5,4,3,1,4,2,53,3,2,2,3,1,1,1,1
1,30,0,9,8072,5,2,2,3,1,3,3,25,1,2,3,3,1,1,1,1
2,24,2,1,12579,1,5,4,2,1,2,4,44,3,3,1,4,1,2,1,2
4,24,2,3,3430,3,5,3,3,1,2,3,31,3,2,1,3,2,2,1,1
4,9,4,0,2134,1,3,4,3,1,4,3,48,3,2,3,3,1,2,1,1
1,6,2,3,2647,3,3,2,3,1,3,1,44,3,1,1,3,2,1,1,1
1,10,4,0,2241,1,2,1,3,1,3,1,48,3,1,2,2,2,1,2,1
2,12,4,1,1804,2,2,3,3,1,4,2,44,3,2,1,3,1,1,1,1
4,10,4,2,2069,5,3,2,4,1,1,3,26,3,2,2,3,1,1,2,1
1,6,2,2,1374,1,3,1,3,1,2,1,36,1,2,1,2,1,2,1,1
4,6,0,3,426,1,5,4,4,1,4,3,39,3,2,1,2,1,1,1,1
3,12,1,3,409,4,3,3,2,1,3,1,42,3,1,2,3,1,1,1,1
2,7,2,3,2415,1,3,3,3,3,2,1,34,3,2,1,3,1,1,1,1
1,60,3,9,6836,1,5,3,3,1,4,4,63,3,2,2,3,1,2,1,2
2,18,2,9,1913,4,2,3,4,1,3,1,36,1,2,1,3,1,2,1,1
1,24,2,2,4020,1,3,2,3,1,2,3,27,2,2,1,3,1,1,1,1
2,18,2,0,5866,2,3,2,3,1,2,3,30,3,2,2,3,1,2,1,1
4,12,4,9,1264,5,5,4,3,1,4,4,57,3,1,1,2,1,1,1,1
3,12,2,2,1474,1,2,4,2,1,1,2,33,1,2,1,4,1,2,1,1
2,45,4,3,4746,1,2,4,3,1,2,2,25,3,2,2,2,1,1,1,2
4,48,4,6,6110,1,3,1,3,1,3,4,31,1,3,1,3,1,2,1,1
3,18,2,3,2100,1,3,4,3,2,2,1,37,2,2,1,3,1,1,1,2
3,10,2,4,1225,1,3,2,3,1,2,3,37,3,2,1,3,1,2,1,1
2,9,2,3,458,1,3,4,3,1,3,1,24,3,2,1,3,1,1,1,1
4,30,2,3,2333,3,5,4,3,1,2,3,30,1,2,1,4,1,1,1,1
2,12,2,3,1158,3,3,3,1,1,1,3,26,3,2,1,3,1,2,1,1
2,18,3,5,6204,1,3,2,3,1,4,1,44,3,2,1,2,2,2,1,1
1,30,4,1,6187,2,4,1,4,1,4,3,24,3,1,2,3,1,1,1,1
1,48,4,1,6143,1,5,4,2,1,4,4,58,2,3,2,2,1,1,1,2
```

```
4,11,4,0,1393,1,2,4,2,1,4,3,35,3,2,2,4,1,1,1,1
4,36,2,3,2299,3,5,4,3,1,4,3,39,3,2,1,3,1,1,1,1
1,6,2,1,1352,3,1,1,2,1,2,2,23,3,1,1,1,1,2,1,1
4,11,4,0,7228,1,3,1,3,1,4,2,39,3,2,2,2,1,1,1,1
4,12,2,3,2073,2,3,4,2,2,2,1,28,3,2,1,3,1,1,1,1
2,24,3,2,2333,5,2,4,3,1,2,2,29,1,2,1,2,1,1,1,1
2,27,3,1,5965,1,5,1,3,1,2,3,30,3,2,2,4,1,2,1,1
4,12,2,3,1262,1,3,3,3,1,2,3,25,3,2,1,3,1,1,1,1
4,18,2,1,3378,5,3,2,3,1,1,2,31,3,2,1,3,1,2,1,1
2,36,3,0,2225,1,5,4,3,1,4,4,57,1,3,2,3,1,2,1,2
4,6,1,0,783,5,3,1,3,3,2,1,26,2,2,1,2,2,1,1,1
2,12,2,3,6468,5,1,2,3,1,1,4,52,3,2,1,4,1,2,1,2
4,36,4,3,9566,1,3,2,2,1,2,3,31,2,2,2,3,1,1,1,1
3,18,2,0,1961,1,5,3,2,1,2,3,23,3,2,1,4,1,1,1,1
1,36,4,2,6229,1,2,4,2,2,4,4,23,3,1,2,2,1,2,1,2
2,9,2,9,1391,1,3,2,4,1,1,1,27,1,2,1,3,1,2,1,1
2,15,4,3,1537,5,5,4,3,3,4,1,50,3,2,2,3,1,2,1,1
2,36,0,9,1953,1,5,4,3,1,4,4,61,3,3,1,4,1,2,1,2
2,48,0,9,14421,1,3,2,3,1,2,3,25,3,2,1,3,1,2,1,2
4,24,2,3,3181,1,2,4,2,1,4,2,26,3,2,1,3,1,2,1,1
4,27,2,5,5190,5,5,4,3,1,4,2,48,3,2,4,3,2,2,1,1
4,12,2,3,2171,1,2,2,2,1,2,3,29,1,2,1,3,1,1,1,1
2,12,2,0,1007,4,3,4,4,1,1,1,22,3,2,1,3,1,1,1,1
4,36,2,6,1819,1,3,4,3,1,4,4,37,2,3,1,3,1,2,1,2
4,36,2,3,2394,5,3,4,2,1,4,3,25,3,2,1,3,1,1,1,1
4,36,2,1,8133,1,3,1,2,1,2,2,30,1,2,1,3,1,1,1,1
4,7,4,3,730,5,5,4,3,1,2,2,46,3,1,2,2,1,2,1,1
1,8,4,10,1164,1,5,3,3,1,4,4,51,1,3,2,4,2,2,1,1
2,42,4,9,5954,1,4,2,2,1,1,1,41,1,2,2,2,1,1,1,1
1,36,2,6,1977,5,5,4,3,1,4,4,40,3,2,1,4,1,2,1,2
1,12,4,1,1526,1,5,4,3,1,4,4,66,3,3,2,4,1,1,1,1
1,42,2,3,3965,1,2,4,3,1,3,3,34,3,2,1,3,1,1,1,2
2,11,3,3,4771,1,4,2,3,1,4,2,51,3,2,1,3,1,1,1,1
4,54,0,1,9436,5,3,2,3,1,2,2,39,3,2,1,2,2,1,1,1
2,30,2,2,3832,1,2,2,4,1,1,2,22,3,2,1,3,1,1,1,1
4,24,2,3,5943,5,2,1,2,1,1,3,44,3,2,2,3,1,2,1,2
4,15,2,3,1213,3,5,4,3,1,3,2,47,2,2,1,3,1,2,1,1
4,18,2,9,1568,2,3,3,2,1,4,2,24,3,1,1,2,1,1,1,1
1,24,2,10,1755,1,5,4,2,3,4,1,58,3,2,1,2,1,2,1,1
1,10,2,3,2315,1,5,3,3,1,4,1,52,3,2,1,2,1,1,1,1
4,12,4,9,1412,1,3,4,2,3,2,1,29,3,2,2,4,1,2,1,1
2,18,4,2,1295,1,2,4,2,1,1,2,27,3,2,2,3,1,1,1,1
2,36,2,6,12612,2,3,1,3,1,4,4,47,3,3,1,3,2,2,1,2
1,18,2,0,2249,2,4,4,3,1,3,3,30,3,2,1,4,2,2,1,1
1,12,0,5,1108,1,4,4,3,1,3,1,28,3,2,2,3,1,1,1,2
4,12,4,3,618,1,5,4,3,1,4,1,56,3,2,1,3,1,1,1,1
1,12,4,1,1409,1,5,4,3,1,3,1,54,3,2,1,3,1,1,1,1
4,12,4,3,797,5,5,4,2,1,3,2,33,1,2,1,2,2,1,1,2
3,24,4,2,3617,5,5,4,3,2,4,4,20,3,1,2,3,1,1,1,1
2,12,2,0,1318,4,5,4,3,1,4,1,54,3,2,1,3,1,2,1,1
2,54,0,9,15945,1,2,3,3,1,4,4,58,3,1,1,3,1,2,1,2
4,12,4,6,2012,5,4,4,2,1,2,3,61,3,2,1,3,1,1,1,1
2,18,2,9,2622,2,3,4,3,1,4,3,34,3,2,1,3,1,1,1,1
2,36,4,3,2337,1,5,4,3,1,4,1,36,3,2,1,3,1,1,1,1
2,20,3,1,7057,5,4,3,3,1,4,2,36,1,1,2,4,2,2,1,1
4,24,2,0,1469,2,5,4,4,1,4,1,41,3,1,1,2,1,1,1,1
```

```
2,36,2,3,2323,1,4,4,3,1,4,3,24,3,1,1,3,1,1,1,1
4,6,3,3,932,1,3,3,2,1,2,1,24,3,2,1,3,1,1,1,1
2,9,4,2,1919,1,4,4,3,1,3,3,35,3,1,1,3,1,2,1,1
4,12,2,1,2445,5,2,2,4,1,4,3,26,3,1,1,3,1,2,1,1
2,24,4,10,11938,1,3,2,3,2,3,3,39,3,2,2,4,2,2,1,2
4,18,1,0,6458,1,5,2,3,1,4,4,39,1,2,2,4,2,2,1,2
2,12,2,0,6078,1,4,2,3,1,2,3,32,3,2,1,3,1,1,1,1
1,24,2,2,7721,5,2,1,2,1,2,2,30,3,2,1,3,1,2,2,1
2,14,2,9,1410,3,5,1,4,1,2,1,35,3,2,1,3,1,2,1,1
2,6,3,9,1449,2,5,1,1,1,2,3,31,1,2,2,3,2,1,1,1
3,15,2,6,392,1,2,4,2,1,4,2,23,3,1,1,3,1,2,1,1
2,18,2,0,6260,1,4,3,3,1,3,1,28,3,1,1,2,1,1,1,1
4,36,4,0,7855,1,3,4,2,1,2,1,25,2,2,2,3,1,2,1,2
1,12,2,3,1680,3,5,3,4,1,1,1,35,3,2,1,3,1,1,1,1
4,48,4,3,3578,5,5,4,3,1,1,1,47,3,2,1,3,1,2,1,1
1,42,2,3,7174,5,4,4,2,1,3,3,30,3,2,1,4,1,2,1,2
1,10,4,2,2132,5,2,2,2,2,3,1,27,3,1,2,3,1,1,2,1
1,33,4,2,4281,3,3,1,2,1,4,3,23,3,2,2,3,1,1,1,2
2,12,4,0,2366,3,4,3,1,1,3,3,36,3,2,1,4,1,2,1,1
1,21,2,3,1835,1,3,3,2,1,2,1,25,3,2,2,3,1,2,1,2
4,24,4,1,3868,1,5,4,2,1,2,3,41,3,1,2,4,1,2,1,1
4,12,2,2,1768,1,3,3,3,1,2,1,24,3,1,1,2,1,1,1,1
3,10,4,0,781,1,5,4,3,1,4,4,63,3,3,2,3,1,2,1,1
2,18,2,2,1924,5,2,4,2,1,3,1,27,3,1,1,3,1,1,1,2
1,12,4,0,2121,1,3,4,3,1,2,2,30,3,2,2,3,1,1,1,1
1,12,2,3,701,1,3,4,4,1,2,1,40,3,2,1,2,1,1,1,1
2,12,2,5,639,1,3,4,3,1,2,3,30,3,2,1,3,1,1,1,2
2,12,4,1,1860,1,1,4,3,1,2,3,34,3,2,2,4,1,2,1,1
1,12,4,0,3499,1,3,3,2,2,2,1,29,3,2,2,3,1,1,1,2
2,48,2,0,8487,5,4,1,2,1,2,3,24,3,2,1,3,1,1,1,1
1,36,3,6,6887,1,3,4,3,1,3,2,29,2,2,1,3,1,2,1,2
4,15,2,2,2708,1,2,2,3,1,3,2,27,1,2,2,2,1,1,1,1
4,18,2,2,1984,1,3,4,3,1,4,4,47,1,3,2,3,1,1,1,1
4,60,2,3,10144,2,4,2,2,1,4,1,21,3,2,1,3,1,2,1,1
4,12,4,3,1240,5,5,4,2,1,2,1,38,3,2,2,3,1,2,1,1
4,27,3,1,8613,4,3,2,3,1,2,3,27,3,2,2,3,1,1,1,1
2,12,2,3,766,3,3,4,3,1,3,1,66,3,2,1,2,1,1,1,2
2,15,4,3,2728,5,4,4,3,3,2,1,35,1,2,3,3,1,2,1,1
3,12,2,3,1881,1,3,2,2,1,2,3,44,3,1,1,2,1,2,1,1
3,6,2,0,709,4,2,2,4,1,2,1,27,3,2,1,1,1,1,2,1
2,36,2,3,4795,1,2,4,2,1,1,4,30,3,2,1,4,1,2,1,1
1,27,2,3,3416,1,3,3,3,1,2,3,27,3,2,1,4,1,1,1,1
1,18,2,2,2462,1,3,2,3,1,2,3,22,3,2,1,3,1,1,1,2
4,21,4,2,2288,1,2,4,2,1,4,2,23,3,2,1,3,1,2,1,1
2,48,1,9,3566,2,4,4,3,1,2,3,30,3,2,1,3,1,1,1,1
1,6,4,0,860,1,5,1,2,1,4,4,39,3,2,2,3,1,2,1,1
4,12,4,0,682,2,4,4,2,1,3,3,51,3,2,2,3,1,2,1,1
1,36,4,2,5371,1,3,3,3,3,2,2,28,3,2,2,3,1,1,1,1
4,18,4,3,1582,4,5,4,3,1,4,3,46,3,2,2,3,1,1,1,1
4,6,2,3,1346,2,5,2,3,1,4,4,42,1,3,1,3,2,2,1,1
4,10,2,3,1924,1,3,1,3,1,4,2,38,3,2,1,3,1,2,2,1
3,36,2,3,5848,1,3,4,3,1,1,3,24,3,2,1,3,1,1,1,1
2,24,4,1,7758,4,5,2,2,1,4,4,29,3,1,1,3,1,1,1,1
2,24,3,9,6967,2,4,4,3,1,4,3,36,3,1,1,4,1,2,1,1
1,12,2,2,1282,1,3,2,2,1,4,3,20,3,1,1,3,1,1,1,2
1,9,4,5,1288,2,5,3,3,3,4,1,48,3,2,2,3,2,1,2,1
```

```
1,12,1,8,339,1,5,4,4,1,1,3,45,1,2,1,2,1,1,1,1
2,24,2,0,3512,2,4,2,3,1,3,3,38,1,2,2,3,1,2,1,1
4,6,4,3,1898,5,3,1,3,1,2,1,34,3,2,2,2,2,1,1,1
4,24,4,3,2872,2,5,3,3,1,4,1,36,3,2,1,3,2,2,1,1
4,18,4,0,1055,1,2,4,2,1,1,2,30,3,2,2,3,1,1,1,1
4,15,2,4,1262,3,4,4,3,1,3,2,36,3,2,2,3,1,2,1,1
2,10,2,0,7308,1,1,2,3,1,4,4,70,1,3,1,4,1,2,1,1
4,36,2,0,909,3,5,4,3,1,4,2,36,3,2,1,3,1,1,1,1
4,6,2,2,2978,3,3,1,3,1,2,3,32,3,2,1,3,1,2,1,1
1,18,2,2,1131,1,1,4,2,1,2,3,33,3,2,1,3,1,1,1,2
2,11,2,2,1577,4,2,4,2,1,1,1,20,3,2,1,3,1,1,1,1
4,24,2,2,3972,1,4,2,2,1,4,2,25,3,1,1,3,1,2,1,1
2,24,4,9,1935,1,5,4,1,1,4,1,31,3,2,2,3,1,2,1,2
1,15,0,0,950,1,5,4,3,1,3,3,33,3,1,2,3,2,1,1,2
4,12,2,2,763,1,3,4,2,1,1,1,26,3,2,1,3,1,2,1,1
2,24,3,2,2064,1,1,3,2,1,2,2,34,3,2,1,4,1,2,1,2
2,8,2,3,1414,1,3,4,3,3,2,1,33,3,2,1,3,1,1,2,1
1,21,3,6,3414,1,2,2,3,1,1,2,26,3,2,2,3,1,1,1,2
4,30,1,1,7485,5,1,4,2,1,1,1,53,1,2,1,4,1,2,1,2
1,12,2,2,2577,1,3,2,1,1,1,3,42,3,2,1,3,1,1,1,1
1,6,4,3,338,3,5,4,3,1,4,3,52,3,2,2,3,1,1,1,1
4,12,2,3,1963,1,4,4,3,1,2,3,31,3,1,2,4,2,2,1,1
1,21,4,0,571,1,5,4,3,1,4,1,65,3,2,2,3,1,1,1,1
4,36,3,9,9572,1,2,1,1,1,1,3,28,3,2,2,3,1,1,1,2
2,36,3,9,4455,1,3,2,1,1,2,1,30,2,2,2,4,1,2,1,2
1,21,1,0,1647,5,3,4,3,1,2,2,40,3,2,2,2,2,1,1,2
4,24,4,2,3777,4,3,4,3,1,4,1,50,3,2,1,3,1,2,1,1
2,18,4,0,884,1,5,4,3,1,4,3,36,1,2,1,3,2,2,1,2
4,15,4,3,1360,1,3,4,3,1,2,2,31,3,2,2,3,1,1,1,1
2,9,1,1,5129,1,5,2,2,1,4,4,74,1,3,1,4,2,2,1,2
2,16,4,0,1175,1,1,2,3,1,3,3,68,3,3,3,1,1,2,1,1
1,12,2,3,674,2,4,4,4,1,1,2,20,3,2,1,3,1,1,1,2
2,18,0,2,3244,1,3,1,2,1,4,3,33,1,2,2,3,1,2,1,1
4,24,2,9,4591,4,3,2,3,1,3,2,54,3,2,3,4,1,2,1,2
2,48,0,9,3844,2,4,4,3,1,4,4,34,3,3,1,2,2,1,1,2
2,27,2,9,3915,1,3,4,3,1,2,3,36,3,2,1,3,2,2,1,2
4,6,2,3,2108,1,4,2,4,1,2,1,29,3,1,1,3,1,1,1,1
2,45,2,3,3031,2,3,4,3,3,4,2,21,3,1,1,3,1,1,1,2
2,9,4,6,1501,1,5,2,2,1,3,3,34,3,2,2,4,1,2,1,2
4,6,4,3,1382,1,3,1,2,1,1,3,28,3,2,2,3,1,2,1,1
2,12,2,2,951,2,2,4,2,1,4,3,27,1,1,4,3,1,1,1,2
2,24,2,1,2760,5,5,4,3,1,4,4,36,1,3,1,3,1,2,1,1
2,18,3,2,4297,1,5,4,1,1,3,4,40,3,2,1,4,1,2,1,2
4,9,4,6,936,3,5,4,3,1,2,3,52,3,2,2,3,1,2,1,1
1,12,2,0,1168,1,3,4,4,1,3,1,27,3,2,1,2,1,1,1,1
4,27,3,9,5117,1,4,3,3,1,4,3,26,3,2,2,3,1,1,1,1
1,12,2,8,902,1,4,4,4,1,4,2,21,3,1,1,3,1,1,1,2
4,12,4,0,1495,1,5,4,3,1,1,1,38,3,2,2,2,2,1,1,1
1,30,4,1,10623,1,5,3,3,1,4,4,38,3,3,3,4,2,2,1,1
4,12,4,2,1935,1,5,4,3,1,4,1,43,3,2,3,3,1,2,1,1
2,12,4,4,1424,1,4,4,3,1,3,2,26,3,2,1,3,1,1,1,1
1,24,2,9,6568,1,3,2,4,1,2,3,21,2,2,1,2,1,1,1,1
4,12,2,1,1413,4,4,3,3,1,2,2,55,3,2,1,3,1,1,2,1
4,9,4,3,3074,5,3,1,3,1,2,1,33,3,2,2,3,2,1,1,1
4,36,2,3,3835,5,5,2,2,1,4,1,45,3,2,1,2,1,2,1,1
1,27,0,9,5293,1,1,2,3,1,4,2,50,2,2,2,3,1,2,1,2
```

```
3,30,3,9,1908,1,5,4,3,1,4,1,66,3,2,1,4,1,2,1,2
4,36,4,3,3342,5,5,4,3,1,2,3,51,3,2,1,3,1,2,1,1
2,6,4,8,932,5,4,1,2,1,3,2,39,3,2,2,2,1,1,1,1
1,18,0,9,3104,1,4,3,3,1,1,2,31,1,2,1,3,1,2,1,1
3,36,2,3,3913,1,3,2,3,1,2,1,23,3,2,1,3,1,2,1,1
1,24,2,2,3021,1,3,2,1,1,2,1,24,3,1,1,2,1,1,1,1
4,10,2,0,1364,1,3,2,2,1,4,3,64,3,2,1,3,1,2,1,1
2,12,2,3,625,1,2,4,4,3,1,1,26,1,2,1,2,1,1,1,1
1,12,2,6,1200,5,3,4,2,1,4,2,23,1,1,1,3,1,2,1,1
4,12,2,3,707,1,3,4,3,1,2,1,30,1,2,2,3,1,1,1,1
4,24,3,9,2978,5,3,4,3,1,4,1,32,3,2,2,3,2,2,1,1
4,15,2,1,4657,1,3,3,3,1,2,3,30,3,2,1,3,1,2,1,1
4,36,0,5,2613,1,3,4,3,1,2,3,27,3,2,2,3,1,1,1,1
2,48,2,3,10961,4,4,1,3,2,2,4,27,1,2,2,3,1,2,1,2
1,12,2,2,7865,1,5,4,3,1,4,4,53,3,3,1,4,1,2,1,2
4,9,2,3,1478,1,4,4,3,1,2,3,22,3,2,1,3,1,1,1,2
1,24,2,2,3149,1,2,4,3,1,1,4,22,1,3,1,3,1,1,1,1
3,36,2,3,4210,1,3,4,3,1,2,3,26,3,2,1,3,1,1,1,2
4,9,2,0,2507,3,5,2,3,1,4,4,51,3,3,1,2,1,1,1,1
4,12,2,3,2141,2,4,3,3,1,1,4,35,3,2,1,3,1,1,1,1
2,18,2,3,866,1,3,4,4,3,2,1,25,3,2,1,2,1,1,1,1
4,4,4,3,1544,1,4,2,3,1,1,1,42,3,2,3,2,2,1,1,1
1,24,2,3,1823,1,1,4,3,1,2,3,30,2,2,1,4,2,1,1,2
2,6,2,0,14555,5,1,1,3,1,2,2,23,3,2,1,1,1,2,1,2
2,21,2,9,2767,2,5,4,1,1,2,3,61,1,1,2,2,1,1,1,2
4,12,4,3,1291,1,3,4,2,1,2,2,35,3,2,2,3,1,1,1,1
1,30,2,3,2522,1,5,1,3,3,3,2,39,3,2,1,3,2,1,1,1
1,24,2,0,915,5,5,4,2,1,2,3,29,1,2,1,3,1,1,1,2
4,6,2,3,1595,1,4,3,3,1,2,2,51,3,2,1,3,2,1,1,1
1,48,0,1,4605,1,5,3,3,1,4,4,24,3,3,2,3,2,1,1,2
4,12,4,9,1185,1,3,3,2,1,2,1,27,3,2,2,3,1,1,1,1
4,12,1,8,3447,3,3,4,2,1,3,1,35,3,2,1,2,2,1,1,1
4,24,2,9,1258,1,4,4,3,1,1,1,25,3,2,1,3,1,2,1,1
4,12,4,3,717,1,5,4,3,1,4,1,52,3,2,3,3,1,1,1,1
4,6,0,0,1204,2,3,4,3,1,1,4,35,1,1,1,3,1,1,2,1
3,24,2,2,1925,1,3,2,3,1,2,1,26,3,2,1,3,1,1,1,1
4,18,2,3,433,1,1,3,2,2,4,1,22,3,1,1,3,1,1,1,2
1,6,4,0,666,4,4,3,2,1,4,1,39,3,2,2,2,1,2,1,1
3,12,2,2,2251,1,3,1,2,1,2,3,46,3,2,1,2,1,1,1,1
2,30,2,0,2150,1,3,4,2,3,2,4,24,1,2,1,3,1,1,1,2
4,24,3,2,4151,2,3,2,3,1,3,2,35,3,2,2,3,1,1,1,1
2,9,2,2,2030,5,4,2,3,1,1,3,24,3,2,1,3,1,2,1,1
2,60,3,3,7418,5,3,1,3,1,1,1,27,3,2,1,2,1,1,1,1
4,24,4,3,2684,1,3,4,3,1,2,1,35,3,2,2,2,1,1,1,1
1,12,1,3,2149,1,3,4,1,1,1,4,29,3,3,1,3,1,1,1,2
4,15,2,1,3812,2,2,1,2,1,4,3,23,3,2,1,3,1,2,1,1
4,11,4,3,1154,2,1,4,2,1,4,1,57,3,2,3,2,1,1,1,1
1,12,2,2,1657,1,3,2,3,1,2,1,27,3,2,1,3,1,1,1,1
1,24,2,3,1603,1,5,4,2,1,4,3,55,3,2,1,3,1,1,1,1
1,18,4,0,5302,1,5,2,3,1,4,4,36,3,3,3,4,1,2,1,1
4,12,4,6,2748,1,5,2,2,1,4,4,57,1,3,3,2,1,1,1,1
4,10,4,0,1231,1,5,3,3,1,4,1,32,3,2,2,2,2,1,2,1
2,15,2,3,802,1,5,4,3,1,3,3,37,3,2,1,3,2,1,1,2
4,36,4,9,6304,5,5,4,3,1,4,1,36,3,2,2,3,1,1,1,1
4,24,2,3,1533,1,2,4,2,1,3,3,38,2,2,1,3,1,2,1,1
1,14,2,0,8978,1,5,1,1,1,4,2,45,3,2,1,4,1,2,2,2
```

```
4,24,2,3,999,5,5,4,3,1,2,3,25,3,2,2,3,1,1,1,1
4,18,2,0,2662,5,4,4,3,1,3,2,32,3,2,1,3,1,1,2,1
4,12,4,2,1402,3,4,3,2,1,4,3,37,3,1,1,3,1,2,1,1
2,48,1,0,12169,5,1,4,3,2,4,4,36,3,3,1,4,1,2,1,1
2,48,2,3,3060,1,4,4,3,1,4,1,28,3,2,2,3,1,1,1,2
1,30,2,5,11998,1,2,1,1,1,1,4,34,3,2,1,2,1,2,1,2
4,9,2,3,2697,1,3,1,3,1,2,1,32,3,2,1,3,2,1,1,1
4,18,4,3,2404,1,3,2,2,1,2,3,26,3,2,2,3,1,1,1,1
1,12,2,2,1262,5,5,2,1,1,4,2,49,3,2,1,2,1,2,1,1
4,6,2,2,4611,1,2,1,2,1,4,2,32,3,2,1,3,1,1,1,2
4,24,2,3,1901,2,3,4,3,1,4,3,29,3,1,1,4,1,2,1,1
4,15,4,1,3368,4,5,3,3,1,4,4,23,3,1,2,3,1,2,1,1
4,12,2,2,1574,1,3,4,3,1,2,1,50,3,2,1,3,1,1,1,1
3,18,1,3,1445,5,4,4,3,1,4,3,49,1,2,1,2,1,1,1,1
4,15,4,2,1520,5,5,4,3,1,4,2,63,3,2,1,3,1,1,1,1
2,24,4,0,3878,2,2,4,1,1,2,3,37,3,2,1,3,1,2,1,1
1,47,2,0,10722,1,2,1,2,1,1,1,35,3,2,1,2,1,2,1,1
1,48,2,1,4788,1,4,4,3,1,3,2,26,3,2,1,3,2,1,1,1
2,48,3,10,7582,2,1,2,3,1,4,4,31,3,3,1,4,1,2,1,1
2,12,2,3,1092,1,3,4,2,3,4,1,49,3,2,2,3,1,2,1,1
1,24,3,3,1024,1,2,4,4,1,4,1,48,2,2,1,3,1,1,1,2
4,12,2,9,1076,1,3,2,4,1,2,1,26,3,2,1,3,1,2,2,1
2,36,2,1,9398,1,2,1,4,1,4,3,28,3,1,1,4,1,2,1,2
1,24,4,1,6419,1,5,2,2,1,4,4,44,3,3,2,4,2,2,1,1
3,42,4,1,4796,1,5,4,3,1,4,4,56,3,3,1,3,1,1,1,1
4,48,4,9,7629,5,5,4,1,1,2,3,46,1,2,2,4,2,1,1,1
2,48,2,2,9960,1,2,1,2,1,2,3,26,3,2,1,3,1,2,1,2
4,12,2,1,4675,5,2,1,2,1,4,3,20,3,1,1,3,1,1,1,1
4,10,2,0,1287,5,5,4,3,2,2,2,45,3,2,1,2,1,1,2,1
4,18,2,2,2515,1,3,3,3,1,4,1,43,3,2,1,3,1,2,1,1
2,21,4,2,2745,4,4,3,3,1,2,3,32,3,2,2,3,1,2,1,1
4,6,2,0,672,1,1,1,2,1,4,1,54,3,2,1,1,1,2,1,1
2,36,0,3,3804,1,3,4,2,1,1,3,42,3,2,1,3,1,2,1,2
3,24,4,0,1344,5,4,4,3,1,2,1,37,1,2,2,2,2,1,1,2
1,10,4,0,1038,1,4,4,3,2,3,2,49,3,2,2,3,1,2,1,1
4,48,4,0,10127,3,3,2,3,1,2,4,44,1,3,1,3,1,1,1,2
4,6,2,2,1543,4,3,4,1,1,2,1,33,3,2,1,3,1,1,1,1
4,30,2,1,4811,5,4,2,2,1,4,2,24,2,1,1,2,1,1,1,1
1,12,2,3,727,2,2,4,4,1,3,4,33,3,2,1,2,1,2,1,2
2,8,2,2,1237,1,3,3,2,1,4,1,24,3,2,1,3,1,1,1,2
2,9,2,0,276,1,3,4,4,1,4,1,22,3,1,1,2,1,1,1,1
2,48,2,10,5381,5,1,3,3,1,4,4,40,1,3,1,1,1,2,1,1
4,24,2,2,5511,2,3,4,3,1,1,3,25,2,2,1,3,1,1,1,1
3,24,2,2,3749,1,2,2,2,1,4,3,26,3,2,1,3,1,1,1,1
2,12,2,0,685,1,4,2,4,1,3,3,25,1,2,1,2,1,1,1,2
3,4,2,0,1494,5,2,1,3,1,2,1,29,3,2,1,2,2,1,2,1
1,36,1,2,2746,1,5,4,3,1,4,3,31,1,2,1,3,1,1,1,2
1,12,2,2,708,1,3,2,3,3,3,2,38,3,2,1,2,2,1,1,1
2,24,2,2,4351,5,3,1,2,1,4,2,48,3,2,1,2,1,2,1,1
4,12,4,6,701,1,3,4,3,1,2,3,32,3,2,2,3,1,1,1,1
1,15,3,2,3643,1,5,1,2,1,4,2,27,3,2,2,2,1,1,1,1
2,30,4,0,4249,1,1,4,4,1,2,3,28,3,2,2,4,1,1,1,2
1,24,2,3,1938,1,2,4,1,1,3,2,32,3,2,1,3,1,1,1,2
1,24,2,1,2910,1,4,2,3,1,1,4,34,3,3,1,4,1,2,1,1
1,18,2,2,2659,4,3,4,3,1,2,3,28,3,2,1,3,1,1,1,1
4,18,4,0,1028,1,3,4,2,1,3,1,36,3,2,2,3,1,1,1,1
```

```
1,8,4,0,3398,1,4,1,3,1,4,1,39,3,2,2,2,1,1,2,1
4,12,4,2,5801,5,5,2,3,1,4,2,49,3,1,1,3,1,2,1,1
4,24,2,0,1525,4,4,4,2,1,3,3,34,3,2,1,3,2,2,1,1
3,36,2,3,4473,1,5,4,3,1,2,3,31,3,2,1,3,1,1,1,1
2,6,2,3,1068,1,5,4,3,1,4,3,28,3,2,1,3,2,1,1,1
1,24,4,1,6615,1,1,2,3,1,4,4,75,3,3,2,4,1,2,1,1
4,18,4,6,1864,2,3,4,2,1,2,1,30,3,2,2,3,1,1,1,2
2,60,2,0,7408,2,2,4,2,1,2,2,24,3,2,1,4,1,1,1,2
4,48,4,1,11590,2,3,2,2,1,4,3,24,1,1,2,2,1,1,1,2
1,24,0,2,4110,1,5,3,3,1,4,4,23,1,1,2,3,2,1,1,2
1,6,4,2,3384,1,3,1,1,1,4,1,44,3,1,1,4,1,2,1,2
2,13,2,3,2101,1,2,2,2,3,4,2,23,3,2,1,2,1,1,1,1
1,15,2,4,1275,5,3,4,2,1,2,3,24,3,1,1,3,1,1,1,2
1,24,2,2,4169,1,3,4,3,1,4,2,28,3,2,1,3,1,1,1,1
2,10,2,2,1521,1,3,4,1,1,2,3,31,3,2,1,2,1,1,1,1
2,24,4,6,5743,1,2,2,2,1,4,4,24,3,3,2,3,1,2,1,1
1,21,2,2,3599,1,4,1,2,1,4,3,26,3,1,1,2,1,1,1,1
2,18,2,3,3213,3,2,1,4,1,3,1,25,3,1,1,3,1,1,1,1
2,18,2,9,4439,1,5,1,3,2,1,1,33,1,2,1,4,1,2,1,1
3,10,2,0,3949,1,2,1,3,3,1,2,37,3,2,1,2,2,1,1,1
4,15,4,3,1459,1,3,4,2,1,2,3,43,3,2,1,2,1,1,1,1
2,13,4,3,882,1,2,4,3,3,4,1,23,3,2,2,3,1,1,1,1
2,24,2,3,3758,3,1,1,2,1,4,4,23,3,1,1,1,1,1,1,1
4,6,3,9,1743,2,3,1,3,1,2,1,34,3,2,2,2,1,1,1,1
2,9,4,6,1136,4,5,4,3,1,3,4,32,3,3,2,3,2,1,1,2
4,9,2,4,1236,1,2,1,2,1,4,1,23,3,1,1,3,1,2,1,1
2,9,2,2,959,1,3,1,2,1,2,3,29,3,2,1,3,1,1,2,2
4,18,4,1,3229,5,1,2,3,1,4,4,38,3,2,1,4,1,2,1,1
1,12,0,3,6199,1,3,4,3,1,2,2,28,3,1,2,3,1,2,1,2
4,10,2,6,727,3,5,4,3,1,4,4,46,3,3,1,3,1,2,1,1
2,24,2,0,1246,1,2,4,3,1,2,1,23,2,2,1,2,1,1,1,2
4,12,4,3,2331,5,5,1,3,2,4,1,49,3,2,1,3,1,2,1,1
4,36,3,3,4463,1,3,4,3,1,2,3,26,3,2,2,4,1,2,1,2
4,12,2,3,776,1,3,4,4,1,2,1,28,3,2,1,3,1,1,1,1
1,30,2,2,2406,1,4,4,2,1,4,1,23,3,1,1,3,1,1,1,2
2,18,2,6,1239,5,3,4,3,1,4,4,61,3,3,1,3,1,1,1,1
3,12,2,3,3399,5,5,2,3,1,3,3,37,3,2,1,4,1,1,1,1
3,12,3,0,2247,1,3,2,2,1,2,3,36,2,2,2,3,1,2,1,1
4,6,2,2,1766,1,3,1,4,1,2,2,21,3,1,1,3,1,1,1,1
1,18,2,2,2473,1,1,4,3,1,1,3,25,3,2,1,1,1,1,1,2
4,12,2,9,1542,1,4,2,3,1,4,3,36,3,2,1,3,1,2,1,1
4,18,4,1,3850,1,4,3,3,1,1,3,27,3,2,2,3,1,1,1,1
1,18,2,2,3650,1,2,1,2,1,4,3,22,3,1,1,3,1,1,1,1
1,36,2,2,3446,1,5,4,3,1,2,3,42,3,2,1,3,2,1,1,2
2,18,2,2,3001,1,4,2,2,1,4,1,40,3,1,1,3,1,1,1,1
4,36,2,0,3079,5,3,4,3,1,4,1,36,3,2,1,3,1,1,1,1
4,18,4,3,6070,1,5,3,3,1,4,3,33,3,2,2,3,1,2,1,1
4,10,4,2,2146,1,2,1,2,1,3,1,23,3,1,2,3,1,1,1,1
4,60,4,0,13756,5,5,2,3,1,4,4,63,1,3,1,4,1,2,1,1
2,60,1,10,14782,2,5,3,2,1,4,4,60,1,3,2,4,1,2,1,2
1,48,1,9,7685,1,4,2,2,3,4,3,37,3,1,1,3,1,1,1,2
4,18,3,3,2320,1,1,2,4,1,3,1,34,3,2,2,3,1,1,1,1
4,7,3,3,846,5,5,3,3,1,4,4,36,3,3,1,3,1,1,1,1
2,36,2,0,14318,1,5,4,3,1,2,4,57,3,3,1,4,1,2,1,2
4,6,4,0,362,2,3,4,2,1,4,3,52,3,2,2,2,1,1,1,1
1,20,2,2,2212,5,4,4,3,1,4,3,39,3,2,1,3,1,2,1,1
```

```
2,18,2,1,12976,1,1,3,2,1,4,4,38,3,3,1,4,1,2,1,2
4,22,2,0,1283,5,4,4,2,1,4,2,25,3,1,1,3,1,1,1,1
3,12,2,0,1330,1,2,4,3,1,1,1,26,3,2,1,3,1,1,1,1
4,30,3,9,4272,2,3,2,3,1,2,2,26,3,2,2,2,1,1,1,1
4,18,4,3,2238,1,3,2,2,1,1,3,25,3,2,2,3,1,1,1,1
4,18,2,3,1126,5,2,4,2,1,2,1,21,3,1,1,3,1,2,1,1
2,18,4,2,7374,1,1,4,3,1,4,2,40,2,2,2,4,1,2,1,1
2,15,4,9,2326,3,3,2,3,1,4,3,27,1,2,1,3,1,1,1,1
4,9,2,9,1449,1,4,3,2,1,2,3,27,3,2,2,3,1,1,1,1
4,18,2,0,1820,1,3,2,4,1,2,2,30,3,2,1,4,1,2,1,1
2,12,2,2,983,4,2,1,2,1,4,1,19,3,1,1,2,1,1,1,1
1,36,2,0,3249,1,4,2,3,1,4,4,39,1,3,1,4,2,2,1,1
1,6,4,3,1957,1,4,1,2,1,4,3,31,3,2,1,3,1,1,1,1
4,9,4,2,2406,1,1,2,3,1,3,3,31,3,2,1,4,1,1,1,1
2,39,3,6,11760,2,4,2,3,1,3,4,32,3,1,1,3,1,2,1,1
1,12,2,2,2578,1,1,3,2,1,4,4,55,3,3,1,4,1,1,1,1
1,36,4,2,2348,1,3,3,4,1,2,2,46,3,2,2,3,1,2,1,1
2,12,2,0,1223,1,5,1,1,1,1,1,46,3,1,2,3,1,1,1,2
4,24,4,3,1516,4,3,4,2,1,1,1,43,3,2,2,2,1,1,1,1
4,18,2,3,1473,1,2,3,4,1,4,1,39,3,2,1,3,1,2,1,1
2,18,4,9,1887,5,3,4,4,1,4,1,28,1,2,2,3,1,1,1,1
4,24,3,9,8648,1,2,2,3,1,2,3,27,1,2,2,3,1,2,1,2
4,14,3,0,802,1,3,4,3,1,2,3,27,3,2,2,2,1,1,1,1
2,18,3,0,2899,5,5,4,3,1,4,3,43,3,2,1,3,2,1,1,1
2,24,2,3,2039,1,2,1,4,1,1,2,22,3,2,1,3,1,2,1,2
4,24,4,1,2197,5,4,4,3,1,4,3,43,3,2,2,3,2,2,1,1
1,15,2,3,1053,1,2,4,4,1,2,1,27,3,2,1,3,1,1,2,1
4,24,2,3,3235,3,5,3,1,1,2,3,26,3,2,1,4,1,2,1,1
3,12,4,0,939,3,4,4,4,1,2,1,28,3,2,3,3,1,2,1,2
2,24,2,3,1967,1,5,4,2,1,4,3,20,3,2,1,3,1,2,1,1
4,33,4,1,7253,1,4,3,3,1,2,3,35,3,2,2,4,1,2,1,1
4,12,4,9,2292,1,1,4,3,1,2,3,42,2,2,2,4,1,2,1,2
4,10,2,0,1597,3,3,3,3,1,2,4,40,3,1,1,2,2,1,2,1
1,24,2,0,1381,5,3,4,2,1,2,2,35,3,2,1,3,1,1,1,2
4,36,4,1,5842,1,5,2,3,1,2,2,35,3,2,2,3,2,2,1,1
1,12,2,0,2579,1,2,4,3,1,1,1,33,3,2,1,2,2,1,1,2
1,18,3,6,8471,5,3,1,2,1,2,3,23,3,1,2,3,1,2,1,1
4,21,2,0,2782,3,4,1,2,1,2,3,31,1,2,1,4,1,1,1,1
2,18,2,0,1042,5,3,4,2,1,2,2,33,3,2,1,3,1,1,1,2
4,15,2,0,3186,4,4,2,2,1,3,3,20,3,1,1,3,1,1,1,1
2,12,2,1,2028,5,3,4,3,1,2,3,30,3,2,1,3,1,1,1,1
2,12,4,0,958,1,4,2,3,1,3,1,47,3,2,2,2,2,1,1,1
4,21,3,2,1591,2,4,4,3,1,3,1,34,3,2,2,4,1,1,1,1
2,12,2,2,2762,5,5,1,2,1,2,2,25,1,2,1,3,1,2,1,2
2,18,2,1,2779,1,3,1,4,1,3,3,21,3,1,1,3,1,2,1,1
4,28,4,3,2743,1,5,4,3,1,2,3,29,3,2,2,3,1,1,1,1
4,18,4,3,1149,4,3,4,3,1,3,1,46,3,2,2,3,1,1,1,1
4,9,2,2,1313,1,5,1,3,1,4,3,20,3,2,1,3,1,1,1,1
1,18,4,5,1190,1,1,2,2,1,4,4,55,3,3,3,1,2,1,1,2
4,5,2,9,3448,1,4,1,3,1,4,1,74,3,2,1,2,1,1,1,1
2,24,2,10,11328,1,3,2,3,2,3,3,29,1,2,2,4,1,2,1,2
1,6,4,2,1872,1,1,4,3,1,4,4,36,3,3,3,4,1,2,1,1
4,24,4,5,2058,1,3,4,1,1,2,1,33,3,2,2,3,1,2,1,1
1,9,2,2,2136,1,3,3,3,1,2,1,25,3,2,1,3,1,1,1,1
2,12,2,3,1484,5,3,2,4,1,1,1,25,3,2,1,3,1,2,1,2
4,6,2,5,660,3,4,2,4,1,4,1,23,3,1,1,2,1,1,1,1
```

```
4,24,4,0,1287,4,5,4,2,1,4,1,37,3,2,2,3,1,2,1,1
1,42,4,5,3394,1,1,4,3,2,4,3,65,3,2,2,1,1,1,1,1
3,12,1,9,609,1,2,4,2,1,1,1,26,3,2,1,1,1,1,1,2
4,12,2,0,1884,1,5,4,3,1,4,3,39,3,2,1,4,1,2,1,1
1,12,2,2,1620,1,3,2,2,2,3,2,30,3,2,1,3,1,1,1,1
2,20,3,10,2629,1,3,2,3,1,3,3,29,1,2,2,3,1,2,1,1
4,12,2,6,719,1,5,4,3,1,4,3,41,1,2,1,2,2,1,1,2
2,48,4,2,5096,1,3,2,2,1,3,3,30,3,2,1,4,1,2,1,2
4,9,4,6,1244,5,5,4,2,1,4,2,41,3,1,2,2,1,1,1,1
1,36,2,0,1842,1,2,4,2,1,4,3,34,3,2,1,3,1,2,1,2
2,7,2,3,2576,1,3,2,3,3,2,1,35,3,2,1,3,1,1,1,1
3,12,2,2,1424,5,5,3,2,1,4,1,55,3,2,1,4,1,2,1,1
2,15,3,5,1512,4,3,3,4,1,3,2,61,2,2,2,3,1,1,1,2
4,36,4,1,11054,5,3,4,3,1,2,3,30,3,2,1,4,1,2,1,1
4,6,2,3,518,1,3,3,2,1,1,1,29,3,2,1,3,1,1,1,1
4,12,0,2,2759,1,5,2,3,1,4,2,34,3,2,2,3,1,1,1,1
4,24,2,1,2670,1,5,4,3,1,4,3,35,3,2,1,4,1,2,1,1
1,24,2,0,4817,1,4,2,3,2,3,2,31,3,2,1,3,1,2,1,2
4,24,2,1,2679,1,2,4,2,1,1,4,29,3,2,1,4,1,2,1,1
1,11,4,0,3905,1,3,2,3,1,2,1,36,3,1,2,3,2,1,1,1
1,12,2,1,3386,1,5,3,3,1,4,4,35,3,3,1,3,1,2,1,2
1,6,2,4,343,1,2,4,2,1,1,1,27,3,2,1,3,1,1,1,1
4,18,2,3,4594,1,2,3,3,1,2,3,32,3,2,1,3,1,2,1,1
1,36,2,2,3620,1,3,1,3,3,2,2,37,3,2,1,3,2,1,1,1
1,15,2,0,1721,1,2,2,3,1,3,1,36,3,2,1,3,1,1,1,1
2,12,2,2,3017,1,2,3,2,1,1,1,34,3,1,1,4,1,1,1,1
2,12,2,8,754,5,5,4,3,1,4,2,38,3,2,2,3,1,1,1,1
4,18,2,9,1950,1,4,4,3,1,1,3,34,2,2,2,3,1,2,1,1
1,24,2,1,2924,1,3,3,3,3,4,4,63,1,2,1,3,2,2,1,1
1,24,3,3,1659,1,2,4,2,1,2,3,29,3,1,1,2,1,2,1,2
4,48,3,3,7238,5,5,3,3,1,3,3,32,1,2,2,3,2,1,1,1
4,33,3,9,2764,1,3,2,2,1,2,3,26,3,2,2,3,1,2,1,1
4,24,3,1,4679,1,4,3,3,1,3,3,35,3,2,2,2,1,2,1,1
2,24,2,3,3092,2,2,3,4,1,2,3,22,3,1,1,3,1,2,1,2
1,6,2,6,448,1,2,4,2,1,4,2,23,3,2,1,3,1,1,1,2
1,9,2,0,654,1,3,4,3,1,3,3,28,3,2,1,2,1,1,1,2
4,6,2,8,1238,5,1,4,3,1,4,2,36,3,2,1,4,2,2,1,1
2,18,4,3,1245,1,3,4,4,1,2,3,33,3,2,1,3,1,1,1,2
1,18,0,2,3114,1,2,1,2,1,4,2,26,3,1,1,3,1,1,1,2
4,39,2,1,2569,3,3,4,3,1,4,3,24,3,2,1,3,1,1,1,1
3,24,2,3,5152,1,4,4,3,1,2,3,25,1,2,1,3,1,1,1,1
2,12,2,9,1037,2,4,3,3,1,4,1,39,3,2,1,2,1,1,1,1
1,15,4,2,1478,1,5,4,3,1,4,3,44,3,2,2,3,2,2,1,1
2,12,4,3,3573,1,3,1,2,1,1,1,23,3,2,1,2,1,1,1,1
2,24,2,0,1201,1,2,4,3,1,1,2,26,3,2,1,3,1,1,1,1
1,30,2,2,3622,4,5,4,2,1,4,2,57,3,1,2,3,1,2,1,1
4,15,3,2,960,4,4,3,2,1,2,2,30,3,2,2,3,1,1,1,1
4,12,4,0,1163,3,3,4,3,1,4,1,44,3,2,1,3,1,2,1,1
2,6,3,0,1209,1,1,4,3,1,4,2,47,3,2,1,4,1,2,1,2
4,12,2,3,3077,1,3,2,3,1,4,3,52,3,2,1,3,1,2,1,1
4,24,2,0,3757,1,5,4,2,2,4,4,62,3,3,1,3,1,2,1,1
4,10,2,0,1418,2,3,3,3,1,2,1,35,3,1,1,2,1,1,2,1
4,6,2,0,3518,1,3,2,3,3,3,2,26,3,1,1,3,1,1,1,1
4,12,4,3,1934,1,5,2,3,1,2,4,26,3,2,2,3,1,1,1,1
2,27,0,9,8318,1,5,2,2,1,4,4,42,3,3,2,4,1,2,1,2
4,6,4,3,1237,2,3,1,2,1,1,2,27,3,2,2,3,1,1,1,1
```

```
2,6,2,3,368,5,5,4,3,1,4,2,38,3,2,1,3,1,1,1,1
1,12,4,0,2122,1,3,3,3,1,2,1,39,3,1,2,2,2,1,2,1
1,24,2,2,2996,5,3,2,4,1,4,3,20,3,2,1,3,1,1,1,2
2,36,2,2,9034,2,2,4,3,2,1,4,29,3,1,1,4,1,2,1,2
4,24,4,2,1585,1,4,4,3,1,3,2,40,3,2,2,3,1,1,1,1
2,18,2,3,1301,1,5,4,4,3,2,1,32,3,2,1,2,1,1,1,1
3,6,4,0,1323,2,5,2,1,1,4,3,28,3,2,2,3,2,2,1,1
1,24,2,0,3123,1,2,4,2,1,1,2,27,3,2,1,3,1,1,1,2
1,36,2,1,5493,1,5,2,3,1,4,4,42,3,3,1,3,2,1,1,1
3,9,2,3,1126,2,5,2,1,1,4,1,49,3,2,1,3,1,1,1,1
2,24,4,3,1216,2,2,4,3,1,4,4,38,1,2,2,3,2,1,1,2
1,24,2,0,1207,1,2,4,2,1,4,2,24,3,1,1,3,1,1,1,2
4,10,2,0,1309,5,3,4,3,3,4,2,27,3,2,1,2,1,1,1,2
3,15,4,1,2360,3,3,2,3,1,2,3,36,3,2,1,3,1,2,1,1
2,15,1,0,6850,2,1,1,3,1,2,2,34,3,2,1,4,2,2,1,2
4,24,2,3,1413,1,3,4,4,1,2,2,28,3,2,1,3,1,1,1,1
4,39,2,1,8588,2,5,4,3,1,2,3,45,3,2,1,4,1,2,1,1
1,12,2,0,759,1,4,4,3,1,2,1,26,3,2,1,3,1,1,1,2
4,36,2,1,4686,1,3,2,3,1,2,4,32,3,3,1,4,1,2,1,1
3,15,2,9,2687,1,4,2,3,1,4,2,26,3,1,1,3,1,2,1,1
2,12,3,3,585,1,3,4,4,2,4,1,20,3,1,2,3,1,1,1,1
4,24,2,0,2255,5,2,4,3,1,1,2,54,3,2,1,3,1,1,1,1
1,6,4,0,609,1,4,4,2,1,3,2,37,3,2,2,3,1,1,2,1
1,6,4,0,1361,1,2,2,3,1,4,1,40,3,2,1,2,2,1,2,1
4,36,4,2,7127,1,2,2,2,1,4,2,23,3,1,2,3,1,2,1,2
1,6,2,0,1203,2,5,3,3,1,2,2,43,3,2,1,3,1,2,1,1
4,6,4,3,700,5,5,4,3,1,4,4,36,3,3,2,3,1,1,1,1
4,24,4,5,5507,1,5,3,3,1,4,4,44,3,3,2,3,1,1,1,1
1,18,2,3,3190,1,3,2,2,1,2,1,24,3,2,1,3,1,1,1,2
1,48,0,2,7119,1,3,3,3,1,4,4,53,3,3,2,3,2,1,1,2
4,24,2,1,3488,2,4,3,2,1,4,3,23,3,2,1,3,1,1,1,1
2,18,2,3,1113,1,3,4,2,3,4,1,26,3,2,1,2,2,1,1,1
2,26,2,1,7966,1,2,2,3,1,3,3,30,3,2,2,3,1,1,1,1
4,15,4,6,1532,2,3,4,2,1,3,3,31,3,2,1,3,1,1,1,1
4,4,4,3,1503,1,4,2,3,1,1,1,42,3,2,2,2,2,1,1,1
1,36,2,3,2302,1,3,4,1,1,4,3,31,3,1,1,3,1,1,1,2
1,6,2,0,662,1,2,3,3,1,4,1,41,3,2,1,2,2,2,1,1
2,36,2,6,2273,1,4,3,3,1,1,3,32,3,2,2,3,2,1,1,1
2,15,2,0,2631,2,3,2,2,1,4,3,28,3,1,2,3,1,2,1,2
4,12,3,1,1503,1,3,4,4,1,4,1,41,3,1,1,3,1,1,1,1
4,24,2,3,1311,2,4,4,4,1,3,2,26,3,2,1,3,1,2,1,1
4,24,2,3,3105,5,2,4,3,1,2,3,25,3,2,2,3,1,1,1,1
3,21,4,6,2319,1,2,2,1,1,1,3,33,3,1,1,3,1,1,1,2
1,6,2,0,1374,5,1,4,2,1,3,2,75,3,2,1,4,1,2,1,1
2,18,4,2,3612,1,5,3,2,1,4,2,37,3,2,1,3,1,2,1,1
1,48,2,0,7763,1,5,4,3,1,4,4,42,1,3,1,4,1,1,1,2
3,18,2,2,3049,1,2,1,2,1,1,2,45,2,2,1,2,1,1,1,1
2,12,2,3,1534,1,2,1,4,1,1,1,23,3,1,1,3,1,1,1,2
4,24,3,0,2032,1,5,4,3,1,4,4,60,3,3,2,3,1,2,1,1
1,30,2,2,6350,5,5,4,3,1,4,2,31,3,2,1,3,1,1,1,2
3,18,2,2,2864,1,3,2,3,1,1,1,34,3,2,1,2,2,1,1,2
4,12,4,0,1255,1,5,4,3,1,4,1,61,3,2,2,2,1,1,1,1
1,24,3,0,1333,1,1,4,3,1,2,1,43,3,3,2,3,2,1,1,2
4,24,4,0,2022,1,3,4,2,1,4,3,37,3,2,1,3,1,2,1,1
4,24,2,3,1552,1,4,3,3,1,1,3,32,1,2,1,3,2,1,1,1
1,12,1,3,626,1,3,4,2,1,4,1,24,1,2,1,2,1,1,1,2
```

```
4,48,4,1,8858,5,4,2,3,1,1,4,35,3,3,2,3,1,2,1,1
4,12,4,5,996,5,4,4,2,1,4,1,23,3,2,2,3,1,1,1,1
4,6,1,3,1750,3,5,2,3,1,4,2,45,1,2,1,2,2,1,1,1
1,48,2,3,6999,1,4,1,4,3,1,1,34,3,2,2,3,1,2,1,2
2,12,4,0,1995,2,2,4,3,1,1,3,27,3,2,1,3,1,1,1,1
2,9,2,6,1199,1,4,4,2,1,4,2,67,3,2,2,4,1,2,1,1
2,12,2,3,1331,1,2,2,3,1,1,3,22,2,2,1,3,1,1,1,2
2,18,0,0,2278,2,2,3,2,1,3,3,28,3,2,2,3,1,1,1,2
4,21,0,0,5003,5,3,1,2,1,4,2,29,1,2,2,3,1,2,1,2
1,24,1,2,3552,1,4,3,3,1,4,3,27,1,2,1,3,1,1,1,2
2,18,4,2,1928,1,2,2,3,1,2,1,31,3,2,2,2,1,1,1,2
1,24,2,1,2964,5,5,4,3,1,4,4,49,1,3,1,3,2,2,1,1
1,24,1,3,1546,1,4,4,3,3,4,3,24,1,1,1,2,1,1,1,2
3,6,3,3,683,1,2,2,2,1,1,2,29,1,2,1,3,1,1,1,1
2,36,2,0,12389,5,3,1,3,1,4,4,37,3,3,1,3,1,2,1,2
2,24,3,9,4712,5,3,4,3,1,2,2,37,1,2,2,4,1,2,1,1
2,24,3,3,1553,2,4,3,2,1,2,2,23,3,1,2,3,1,2,1,1
1,12,2,0,1372,1,4,2,1,1,3,3,36,3,2,1,3,1,1,1,2
4,24,4,3,2578,4,5,2,3,1,2,3,34,3,2,1,3,1,1,1,1
2,48,2,3,3979,5,4,4,3,1,1,3,41,3,2,2,3,2,2,1,1
1,48,2,3,6758,1,3,3,2,1,2,3,31,3,2,1,3,1,2,1,2
1,24,2,2,3234,1,2,4,2,1,4,1,23,3,1,1,2,1,2,1,2
4,30,4,3,5954,1,4,3,3,2,2,3,38,3,2,1,3,1,1,1,1
4,24,2,1,5433,5,1,2,2,1,4,2,26,3,1,1,4,1,2,1,1
1,15,2,9,806,1,3,4,2,1,4,2,22,3,2,1,2,1,1,1,1
2,9,2,3,1082,1,5,4,3,1,4,3,27,3,2,2,2,1,1,1,1
4,15,4,2,2788,1,4,2,2,2,3,3,24,1,2,2,3,1,1,1,1
2,12,2,3,2930,1,4,2,2,1,1,1,27,3,2,1,3,1,1,1,1
4,24,4,6,1927,5,3,3,2,1,2,3,33,3,2,2,3,1,2,1,1
2,36,4,0,2820,1,2,4,1,1,4,3,27,3,2,2,3,1,1,1,2
4,24,2,8,937,1,2,4,4,1,3,3,27,3,2,2,2,1,1,1,1
2,18,4,0,1056,1,5,3,3,3,3,1,30,1,2,2,3,1,1,1,2
2,12,4,0,3124,1,2,1,3,1,3,1,49,1,2,2,2,2,1,1,1
4,9,2,2,1388,1,3,4,2,1,2,1,26,3,1,1,3,1,1,1,1
2,36,2,5,2384,1,2,4,3,1,1,4,33,3,1,1,2,1,1,1,2
4,12,2,0,2133,5,5,4,2,1,4,4,52,3,3,1,4,1,2,1,1
1,18,2,2,2039,1,3,1,2,1,4,1,20,1,1,1,3,1,1,1,2
1,9,4,0,2799,1,3,2,3,1,2,1,36,3,1,2,3,2,1,1,1
1,12,2,2,1289,1,3,4,3,3,1,2,21,3,2,1,2,1,1,1,1
1,18,2,4,1217,1,3,4,4,1,3,1,47,3,2,1,2,1,2,1,2
1,12,4,2,2246,1,5,3,3,1,3,2,60,3,2,2,3,1,1,1,2
1,12,4,3,385,1,4,4,2,1,3,1,58,3,2,4,2,1,2,1,1
2,24,3,0,1965,5,3,4,2,1,4,3,42,3,1,2,3,1,2,1,1
4,21,2,9,1572,4,5,4,2,1,4,1,36,1,2,1,2,1,1,1,1
2,24,2,0,2718,1,3,3,2,1,4,2,20,3,1,1,2,1,2,1,2
1,24,1,10,1358,5,5,4,3,1,3,3,40,2,2,1,4,1,2,1,2
2,6,1,0,931,2,2,1,2,1,1,2,32,2,2,1,2,1,1,1,2
1,24,2,0,1442,1,4,4,2,1,4,3,23,3,1,2,3,1,1,1,2
2,24,0,9,4241,1,3,1,3,1,4,1,36,3,2,3,2,1,2,1,2
4,18,4,0,2775,1,4,2,3,1,2,2,31,1,2,2,3,1,1,1,2
4,24,3,9,3863,1,3,1,3,1,2,4,32,3,3,1,3,1,1,1,1
2,7,2,3,2329,1,2,1,2,3,1,1,45,3,2,1,3,1,1,1,1
2,9,2,2,918,1,3,4,2,1,1,2,30,3,2,1,3,1,1,1,2
2,24,1,6,1837,1,4,4,2,1,4,4,34,1,3,1,2,1,1,1,2
4,36,2,2,3349,1,3,4,2,1,2,3,28,3,2,1,4,1,2,1,2
3,10,2,2,1275,1,2,4,2,1,2,2,23,3,2,1,3,1,1,1,1
```

```
1,24,1,2,2828,3,3,4,3,1,4,1,22,2,2,1,3,1,2,1,1
4,24,4,9,4526,1,3,3,3,1,2,1,74,3,2,1,4,1,2,1,1
2,36,2,3,2671,2,3,4,2,2,4,4,50,3,3,1,3,1,1,1,2
4,18,2,3,2051,1,2,4,3,1,1,1,33,3,2,1,3,1,1,1,1
4,15,2,1,1300,5,5,4,3,1,4,4,45,1,3,1,3,2,1,1,1
1,12,2,4,741,2,1,4,2,1,3,2,22,3,2,1,3,1,1,1,2
3,10,2,0,1240,2,5,1,2,1,4,4,48,3,3,1,2,2,1,1,2
1,21,2,3,3357,4,2,4,2,1,2,3,29,1,2,1,3,1,1,1,1
1,24,1,1,3632,1,3,1,2,3,4,3,22,1,1,1,3,1,1,2,1
4,18,3,2,1808,1,4,4,2,1,1,1,22,3,2,1,3,1,1,1,2
2,48,0,9,12204,5,3,2,3,1,2,3,48,1,2,1,4,1,2,1,1
2,60,3,3,9157,5,3,2,3,1,2,4,27,3,3,1,4,1,1,1,1
1,6,4,0,3676,1,3,1,3,1,3,1,37,3,1,3,3,2,1,1,1
2,30,2,2,3441,2,3,2,2,2,4,3,21,3,1,1,3,1,1,1,2
4,12,2,0,640,1,3,4,1,1,2,1,49,3,2,1,2,1,1,1,1
2,21,4,9,3652,1,4,2,3,1,3,2,27,3,2,2,3,1,1,1,1
4,18,4,0,1530,1,3,3,3,1,2,2,32,1,2,2,3,1,1,1,2
4,48,2,9,3914,5,3,4,1,1,2,1,38,1,2,1,3,1,1,1,2
1,12,2,2,1858,1,2,4,2,1,1,3,22,3,1,1,3,1,1,1,1
1,18,2,3,2600,1,3,4,3,1,4,4,65,3,3,2,3,1,1,1,2
4,15,2,3,1979,5,5,4,3,1,2,3,35,3,2,1,3,1,1,1,1
3,6,2,2,2116,1,3,2,3,1,2,1,41,3,2,1,3,1,2,1,1
2,9,1,0,1437,2,4,2,3,1,3,4,29,3,2,1,3,1,1,1,2
4,42,4,2,4042,3,3,4,3,1,4,1,36,3,2,2,3,1,2,1,1
4,9,2,6,3832,5,5,1,3,1,4,1,64,3,2,1,2,1,1,1,1
1,24,2,3,3660,1,3,2,2,1,4,3,28,3,2,1,3,1,1,1,1
1,18,1,2,1553,1,3,4,3,1,3,3,44,1,2,1,3,1,1,1,2
2,15,2,3,1444,5,2,4,3,1,1,2,23,3,2,1,3,1,1,1,1
4,9,2,2,1980,1,2,2,2,2,2,3,19,3,1,2,3,1,1,1,2
2,24,2,0,1355,1,2,3,2,1,4,3,25,3,2,1,2,1,2,1,2
4,12,2,6,1393,1,5,4,3,1,4,2,47,1,2,3,3,2,2,1,1
4,24,2,3,1376,3,4,4,2,1,1,3,28,3,2,1,3,1,1,1,1
4,60,3,3,15653,1,4,2,3,1,4,3,21,3,2,2,3,1,2,1,1
4,12,2,3,1493,1,2,4,2,1,3,3,34,3,2,1,3,2,1,1,1
1,42,3,3,4370,1,4,3,3,1,2,2,26,1,2,2,3,2,2,1,2
1,18,2,6,750,1,1,4,2,1,1,1,27,3,2,1,1,1,1,1,2
2,15,2,5,1308,1,5,4,3,1,4,3,38,3,2,2,2,1,1,1,1
4,15,2,6,4623,2,3,3,3,1,2,2,40,3,2,1,4,1,2,1,2
4,24,4,3,1851,1,4,4,4,3,2,3,33,3,2,2,3,1,2,1,1
1,18,4,3,1880,1,4,4,4,1,1,2,32,3,2,2,4,1,2,1,1
4,36,3,9,7980,5,2,4,3,1,4,3,27,3,1,2,3,1,2,1,2
1,30,0,2,4583,1,3,2,1,3,2,1,32,3,2,2,3,1,1,1,1
4,12,2,0,1386,3,3,2,2,1,2,2,26,3,2,1,3,1,1,1,2
3,24,2,0,947,1,4,4,3,1,3,4,38,1,3,1,3,2,1,1,2
1,12,2,6,684,1,3,4,3,1,4,3,40,3,1,1,2,2,1,1,2
1,48,2,6,7476,1,4,4,3,1,1,4,50,3,3,1,4,1,2,1,1
2,12,2,2,1922,1,3,4,3,1,2,2,37,3,2,1,2,1,1,1,2
1,24,2,0,2303,1,5,4,3,2,1,1,45,3,2,1,3,1,1,1,2
2,36,3,0,8086,2,5,2,3,1,4,3,42,3,2,4,4,1,2,1,2
4,24,4,1,2346,1,4,4,3,1,3,3,35,3,2,2,3,1,2,1,1
1,14,2,0,3973,1,1,1,3,1,4,4,22,3,3,1,3,1,1,1,1
2,12,2,0,888,1,5,4,3,1,4,3,41,1,2,1,2,2,1,1,2
4,48,2,3,10222,5,4,4,3,1,3,3,37,2,2,1,3,1,2,1,1
2,30,0,9,4221,1,3,2,2,1,1,3,28,3,2,2,3,1,1,1,1
2,18,4,2,6361,1,5,2,3,1,1,4,41,3,2,1,3,1,2,1,1
3,12,2,3,1297,1,3,3,4,1,4,1,23,3,1,1,3,1,1,1,1
```

```
1,12,2,0,900,5,3,4,4,1,2,3,23,3,2,1,3,1,1,1,2
4,21,2,2,2241,1,5,4,3,1,2,1,50,3,2,2,3,1,1,1,1
2,6,3,2,1050,1,1,4,3,1,1,2,35,2,2,2,4,1,2,1,1
3,6,4,6,1047,1,3,2,2,1,4,2,50,3,2,1,2,1,1,1,1
4,24,4,10,6314,1,1,4,3,2,2,4,27,1,2,2,4,1,2,1,1
2,30,1,2,3496,4,3,4,3,1,2,3,34,2,2,1,3,2,2,1,1
4,48,1,9,3609,1,3,1,2,1,1,1,27,2,2,1,3,1,1,1,1
1,12,4,0,4843,1,5,3,3,2,4,2,43,3,1,2,3,1,2,1,2
3,30,4,3,3017,1,5,4,3,1,4,2,47,3,2,1,3,1,1,1,1
4,24,4,9,4139,2,3,3,3,1,3,2,27,3,2,2,2,1,2,1,1
4,36,2,9,5742,2,4,2,3,1,2,3,31,3,2,2,3,1,2,1,1
4,60,2,0,10366,1,5,2,3,1,4,2,42,3,2,1,4,1,2,1,1
4,6,4,0,2080,3,3,1,4,1,2,3,24,3,2,1,3,1,1,1,1
4,21,3,9,2580,3,2,4,3,1,2,1,41,1,2,1,2,2,1,1,2
4,30,4,3,4530,1,4,4,2,1,4,3,26,3,1,1,4,1,2,1,1
4,24,4,2,5150,1,5,4,3,1,4,3,33,3,2,1,3,1,2,1,1
2,72,2,3,5595,2,3,2,4,1,2,3,24,3,2,1,3,1,1,1,2
1,24,2,3,2384,1,5,4,3,1,4,1,64,1,1,1,2,1,1,1,1
4,18,2,3,1453,1,2,3,2,1,1,1,26,3,2,1,3,1,1,1,1
4,6,2,6,1538,1,2,1,2,1,2,4,56,3,2,1,3,1,1,1,1
4,12,2,3,2279,5,3,4,3,1,4,4,37,3,3,1,3,1,2,1,1
4,15,3,3,1478,1,3,4,4,1,3,1,33,1,2,2,3,1,1,1,1
4,24,4,3,5103,1,2,3,4,1,3,4,47,3,3,3,3,1,2,1,1
2,36,3,9,9857,2,4,1,3,1,3,2,31,3,2,2,2,2,2,1,1
4,60,2,0,6527,5,3,4,3,1,4,4,34,3,3,1,3,2,2,1,1
3,10,4,3,1347,5,4,4,3,1,2,2,27,3,2,2,3,1,2,1,1
2,36,3,0,2862,2,5,4,3,1,3,4,30,3,3,1,3,1,1,1,1
4,9,2,3,2753,2,5,3,3,2,4,3,35,3,2,1,3,1,2,1,1
1,12,2,0,3651,4,3,1,3,1,3,2,31,3,2,1,3,2,1,1,1
1,15,4,2,975,1,3,2,1,1,3,2,25,3,2,2,3,1,1,1,1
2,15,2,5,2631,2,3,3,2,1,2,1,25,3,2,1,2,1,1,1,1
2,24,2,3,2896,2,2,2,3,1,1,3,29,3,2,1,3,1,1,1,1
1,6,4,0,4716,5,2,1,3,1,3,1,44,3,2,2,2,2,1,1,1
4,24,2,3,2284,1,4,4,3,1,2,3,28,3,2,1,3,1,2,1,1
4,6,2,1,1236,3,3,2,3,1,4,2,50,3,1,1,3,1,1,1,1
2,12,2,3,1103,1,4,4,3,3,3,1,29,3,2,2,3,1,1,2,1
4,12,4,0,926,1,1,1,2,1,2,2,38,3,2,1,1,1,1,1,1
4,18,4,3,1800,1,3,4,3,1,2,3,24,3,2,2,3,1,1,1,1
3,15,2,6,1905,1,5,4,3,1,4,3,40,3,1,1,4,1,2,1,1
4,12,2,2,1123,3,3,4,2,1,4,3,29,3,1,1,2,1,1,1,2
1,48,4,1,6331,1,5,4,3,1,4,4,46,3,3,2,3,1,2,1,2
3,24,2,3,1377,2,5,4,2,1,2,4,47,3,3,1,3,1,2,1,1
2,30,3,9,2503,2,5,4,3,1,2,2,41,2,2,2,3,1,1,1,1
2,27,2,9,2528,1,2,4,2,1,1,2,32,3,2,1,3,2,2,1,1
4,15,2,0,5324,3,5,1,2,1,4,4,35,3,3,1,3,1,1,1,1
2,48,2,0,6560,2,4,3,3,1,2,2,24,3,2,1,3,1,1,1,2
2,12,0,2,2969,1,2,4,2,1,3,2,25,3,1,2,3,1,1,1,2
2,9,2,3,1206,1,5,4,2,1,4,1,25,3,2,1,3,1,1,1,1
2,9,2,3,2118,1,3,2,3,1,2,1,37,3,2,1,2,2,1,1,1
4,18,4,3,629,3,5,4,3,1,3,2,32,1,2,2,4,1,2,1,1
1,6,1,6,1198,1,5,4,2,1,4,4,35,3,3,1,3,1,1,1,2
4,21,2,1,2476,5,5,4,3,1,4,1,46,3,2,1,4,1,2,1,1
1,9,4,3,1138,1,3,4,3,1,4,1,25,3,2,2,2,1,1,1,1
2,60,2,0,14027,1,4,4,3,1,2,4,27,3,2,1,4,1,2,1,2
4,30,4,1,7596,5,5,1,3,1,4,3,63,3,2,2,3,1,1,1,1
4,30,4,3,3077,5,5,3,3,1,2,3,40,3,2,2,3,2,2,1,1
```

```
4,18,2,3,1505,1,3,4,3,1,2,4,32,3,3,1,4,1,2,1,1
3,24,4,3,3148,5,3,3,3,1,2,3,31,3,2,2,3,1,2,1,1
2,20,0,1,6148,2,5,3,4,1,4,3,31,1,2,2,3,1,2,1,1
3,9,0,3,1337,1,2,4,3,1,2,3,34,3,2,2,4,1,2,1,2
2,6,1,6,433,4,2,4,2,1,2,2,24,1,1,1,3,2,1,1,2
1,12,2,0,1228,1,3,4,2,1,2,1,24,3,2,1,2,1,1,1,2
2,9,2,3,790,3,3,4,2,1,3,1,66,3,2,1,2,1,1,1,1
4,27,2,0,2570,1,3,3,2,1,3,1,21,3,1,1,3,1,1,1,2
4,6,4,0,250,4,3,2,2,1,2,1,41,1,2,2,2,1,1,1,1
4,15,4,3,1316,3,3,2,4,1,2,2,47,3,2,2,2,1,1,1,1
1,18,2,3,1882,1,3,4,2,1,4,3,25,1,1,2,3,1,1,1,2
2,48,1,9,6416,1,5,4,2,1,3,4,59,3,1,1,3,1,1,1,2
3,24,4,9,1275,4,3,2,1,1,4,1,36,3,2,2,3,1,2,1,1
2,24,3,3,6403,1,2,1,3,1,2,3,33,3,2,1,3,1,1,1,1
1,24,2,3,1987,1,3,2,3,1,4,1,21,3,1,1,2,2,1,1,2
2,8,2,3,760,1,4,4,2,3,2,1,44,3,2,1,2,1,1,1,1
4,24,2,1,2603,4,3,2,2,1,4,3,28,3,1,1,3,1,2,1,1
4,4,4,0,3380,1,4,1,2,1,1,1,37,3,2,1,3,2,1,1,1
2,36,1,4,3990,5,2,3,2,1,2,4,29,1,2,1,1,1,1,1,1
2,24,2,1,11560,1,3,1,2,1,4,3,23,3,1,2,4,1,1,1,2
1,18,2,0,4380,2,3,3,3,1,4,3,35,3,2,1,2,2,2,1,1
4,6,4,0,6761,1,4,1,3,1,3,4,45,3,2,2,4,2,2,1,1
2,30,0,9,4280,2,3,4,2,1,4,3,26,3,1,2,2,1,1,1,2
1,24,1,0,2325,2,4,2,3,1,3,3,32,1,2,1,3,1,1,1,1
2,10,1,3,1048,1,3,4,3,1,4,1,23,2,2,1,2,1,1,1,1
4,21,2,3,3160,5,5,4,3,1,3,2,41,3,2,1,3,1,2,1,1
1,24,1,2,2483,3,3,4,3,1,4,1,22,2,2,1,3,1,2,1,1
1,39,4,2,14179,5,4,4,3,1,4,2,30,3,2,2,4,1,2,1,1
1,13,4,9,1797,1,2,3,3,1,1,2,28,1,2,2,2,1,1,1,1
1,15,2,0,2511,1,1,1,2,1,4,3,23,3,1,1,3,1,1,1,1
1,12,2,0,1274,1,2,3,2,1,1,1,37,3,2,1,2,1,1,1,2
4,21,2,1,5248,5,3,1,3,1,3,3,26,3,2,1,3,1,1,1,1
4,15,2,1,3029,1,4,2,3,1,2,3,33,3,2,1,3,1,1,1,1
1,6,2,2,428,1,5,2,2,1,1,2,49,1,2,1,3,1,2,1,1
1,18,2,0,976,1,2,1,2,1,2,3,23,3,2,1,2,1,1,1,2
2,12,2,9,841,2,4,2,2,1,4,1,23,3,1,1,2,1,1,1,1
4,30,4,3,5771,1,4,4,2,1,2,3,25,3,2,2,3,1,1,1,1
4,12,3,5,1555,4,5,4,3,1,4,4,55,3,3,2,3,2,1,1,2
1,24,2,0,1285,5,4,4,2,1,4,4,32,3,1,1,3,1,1,1,2
3,6,4,0,1299,1,3,1,3,1,1,1,74,3,2,3,1,2,1,2,1
3,15,4,3,1271,5,3,3,3,1,4,4,39,3,3,2,3,1,2,1,2
4,24,2,0,1393,1,3,2,3,3,2,1,31,3,2,1,3,1,2,1,1
1,12,4,0,691,1,5,4,3,1,3,2,35,3,2,2,3,1,1,1,2
4,15,4,0,5045,5,5,1,2,1,4,3,59,3,2,1,3,1,2,1,1
1,18,4,2,2124,1,3,4,2,1,4,1,24,3,1,2,3,1,1,1,2
1,12,2,3,2214,1,3,4,3,1,3,2,24,3,2,1,2,1,1,1,1
4,21,4,0,12680,5,5,4,3,1,4,4,30,3,3,1,4,1,2,1,2
4,24,4,0,2463,2,4,4,4,1,3,2,27,3,2,2,3,1,2,1,1
2,12,2,3,1155,1,5,3,4,3,3,1,40,1,2,2,2,1,1,1,1
1,30,2,2,3108,1,2,2,1,1,4,2,31,3,2,1,2,1,1,1,2
4,10,2,1,2901,5,2,1,2,1,4,1,31,3,1,1,3,1,1,1,1
2,12,4,2,3617,1,5,1,3,1,4,3,28,3,1,3,3,1,2,1,1
4,12,4,3,1655,1,5,2,3,1,4,1,63,3,2,2,2,1,2,1,1
1,24,2,1,2812,5,5,2,2,1,4,1,26,3,1,1,3,1,1,1,1
1,36,4,6,8065,1,3,3,2,1,2,4,25,3,2,2,4,1,2,1,2
4,21,4,1,3275,1,5,1,3,1,4,3,36,3,2,1,4,1,2,1,1
```

```
4,24,4,3,2223,2,5,4,3,1,4,2,52,1,2,2,3,1,1,1,1
3,12,4,0,1480,3,1,2,3,1,4,4,66,1,3,3,1,1,1,1,1
1,24,2,0,1371,5,3,4,2,1,4,1,25,3,1,1,3,1,1,1,2
4,36,4,0,3535,1,4,4,3,1,4,3,37,3,2,2,3,1,2,1,1
1,18,2,3,3509,1,4,4,2,3,1,1,25,3,2,1,3,1,1,1,1
4,36,4,1,5711,4,5,4,3,1,2,3,38,3,2,2,4,1,2,1,1
2,18,2,5,3872,1,1,2,2,1,4,3,67,3,2,1,3,1,2,1,1
2,39,4,3,4933,1,4,2,3,3,2,1,25,3,2,2,3,1,1,1,2
4,24,4,0,1940,4,5,4,3,1,4,1,60,3,2,1,3,1,2,1,1
2,12,0,8,1410,1,3,2,3,1,2,1,31,3,2,1,2,1,2,1,1
2,12,2,0,836,2,2,4,2,1,2,2,23,1,2,1,2,1,1,1,2
2,20,2,1,6468,5,1,1,1,1,4,1,60,3,2,1,4,1,2,1,1
2,18,2,9,1941,4,3,4,3,1,2,2,35,3,2,1,2,1,2,1,1
4,22,2,3,2675,3,5,3,3,1,4,3,40,3,2,1,3,1,1,1,1
4,48,4,1,2751,5,5,4,3,1,3,3,38,3,2,2,3,2,2,1,1
2,48,3,6,6224,1,5,4,3,1,4,4,50,3,3,1,3,1,1,1,2
1,40,4,6,5998,1,3,4,3,1,3,4,27,1,2,1,3,1,2,1,2
2,21,2,9,1188,1,5,2,2,1,4,2,39,3,2,1,3,2,1,1,2
4,24,2,1,6313,5,5,3,3,1,4,3,41,3,2,1,4,2,2,1,1
4,6,4,2,1221,5,3,1,4,1,2,2,27,3,2,2,3,1,1,1,1
3,24,2,2,2892,1,5,3,1,1,4,4,51,3,3,1,3,1,1,1,1
4,24,2,2,3062,3,5,4,3,1,3,4,32,3,1,1,3,1,2,1,1
4,9,2,2,2301,2,2,2,2,1,4,2,22,3,1,1,3,1,1,1,1
1,18,2,1,7511,5,5,1,3,1,4,2,51,3,3,1,3,2,2,1,2
4,12,4,2,1258,1,2,2,2,1,4,2,22,3,1,2,2,1,1,1,1
4,24,3,0,717,5,5,4,4,1,4,3,54,3,2,2,3,1,2,1,1
2,9,2,0,1549,5,2,4,3,1,2,1,35,3,2,1,1,1,1,1,1
4,24,4,6,1597,1,5,4,3,1,4,4,54,3,3,2,3,2,1,1,1
2,18,4,3,1795,1,5,3,2,3,4,1,48,1,1,2,2,1,2,1,1
1,20,4,2,4272,1,5,1,2,1,4,2,24,3,2,2,3,1,1,1,1
4,12,4,3,976,5,5,4,3,1,4,3,35,3,2,2,3,1,1,1,1
2,12,2,0,7472,5,1,1,2,1,2,1,24,3,1,1,1,1,1,1,1
1,36,2,0,9271,1,4,2,3,1,1,3,24,3,2,1,3,1,2,1,2
2,6,2,3,590,1,2,3,4,1,3,1,26,3,2,1,2,1,1,2,1
4,12,4,3,930,5,5,4,3,1,4,1,65,3,2,4,3,1,1,1,1
2,42,1,1,9283,1,1,1,3,1,2,4,55,1,3,1,4,1,2,1,1
2,15,0,0,1778,1,2,2,2,1,1,1,26,3,1,2,1,1,1,1,2
2,8,2,9,907,1,2,3,4,1,2,1,26,3,2,1,3,1,2,1,1
2,6,2,3,484,1,4,3,4,3,3,1,28,1,2,1,2,1,1,1,1
1,36,4,1,9629,1,4,4,3,1,4,3,24,3,2,2,3,1,2,1,2
1,48,2,4,3051,1,3,3,3,1,4,3,54,3,2,1,3,1,1,1,2
1,48,2,0,3931,1,4,4,3,1,4,4,46,3,3,1,3,2,1,1,2
2,36,3,0,7432,1,3,2,2,1,2,2,54,3,1,1,3,1,1,1,1
4,6,2,4,1338,3,3,1,1,1,4,1,62,3,2,1,3,1,1,1,1
4,6,4,3,1554,1,4,1,2,1,2,3,24,3,1,2,3,1,2,1,1
1,36,2,10,15857,1,1,2,1,2,3,3,43,3,2,1,4,1,1,1,1
1,18,2,3,1345,1,3,4,4,1,3,1,26,1,2,1,3,1,1,1,2
4,12,2,0,1101,1,3,3,4,1,2,1,27,3,2,2,3,1,2,1,1
3,12,2,3,3016,1,3,3,4,1,1,3,24,3,2,1,3,1,1,1,1
1,36,2,2,2712,1,5,2,3,1,2,2,41,1,2,1,3,2,1,1,2
1,8,4,0,731,1,5,4,3,1,4,1,47,3,2,2,2,1,1,1,1
4,18,4,2,3780,1,2,3,1,1,2,3,35,3,2,2,4,1,2,1,1
1,21,4,0,1602,1,5,4,4,1,3,3,30,3,2,2,3,1,2,1,1
1,18,4,0,3966,1,5,1,2,1,4,1,33,1,1,3,3,1,2,1,2
4,18,0,9,4165,1,3,2,3,1,2,3,36,2,2,2,3,2,1,1,2
1,36,2,1,8335,5,5,3,3,1,4,4,47,3,3,1,3,1,1,1,2
```

```
2,48,3,9,6681,5,3,4,3,1,4,4,38,3,3,1,3,2,2,1,1
4,24,3,9,2375,3,3,4,3,1,2,3,44,3,2,2,3,2,2,1,1
1,18,2,0,1216,1,2,4,2,1,3,3,23,3,1,1,3,1,2,1,2
1,45,0,9,11816,1,5,2,3,1,4,3,29,3,1,2,3,1,1,1,2
2,24,2,3,5084,5,5,2,2,1,4,3,42,3,2,1,3,1,2,1,1
3,15,2,3,2327,1,2,2,2,1,3,1,25,3,2,1,2,1,1,1,2
1,12,0,0,1082,1,3,4,3,1,4,3,48,1,2,2,3,1,1,1,2
4,12,2,3,886,5,3,4,2,1,2,3,21,3,2,1,3,1,1,1,1
4,4,2,2,601,1,2,1,2,1,3,1,23,3,1,1,2,2,1,1,1
1,24,4,1,2957,1,5,4,3,1,4,2,63,3,2,2,3,1,2,1,1
4,24,4,3,2611,1,5,4,4,2,3,1,46,3,2,2,3,1,1,1,1
1,36,2,2,5179,1,4,4,3,1,2,2,29,3,2,1,3,1,1,1,2
4,21,3,1,2993,1,3,3,3,1,2,1,28,2,2,2,2,1,1,1,1
4,18,2,5,1943,1,2,4,2,1,4,1,23,3,2,1,3,1,1,1,2
4,24,1,9,1559,1,4,4,3,1,4,3,50,1,2,1,3,1,2,1,1
4,18,2,2,3422,1,5,4,3,1,4,2,47,1,2,3,3,2,2,1,1
2,21,2,2,3976,5,4,2,3,1,3,3,35,3,2,1,3,1,2,1,1
4,18,2,0,6761,5,3,2,3,1,4,3,68,3,1,2,3,1,1,1,2
4,24,2,0,1249,1,2,4,4,1,2,1,28,3,2,1,3,1,1,1,1
1,9,2,3,1364,1,4,3,3,1,4,1,59,3,2,1,3,1,1,1,1
1,12,2,3,709,1,5,4,3,1,4,1,57,2,2,1,2,1,1,1,2
1,20,4,0,2235,1,3,4,4,3,2,2,33,1,1,2,3,1,1,2,2
4,24,4,1,4042,5,4,3,3,1,4,2,43,3,2,2,3,1,2,1,1
4,15,4,3,1471,1,3,4,3,1,4,4,35,3,3,2,3,1,2,1,1
1,18,1,0,1442,1,4,4,3,1,4,4,32,3,3,2,2,2,1,1,2
4,36,3,0,10875,1,5,2,3,1,2,3,45,3,2,2,3,2,2,1,1
4,24,2,0,1474,2,2,4,4,1,3,1,33,3,2,1,3,1,2,1,1
4,10,2,8,894,5,4,4,2,1,3,2,40,3,2,1,3,1,2,1,1
4,15,4,2,3343,1,3,4,3,1,2,4,28,3,3,1,3,1,2,1,1
1,15,2,0,3959,1,3,3,2,1,2,2,29,3,2,1,3,1,2,1,2
4,9,2,0,3577,2,3,1,3,3,2,1,26,3,1,1,3,2,1,2,1
4,24,4,1,5804,4,3,4,3,1,2,1,27,3,2,2,3,1,1,1,1
4,18,3,9,2169,1,3,4,4,1,2,3,28,3,2,1,3,1,2,1,2
1,24,2,3,2439,1,2,4,2,1,4,1,35,3,2,1,3,1,2,1,2
4,27,4,2,4526,4,2,4,3,1,2,1,32,2,2,2,2,2,2,1,1
4,10,2,2,2210,1,3,2,3,1,2,1,25,1,1,1,2,1,1,1,2
4,15,2,2,2221,3,3,2,2,1,4,3,20,3,1,1,3,1,1,1,1
1,18,2,3,2389,1,2,4,2,1,1,3,27,2,2,1,3,1,1,1,1
4,12,4,2,3331,1,5,2,3,1,4,2,42,2,2,1,3,1,1,1,1
4,36,2,9,7409,5,5,3,3,1,2,2,37,3,2,2,3,1,1,1,1
1,12,2,2,652,1,5,4,2,1,4,2,24,3,1,1,3,1,1,1,1
4,36,3,2,7678,3,4,2,2,1,4,3,40,3,2,2,3,1,2,1,1
3,6,4,0,1343,1,5,1,3,1,4,1,46,3,2,2,3,2,1,2,1
1,24,4,9,1382,2,4,4,3,1,1,1,26,3,2,2,3,1,2,1,1
4,15,2,4,874,5,2,4,2,1,1,1,24,3,2,1,3,1,1,1,1
1,12,2,2,3590,1,3,2,3,2,2,2,29,3,2,1,2,2,1,1,1
2,11,4,0,1322,4,3,4,2,1,4,3,40,3,2,2,3,1,1,1,1
1,18,1,3,1940,1,2,3,3,2,4,4,36,1,3,1,4,1,2,1,1
4,36,2,3,3595,1,5,4,3,1,2,3,28,3,2,1,3,1,1,1,1
1,9,2,0,1422,1,2,3,3,1,2,4,27,3,3,1,4,1,2,1,2
4,30,4,3,6742,5,4,2,3,1,3,2,36,3,2,2,3,1,1,1,1
4,24,2,1,7814,1,4,3,3,1,3,3,38,3,2,1,4,1,2,1,1
4,24,2,1,9277,5,3,2,1,1,4,4,48,3,3,1,3,1,2,1,1
2,30,4,0,2181,5,5,4,3,1,4,1,36,3,2,2,3,1,1,1,1
4,18,4,3,1098,1,1,4,2,1,4,3,65,3,2,2,1,1,1,1,1
2,24,2,2,4057,1,4,3,1,1,3,3,43,3,2,1,3,1,2,1,2
```

```
1,12,2,6,795,1,2,4,2,1,4,2,53,3,2,1,3,1,1,1,2
2,24,4,9,2825,5,4,4,3,1,3,4,34,3,2,2,3,2,2,1,1
2,48,2,9,15672,1,3,2,3,1,2,3,23,3,2,1,3,1,2,1,2
4,36,4,0,6614,1,5,4,3,1,4,3,34,3,2,2,4,1,2,1,1
4,28,1,1,7824,5,2,3,3,3,4,1,40,1,1,2,3,2,2,1,1
1,27,4,9,2442,1,5,4,3,1,4,3,43,2,2,4,4,2,2,1,1
4,15,4,3,1829,1,5,4,3,1,4,3,46,3,2,2,3,1,2,1,1
1,12,4,0,2171,1,3,4,3,1,4,2,38,1,2,2,2,1,1,2,1
2,36,4,1,5800,1,3,3,3,1,4,3,34,3,2,2,3,1,2,1,1
4,18,4,3,1169,5,3,4,3,1,3,2,29,3,2,2,3,1,2,1,1
4,36,3,1,8947,5,4,3,3,1,2,3,31,2,2,1,4,2,2,1,1
1,21,2,3,2606,1,2,4,2,1,4,2,28,3,1,1,4,1,2,1,1
4,12,4,2,1592,4,4,3,2,1,2,2,35,3,2,1,3,1,1,2,1
4,15,2,2,2186,5,4,1,2,1,4,1,33,1,1,1,2,1,1,1,1
1,18,2,2,4153,1,3,2,3,2,3,3,42,3,2,1,3,1,1,1,2
1,16,4,0,2625,1,5,2,3,3,4,2,43,1,1,1,3,1,2,1,2
4,20,4,0,3485,5,2,2,1,1,4,1,44,3,2,2,3,1,2,1,1
4,36,4,1,10477,5,5,2,3,1,4,4,42,3,3,2,3,1,1,1,1
4,15,2,3,1386,5,3,4,4,1,2,1,40,3,1,1,3,1,2,1,1
4,24,2,3,1278,1,5,4,3,1,1,1,36,3,2,1,4,1,2,1,1
1,12,2,3,1107,1,3,2,3,1,2,1,20,3,1,1,4,2,2,1,1
1,21,2,0,3763,5,4,2,3,2,2,1,24,3,2,1,2,1,1,2,1
2,36,2,6,3711,5,3,2,4,1,2,3,27,3,2,1,3,1,1,1,1
4,15,3,1,3594,1,2,1,2,1,2,2,46,3,2,2,2,1,1,1,1
2,9,2,0,3195,5,3,1,2,1,2,1,33,3,2,1,2,1,1,1,1
4,36,3,3,4454,1,3,4,2,1,4,1,34,3,2,2,3,1,1,1,1
2,24,4,2,4736,1,2,2,2,1,4,3,25,1,2,1,2,1,1,1,2
2,30,2,3,2991,5,5,2,2,1,4,3,25,3,2,1,3,1,1,1,1
4,11,2,9,2142,4,5,1,1,1,2,1,28,3,2,1,3,1,2,1,1
1,24,1,9,3161,1,3,4,3,1,2,2,31,3,1,1,3,1,2,1,2
2,48,0,10,18424,1,3,1,2,1,2,2,32,1,2,1,4,1,2,2,2
4,10,2,1,2848,2,3,1,3,2,2,1,32,3,2,1,3,2,1,1,1
1,6,2,0,14896,1,5,1,3,1,4,4,68,1,2,1,4,1,2,1,2
1,24,2,2,2359,2,1,1,1,1,1,2,33,3,2,1,3,1,1,1,2
1,24,2,2,3345,1,5,4,3,1,2,2,39,3,1,1,4,1,2,1,2
4,18,4,2,1817,1,3,4,2,1,2,4,28,3,2,2,3,1,1,1,1
4,48,3,3,12749,3,4,4,3,1,1,3,37,3,2,1,4,1,2,1,1
1,9,2,3,1366,1,2,3,2,1,4,2,22,3,1,1,3,1,1,1,2
2,12,2,0,2002,1,4,3,3,1,4,2,30,3,1,1,3,2,2,1,1
1,24,1,2,6872,1,2,2,1,1,1,2,55,1,2,1,3,1,2,1,2
1,12,1,0,697,1,2,4,3,1,2,3,46,1,2,2,3,1,2,1,2
1,18,4,2,1049,1,2,4,2,1,4,2,21,3,1,1,3,1,1,1,1
1,48,2,1,10297,1,4,4,3,1,4,4,39,2,3,3,3,2,2,1,2
4,30,2,3,1867,5,5,4,3,1,4,3,58,3,2,1,3,1,2,1,1
1,12,3,0,1344,1,3,4,3,1,2,1,43,3,2,2,2,2,1,1,1
1,24,2,2,1747,1,2,4,3,2,1,2,24,3,2,1,2,1,1,2,1
2,9,2,3,1670,1,2,4,2,1,2,3,22,3,2,1,3,1,2,1,2
4,9,4,0,1224,1,3,3,3,1,1,1,30,3,2,2,3,1,1,1,1
4,12,4,3,522,3,5,4,3,1,4,2,42,3,2,2,3,2,2,1,1
1,12,2,3,1498,1,3,4,2,1,1,3,23,1,2,1,3,1,1,1,1
2,30,3,3,1919,2,2,4,3,1,3,4,30,2,2,2,4,1,1,1,2
3,9,2,3,745,1,3,3,2,1,2,1,28,3,2,1,2,1,1,1,2
2,6,2,3,2063,1,2,4,4,1,3,3,30,3,1,1,4,1,2,1,1
2,60,2,6,6288,1,3,4,3,1,4,4,42,3,3,1,3,1,1,1,2
4,24,4,1,6842,5,3,2,3,1,4,2,46,3,2,2,4,2,2,1,1
4,12,2,0,3527,5,2,2,3,1,3,2,45,3,2,1,4,2,2,1,1
```

```
4,10,2,0,1546,1,3,3,3,1,2,1,31,3,2,1,2,2,1,2,1
4,24,2,2,929,5,4,4,3,1,2,3,31,2,2,1,3,1,2,1,1
4,4,4,0,1455,1,4,2,3,1,1,1,42,3,2,3,2,2,1,1,1
1,15,2,2,1845,1,2,4,2,3,1,2,46,3,1,1,3,1,1,1,1
2,48,0,0,8358,3,2,1,2,1,1,3,30,3,2,2,3,1,1,1,1
1,24,1,2,3349,3,2,4,3,1,4,4,30,3,3,1,3,2,2,1,2
4,12,2,0,2859,5,1,4,3,1,4,4,38,3,2,1,4,1,2,1,1
4,18,2,2,1533,1,2,4,4,2,1,2,43,3,2,1,2,2,1,1,2
4,24,2,3,3621,2,5,2,3,1,4,3,31,3,2,2,3,1,1,1,2
2,18,4,9,3590,1,1,3,4,1,3,3,40,3,2,3,1,2,2,1,1
1,36,3,9,2145,1,4,2,3,1,1,3,24,3,2,2,3,1,2,1,2
2,24,2,1,4113,3,2,3,2,1,4,3,28,3,1,1,3,1,1,1,2
4,36,2,2,10974,1,1,4,2,1,2,3,26,3,2,2,4,1,2,1,2
1,12,2,0,1893,1,3,4,2,3,4,2,29,3,2,1,3,1,2,1,1
1,24,4,3,1231,4,5,4,2,1,4,2,57,3,1,2,4,1,2,1,1
3,30,4,3,3656,5,5,4,3,1,4,2,49,2,2,2,2,1,1,1,1
2,9,4,3,1154,1,5,2,3,1,4,1,37,3,2,3,2,1,1,1,1
1,28,2,0,4006,1,3,3,3,1,2,3,45,3,2,1,2,1,1,1,2
2,24,2,2,3069,2,5,4,3,1,4,4,30,3,3,1,3,1,1,1,1
4,6,4,3,1740,1,5,2,4,1,2,1,30,3,1,2,3,1,1,1,1
2,21,3,0,2353,1,3,1,1,1,4,2,47,3,2,2,3,1,1,1,1
4,15,2,0,3556,5,3,3,3,1,2,4,29,3,2,1,3,1,1,1,1
4,24,2,3,2397,3,5,3,3,1,2,3,35,1,2,2,3,1,2,1,2
2,6,2,5,454,1,2,3,4,1,1,2,22,3,2,1,2,1,1,1,1
2,30,2,3,1715,5,3,4,2,1,1,3,26,3,2,1,3,1,1,1,1
2,27,4,3,2520,3,3,4,3,1,2,2,23,3,2,2,2,1,1,1,2
4,15,2,3,3568,1,5,4,2,1,2,3,54,1,1,1,4,1,2,1,1
4,42,2,3,7166,5,4,2,4,1,4,2,29,3,1,1,3,1,2,1,1
1,11,4,0,3939,1,3,1,3,1,2,1,40,3,2,2,2,2,1,1,1
2,15,2,5,1514,2,3,4,3,3,2,1,22,3,2,1,3,1,1,1,1
4,24,2,0,7393,1,3,1,3,1,4,2,43,3,2,1,2,2,1,1,1
1,24,1,0,1193,1,1,1,2,2,4,4,29,3,1,2,1,1,1,1,2
1,60,2,9,7297,1,5,4,3,2,4,4,36,3,1,1,3,1,1,1,2
4,30,4,3,2831,1,3,4,2,1,2,3,33,3,2,1,3,1,2,1,1
3,24,2,3,1258,3,3,3,2,1,3,3,57,3,2,1,2,1,1,1,1
2,6,2,3,753,1,3,2,2,3,3,1,64,3,2,1,3,1,1,1,1
2,18,3,9,2427,5,5,4,3,1,2,2,42,3,2,2,3,1,1,1,1
4,24,3,0,2538,1,5,4,3,1,4,3,47,3,2,2,2,2,1,1,2
2,15,1,0,1264,2,3,2,4,1,2,2,25,3,1,1,3,1,1,1,2
2,30,4,2,8386,1,4,2,3,1,2,2,49,3,2,1,3,1,1,1,2
4,48,2,9,4844,1,1,3,3,1,2,3,33,1,1,1,4,1,2,1,2
3,21,2,0,2923,2,3,1,2,1,1,3,28,1,2,1,4,1,2,1,1
1,36,2,1,8229,1,3,2,3,1,2,2,26,3,2,1,3,2,1,1,2
4,24,4,2,2028,1,4,2,3,1,2,2,30,3,2,2,2,1,1,1,1
1,15,4,2,1433,1,3,4,2,1,3,2,25,3,1,2,3,1,1,1,1
3,42,0,9,6289,1,2,2,1,1,1,2,33,3,2,2,3,1,1,1,1
4,13,2,3,1409,2,1,2,2,1,4,1,64,3,2,1,3,1,1,1,1
1,24,2,1,6579,1,1,4,3,1,2,4,29,3,3,1,4,1,2,1,1
2,24,4,3,1743,1,5,4,3,1,2,2,48,3,2,2,2,1,1,1,1
4,12,4,6,3565,5,2,2,3,1,1,2,37,3,2,2,2,2,1,1,1
4,15,1,3,1569,2,5,4,3,1,4,3,34,1,2,1,2,2,1,1,1
1,18,2,3,1936,5,4,2,4,1,4,3,23,3,1,2,2,1,1,1,1
1,36,2,2,3959,1,1,4,3,1,3,2,30,3,2,1,4,1,2,1,1
4,12,2,0,2390,5,5,4,3,1,3,3,50,3,2,1,3,1,2,1,1
4,12,2,2,1736,1,4,3,2,1,4,1,31,3,2,1,2,1,1,1,1
1,30,2,1,3857,1,3,4,1,1,4,2,40,3,2,1,4,1,2,1,1
```

```
4,12,2,3,804,1,5,4,3,1,4,3,38,3,2,1,3,1,1,1,1
1,45,2,3,1845,1,3,4,3,1,4,4,23,3,3,1,3,1,2,1,2
2,45,4,1,4576,2,1,3,3,1,4,3,27,3,2,1,3,1,1,1,1
```

Lampiran 3

Hasil Percobaan Untuk Menentukan Nilai Bobot Inersia (*w*)

Bobot inersia (*w*)	Akurasi
0,01	82,5 %
0,02	82,5 %
0,03	82,5 %
0,04	82,5 %
0,05	82,5 %
0,06	82,5 %
0,07	82,5 %
0,08	82,5 %
0,09	82,5 %
0,1	82,5 %
0,11	82,5 %
0,12	82,5 %
0,13	82,5 %
0,14	82,5 %
0,15	82,5 %
0,16	82,5 %
0,17	82,5 %
0,18	82,5 %
0,19	82,5 %
0,2	82,5 %
0,21	82,5 %
0,22	82,5 %
0,23	82,5 %
0,24	82,5 %
0,25	82,5 %
0,26	82,5 %
0,27	82,5 %
0,28	82,5 %
0,29	82,5 %
0,3	82,5 %
0,31	82,5 %
0,32	82,5 %
0,33	82,5 %
0,34	82,5 %

0,35	82,5 %
0,36	82,5 %
0,37	82,5 %
0,38	82,5 %
0,39	82,5 %
0,4	82,5 %
0,41	82,5 %
0,42	82,5 %
0,43	82,5 %
0,44	82,5 %
0,45	82,5 %
0,46	82,5 %
0,47	82,5 %
0,48	82,5 %
0,49	82,5 %
0,5	82,5 %
0,51	82,5 %
0,52	82,5 %
0,53	82,5 %
0,54	82,5 %
0,55	82,5 %
0,56	82,5 %
0,57	82,5 %
0,58	82,5 %
0,59	82,5 %
0,6	82,5 %
0,61	82,5 %
0,62	82,5 %
0,63	82,5 %
0,64	82,5 %
0,65	82,5 %
0,66	82,5 %
0,67	82,5 %
0,68	82,5 %
0,69	82,5 %
0,7	82,5 %
0,71	82,5 %
0,72	82,5 %
0,73	82,5 %
0,74	82,5 %
0,75	82,5 %

0,76	82,5 %
0,77	82,5 %
0,78	82,5 %
0,79	82,5 %
0,8	82,5 %
0,81	82,5 %
0,82	82,5 %
0,83	82,5 %
0,84	82,5 %
0,85	82,5 %
0,86	82,5 %
0,87	82,5 %
0,88	82,5 %
0,89	82,5 %
0,9	82,5 %
0,91	82,5 %
0,92	82,5 %
0,93	82,5 %
0,94	82,5 %
0,95	82,5 %
0,96	82,5 %
0,97	82,5 %
0,98	82,5 %
0,99	82,5 %
1	82,5 %

Lampiran 3

Surat Keputusan Penetapan Dosen Pembimbing Skripsi

UNNES

KEPUTUSAN
DEKAN FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
Nomor: 12026/UN37.1.9/TD.06/2019
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	: III/b
Jabatan Akademik	: Asisten Ahli
Sebagai Pembimbing	

Untuk membimbing mahasiswa penyusun skripsi/Tugas Akhir :

Nama	: MUHAMMAD ALI IMRON
NIM	: 4611415034
Jurusan/Prodi	: Ilmu Komputer/Teknik Informatika
Topik	: DATA MINING

KEDUA : Keputusan ini mulai berlaku sejak tanggal ditetapkan.

DITETAPKAN DI : SEMARANG
PADA TANGGAL : 21 Oktober 2019
DEKAN,

Dr. Sugianto, M.Si.
NIP 196102191993031001

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

4611415034
FM-03-AKD-24/Rev. 00