PEMODELAN PEMILAHAN CITRA GUNUNG MERAPI SECARA OTOMATIS PADA BPPTKG YOGYAKARTA

Disusun Oleh:

N a m a : Rifqi Afif
NIM : 15523072

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PEMODELAN PEMILAHAN CITRA GUNUNG MERAPI SECARA OTOMATIS PADA BPPTKG YOGYAKARTA

TUGAS AKHIR

Disusun Oleh:

N a m a : Rifqi Afif
NIM : 15523072

Yogyakarta, 14 Februari 2020
Pembimbing,

(Arrie Kurniawardhani, S.Si., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PEMODELAN PEMILAHAN CITRA GUNUNG MERAPI SECARA OTOMATIS PADA BPPTKG YOGYAKARTA

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 14 Februari 2020

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom. _____

Anggota 1

Dr. Sri Kusumadewi, S.Si., M.T. _____

Anggota 2

Ari Sujarwo, S.Kom., M.I.T. _____

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertandatangan di bawahini:

Nama : RifqiAfif
NIM : 15523072

Tugasakhirdenganjudul:

PEMODELAN PEMILAHAN CITRA GUNUNG MERAPI SECARA OTOMATIS PADA BPPTKG YOGYAKARTA

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 14Februari 2020

(RifqiAfif)

HALAMAN PERSEMBAHAN

Alhamdulillahirabbil'alamin. Segala puji bagi Allah SWT, Dzat yang telah memberikan rahmat dan hidayahNya sehingga memudahkan penulis untuk menyelesaikan Tugas Akhir dengan lancar tanpa halangan apapun. Tugas Akhir ini saya persembahkan kepada:

- Kedua Orang Tua saya yang selalu mendukung bersusah payah hingga saya berhasil mengemban Pendidikan di perguruan tinggi.
- Ibu Arrie Kurniawardhani, S.Si., M.Kom. yang selalu meluangkan waktunya dan memotivasi saya selama pengerjaan Tugas Akhir ini.
- Saudariku yang selalu membantu dan mendukungku selama ada kesulitan.
- Dan semua yang telah memberikan dukungan, baik moril, materi dan doanya. Terima kasih.
- Teman-teman BASECAMP yang selalu menemani selama masa perjuangan kuliah mulai dari awal hingga akhir.
- Teman-teman 5 Swan yang selalu ada ketika suka maupun duka.
- Teman-teman Softball Unisi yang sudah menerimaku menjadi salah satu keluarga kalian.
- Sobat Eventure yang merelakan waktu dan tenaganya untuk berjuang dalam suka dan duka.
- Dan semua pihak yang telah banyak membantu saya dalam pelaksanaan Tugas Akhir yang tidak dapat disebutkan satu persatu.

HALAMAN MOTO

“Jangan pernah berhenti mengejar impianmu sekalipun harus merangkak”
-R.A.

“Bukan kesulitan yang membuat takut, namun ketakutan itu yang membuat sulit”
-Unkown

“Hiduplah seperti pohon yang lebat buahnya”
-Abu Bakr Ash-Shiddiq

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Alhamdulillahirobbi'alamin segala puji dan syukur dihanturkan kepada Dzat penyempurna sesuatu Allah Ta'ala Rabb semesta alam atas segala tauhid, berkat, rahmat dan hidayah-Nya. Shalawat serta salam dicurahkan kepada baginda besar umat Islam, Nabi Agung Muhammad SAW yang telah membawa kebaikan dan keterangan dari jaman Jahiliyah menuju jaman Islamiyah seperti saat ini. Tak lupa juga mengucapkan rasa syukur dan doa yang tak terputus karena dapat menempuh Tugas Akhir (TA) Universitas Islam Indonesia (UII) dari awal pelaksanaan hingga berakhirnya dengan pembuatan laporan akhir kegiatan TA ini dengan lancar dan dapat terselesaikan dengan baik.

Adapun tujuan dari pelaksanaan Tugas Akhir ini adalah sebagai salah satu syarat yang harus dipenuhi untuk penyelesaian akhir untuk memperoleh gelar sarjana di Jurusan Informatika Universitas Islam Indonesia (UII). Adapun Tugas Akhir saya mengenai Pemodelan Pemilahan Citra Gunung Merapi Secara Otomatis Pada BPPTKG Yogyakarta.

Pelaksanaan Tugas Akhir ini merupakan salah satu mata kuliah wajib dari jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia (UII) dan juga merupakan sarana untuk menambah rasa kepercayaan diri dari mahasiswa untuk menjadi *fresh graduate* karena telah memiliki pengalaman kerja, serta menambah persiapan untuk menghadapi dunia kerja yang jauh berbeda dengan lingkungan akademik.

Penulis menyadari bahwa keberhasilan terlaksananya proyek Tugas Akhir (TA) dan penyusunan laporan akhir TA ini tidak terlepas dari bimbingan, dorongan, dan bantuan dari berbagai pihak baik bersifat material maupun spiritual. Oleh karena itu, dalam kesempatan ini ijinkan penulis mengucapkan terimakasih kepada:

1. ALLAH SWT, berkat limpahan karunia, rahmat dan hidayah-Nya sehingga penulis mampu melaksanakan proyek TA serta penyusunan laporan TA ini dengan lancar.
2. Keluarga, khususnya kedua orang tua, kakak perempuan, dan kakak laki-laki, beserta keponakan yang tidak pernah lupa selalu memberi inspirasi, semangat, dan motivasi selama penulis melaksanakan proyek TA dan penyusunan laporan TA.
3. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D. sebagai Rektor Universitas Islam Indonesia.
4. Bapak Prof. Dr. Ir. Hari Purnomo., M.T. sebagai Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.

5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. sebagai Ketua Prodi Studi Teknik Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Ibu Arrie Kurniawardhani, S.Si., M.Kom. sebagai dosen pembimbing Tugas Akhir yang selalu memberikan semangat dan membimbing saya dengan sabar.
7. Bapak Dr. Agus Budi Santoso, S.Si., M.Sc. sebagai ahli yang membantu selama proses analisis data citra Gunung Merapi.

Penulis menyadari sepenuhnya bahwa dalam pelaksanaan TA ini masih banyak kekeliruan dan kekurangan dalam tindak tanduk untuk itu penulis mohon maaf, semoga laporan akhir TA ini dapat bermanfaat bagi semua semua pihak serta mendapat Ridho Allah SWT.

Namun, perlu juga disadari bahwa dalam pelaporan ini tentunya terdapat kekurangan-kekurangan yang secara manusiawi tidak mampu dibenarkan, hal ini tentu menjadi inspirasi bagi generasi berikutnya untuk terus menerus melakukan perbaikan.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 25 Februari 2020

(Rifqi Afif)

SARI

Gunung Merapi merupakan salah satu gunung api teraktif yang ada di Indonesia. Guna menanggulangi dampak bencana dari Gunung Merapi, maka setiap harinya para pakar harus mengamati segala fenomena yang terjadi pada Gunung Merapi. Salah satu metode pengamatan yang dilakukan oleh pakar yaitu secara visual/penglihatan. Metode pengamatan secara visual dilakukan dengan cara mengamati citra dari Gunung Merapi yang ditangkap oleh masing-masing pos pengamatan Gunung Merapi. Setiap harinya terdapat ratusan data berupa citra Gunung Merapi yang diterima dari tiap pos. Dari data tersebut ada beberapa yang tidak layak untuk diamati oleh pakar karena citra yang ditangkap terhalang oleh kabut atau awan. Faktanya terdapat banyak data yang dikirim oleh tiap pos, tidak layak bagi pakar untuk dilakukan analisis. Data yang tidak layak ini perlu dihapus dari media penyimpanan agar tidak membebani Server, namun untuk saat ini penghapusan data yang tidak layak tersebut dilakukan secara manual sehingga akan meberatkan pekerjaan pengamat/pakar. Penilitan ini bertujuan untuk menyusun pemodelan yang dapat memilah citra Gunung Merapi secara otomatis dengan *Machine Learning*. Model yang telah dibangun akan memilah data yang layak dan tidak layak. Dalam pengembangan model ini terdapat beberapa skenario yang dilakukan guna menghasilkan performa model yang terbaik. Model yang digunakan yaitu adalah Support Vector Machine dan K-Nearest Neighbour. Dalam penelitian ini, digunakan dua metode seleksi fitur yaitu Correlation-based Feature Selection dan Chi-Square. Seleksi fitur diharapkan mampu mengeleminasi fitur-fitur yang tidak mendukung selama proses pembangunan model *Machine Learning*. Hasil dari penelitian ini menyimpulkan bahwa performa yang diberikan dari kombinasi antara klasifikasi KNN dan seleksi fitur Chi-Square dengan nilai k=27 (jumlah sampel dari fitur) menghasilkan nilai akurasi sebesar 90% dan waktu komputasi selama 0.001187 detik.

Kata kunci: klasifikasi, *SVM*, *KNN*, seleksi fitur, *CFS*, *Chi-Square*, Merapi.

GLOSARIUM

Library	Sekumpulan program yang digunakan untuk membantu proses pembuatan perangkat lunak
Micro Web Framework	Sebuah framework website yang minimalis
Plot	Representasi data kedalam bentuk grafik agar penyajian datanya lebih menarik
Hyperplane	Sebuah garis lurus atau bidang mendatar yang memisahkan kelas-kelas
Preprocessing	Suatu proses atau langkah yang dilakukan untuk membuat data mentah menjadi data yang berkualitas
Channel	Warna primer (Contoh: *Red, Green, Blue*)
Hypertuning	Sebuah upaya dalam menentukan parameter yang dapat mengoptimalkan proses pembelajaran algoritma
Input	Proses Masukkan
Output	Proses Keluaran
Fitting	Suatu proses pembentukan pola model terhadap data
Fold	Lipatan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Gunung Merapi menjadi salah satu gunung berapi yang paling aktif di Indonesia di antara 130 gunung berapi aktif lainnya (PRATOMO, 2006). Gunung ini terletak di antara perbatasan wilayah Daerah Istimewa Yogyakarta dan Jawa Tengah. Gunung Merapi menjadi salah satu gunung berapi yang paling aktif berdasarkan seringnya terjadi erupsi dalam rentang waktu 2-7 tahun sekali (IKHSAN, FUJITA, & TAKEBAYASHI, 2009). Dampak yang ditimbulkan dari erupsi ini dapat merugikan bagi manusia salah satunya tercemarnya udara karena kandungan zat berbahaya sehingga menyebabkan kematian. Untuk itu diperlukan pemantauan aktivitas Gunung Merapi, salah satunya secara visual (Penglihatan). Pemantauan aktivitas Gunung Merapi dilakukan oleh Balai Penyelidikan dan Pengembangan Teknologi Kebencanaan (BPPTKG).

BPPTKG adalah sebuah Unit Pelaksana Teknis (UPT) yang setara dengan tingkat eselon III dan merupakan organisasi yang berada di bawah Pusat Vulkanologi dan Mitigasi Bencana Geologi (PVMBG). BPPTKG memiliki lima pos pengamatan yang terletak di sekitar Gunung Merapi yaitu, pos Kaliurang (lereng selatan), Babadan (lereng barat laut), Selo (lereng utara), Jrakah (lereng utara), Ngepos (lereng barat daya). Kelima pos ini melakukan pencatatan aktivitas Gunung Merapi seperti data visual, Seismisitas, Deformasi, Guguran, dsb. Data visual memuat informasi tentang aktivitas Gunung Merapi dalam bentuk citra yang ditangkap dengan menggunakan kamera DSLR. Data visual termasuk salah satu tolak ukur yang paling sering digunakan oleh pakar dalam menentukan tingkat aktivitas Gunung Merapi. Sebuah data visual yang baik/layak untuk diamati oleh pakar, merupakan sebuah citra yang menampakkan objek Gunung Merapi secara jelas. Data visual ini ditransmisikan dari kelima pos pengamatan Merapi ke ruang Monitoring BPPTKG setiap 10 menit/gambar. Dari semua citra Gunung Merapi yang diperoleh ternyata tidak semuanya dapat menangkap objek Gunung Merapi secara jelas. Terdapat banyak citra dimana objek Gunung Merapi tidak tampak jelas karena terhalang oleh awan atau kabut. Dengan kondisi tersebut, para pakar mengalami kesulitan dalam mengamati kondisi Gunung Merapi. Data visual yang tidak layak untuk diamati perlu dihapus secara manual oleh pakar agar tidak memenuhi media penyimpanan data. Jika data visual tersebut diterima setiap 10 menit/gambar dan dalam satu harinya ada 1440 menit, maka terdapat sebanyak 720 (1440 menit/10 menit x 5 pos) data citra yang diperoleh. Berdasarkan

perhitungan tersebut tentu akan membutuhkan waktu dan usaha yang tinggi untuk mengapus data yang tidak layak tersebut secara manual.

Oleh karena itu, di dalam penelitian ini penulis mengusulkan suatu pemodelan yang dapat mengidentifikasi apakah citra tersebut menampakkan objek Gunung Merapi secara jelas atau tidak. Model ini nantinya akan digunakan pada pengembangan sistem penghapusan otomatis. Penelitian ini merupakan riset awal dalam menyelesaikan permasalahan, yang mana lebih fokus pada pengembangan model saja. Dengan adanya model ini kedepannya media penyimpanan data pada Server BPPTKG dapat terkelola dengan baik.

1.2 Rumusan Masalah

Adapun perumusan masalah dari permasalahan yang sudah dipaparkan sebelumnya adalah sebagai berikut:

a. Metode apa yang tepat untuk digunakan dalam memilah citra dari Gunung Merapi secara otomatis?
b. Fitur apa saja yang paling memengaruhi kinerja model pemilahan citra Gunung Merapi?

1.3 Batasan Masalah

Dari permasalahan di atas terdapat beberapa batasan sebagai berikut:

c. Data yang menjadi masukan hanya berupa data citra dari Gunung Merapi.
d. Citra Gunung Merapi dikelompokkan menjadi dua kelompok yaitu siang jelas, dan siang kabut.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini dilakukan adalah untuk menentukan metode yang dapat memilah citra Gunung Merapi secara otomatis.

1.5 Usulan Penyelesaian

Dengan adanya permasalahan yang telah disebutkan di atas, penulis mengajukan sebuah langkah penyelesaian dengan cara membangun sebuah model dengan memanfaatkan metode-metode di bidang Visi Komputer yang mampu mengelompokkan citra berdasarkan karakteristiknya secara otomatis.

1.6 Metode Penelitian

a. Pengumpulan Data

1. Observasi

   Pada tahap ini dilakukan observasi terhadap teknologi yang digunakan oleh BPPTKG untuk saat ini.

2. Wawancara

   Tahap ini dilakukan dengan cara melakukan wawancara terhadap pihak terkait pengembangan teknologi BPPTKG untuk memenuhi kebutuhan sistem.

3. Studi Pustaka

   Tahap ini dilakukan pengumpulan data dengan cara mempelajari penelitian terkait yang sudah ada sebelumnya.

b. Pengembangan Model

   Dalam Pengembangan model ini dilakukan dengan cara menerapkan metode Visi Komputer dan dalam pengaplikasiannya menggunakan *Library* Python seperti Matplotlib, Scipy, Numpy, SKlearn, dan OpenCV.

c. Pengujian Model

   Pengujian model dilakukan untuk melihat seberapa baik kinerja model yang diamati melalui nilai akurasi dalam melakukan pengelompokkan citra Gunung Merapi dengan membandingkan hasil kesimpulan Bapak Agus Budi Santoso selaku pakar dan Kepala Seksi Gunung Merapi BPPTKG Yogyakarta.

1.7 Sistematika Penulisan

Tujuan utama dari pembuatan sistematika penulisan adalah agar membantu dalam memahami laporan tugas akhir ini. Dalam sistematika penulisan ini terdapat beberapa pembagian bab pada laporan tugas akhir ini diantaranya sebagai berikut:

BAB I PENDAHULUAN

Dalam bab ini mendeskripsikan tentang hal-hal dasar dari terwujudnya penelitan ini yang mana didalamnya memuat latar belakang, rumusan masalah, Batasan masalah, tujuan penelitian, manfaat penelitian, dan metode penelitian.

BAB II LANDASAN TEORI

Pada bab ini berisikan tentang landasan teori yang digunakan atau dirujuk selama berjalannya penelitian ini, contohnya seperti algoritma, metode analisa, dll.

BAB III METODOLOGI PENELITIAN

Pada bagian bab ini berisikan tentang bagaimana metode pengembangan model diterapkan, mulai dari tahap perancangan hingga pengujian.

BAB IV HASIL DAN PEMBAHASAN

Dalam bab ini mendeskripsikan tentang hasil dari pengembangan model yang telah dilakukan. Pengembangan dan pengujian model dilakukan sesuai dengan rancangan yang dibuat sebelumnya.

BAB V KESIMPULAN DAN SARAN

Pada bab ini berisikan tentang kesimpulan dari penelitian yang telah dilakukan dan juga saran untuk memberikan wawasan terhadap penelitian yang lebih lanjut.

BAB II
LANDASAN TEORI

2.1 *Machine Learning*

Machine Learning merupakan sebuah cabang dari algoritma komputasi yang di desain untuk meniru kecerdasan manusia berdasarkan di lingkungan sekitar. *Machine Learning* dapat mengerjakan pekerjaan yang berat, terlebih di era industri 4.0 yang memiliki jumlah data yang begitu besar (*Big Data*). Berbagai teknik di dalam *Machine Learning* telah diterapkan di beragam bidang mulai dari visi komputer, pengenalan pola, keuangan, hiburan, dan biomedis (Murphy, 2015). Arthur Samuel merupakan sosok yang pertama kali mengenalkan Machine Learning pada tahun 1952. Pada dasarnya *Machine Learning* merupakan sebuah kemampuan komputer dalam memproses data dan mempelajarinya. *Machine Learning* juga didefinisikan sebagai suatu metode komputasi dimana pengalaman digunakan untuk menghasilkan keakuratan prediksi dan meningkatkan kinerja, pengalaman yang dimaksud pada *Machine Learning* adalah berupa informasi di masa lalu yang disediakan untuk pembelajaran, umumnya berupa data elektronik yang dikumpulkan dan tersedia untuk analisis data (Mohri, Rostamizadeh, & Talwalkar, 2018).

Dalam Kecerdasan Buatan (*Artificial Intelligence*), *Machine Learning* menjadi salah satu disiplin ilmu dimana mengkaji tentang pembangunan sebuah sistem berdasarkan data. Terdapat 4 hal yang dipelajari dalam *Machine Learning*, di antaranya:

1. *Unsupervised* Learning

   Metode *Unsupervised* Learning merupakan sebuah metode pembelajaran tanpa arahan, sehingga algoritma dapat menindaki informasi yang diberikan tanpa adanya arahan. Contohnya penerapan algoritma Clustering (pengelompokkan).

2. *Supervised Learning*

   Berbeda dari *Unsupervised Learning, Supervised Learning* adalah sebuah metode pembelajaran yang terarah, metode ini memiliki *target output* yang dijadikan sebagai arahan dalam proses pembelajaran.

2.2 *K-Nearest Neighbour*

K-Nearest Neighbour (KNN), merupakan salah satu bagian dari kelompok *instance-based learning.* Selin itu algoritma KNN merupakan salah satu algoritma *Machine* Learning yang menggunakan teknik *lazy learning.* Proses pembelajaran pada KNN dilakukan dengan cara mencari kelompok k (tetangga) pada data *training*, Banyak cara yang dilakukan dalam mengukur jarak terdekat antara data *testing* dengan data *training*, diantaranya adalah menggunakan metode *Euclidean distance* dan *Manhattan distance*, namun metode yang paling sering digunakan adalah *Euclidean distance* (Leidiyana, 2013), yang mana persamaannya dituliskan seperti persamaan (.

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} \qquad (2.1)$$

Dimana a = $a_1, a_2, \ldots, a_n$, dan b = $b, b_2, \ldots, b_n$, dua *record* tersebut diwakili n nilai atribut.

2.3 SVM (*Support Vector Machine*)

Support Vector Machine (SVM) adalah metode *supervised learning* yang dipopulerkan oleh Vapnik dan Cortes (1995). SVM memiliki pengaplikasian yang luas di bidang statistika seperti analisis klasifikasi dan regresi. Banyak keuntungan khusus yang dimiliki SVM dalam menyelesaikan kasus sampel kecil, non-linear, dan pengenalan pola berdimensi tinggi, selain itu SVM secara luas dapat diterapkan pada kasus *Machine Learning* seperti *function fitting*. Prinsip dasar dari metode ini adalah untuk menemukan Hyperplane fractal untuk set pelatihan V di ruang sampel, yang akan memaksimalkan pemisahan kategori. Selain itu, sebagai masalah pemrograman kuadratik, SVM dapat menemukan solusi optimal unik secara global, sehingga dapat menghindari terjadinya minimum lokal. Prinsip dan proses solusinya adalah sebagai berikut:

Diberikan sebuah sampel dataset $V = \{(x_i, y_i) | x_i \in R^m, y_i \in \{-1, +1\}, i = 1, \ldots n\}$, maka fungsi diskriminan *Support Vector Machine* yang dituliskan ke dalam persamaan (2.2).

$$f(x) = sign(\sum_{i=1}^{n} A_i y_i K(x, x_i) + B) \qquad (2.2)$$

Dimana $K(x, x_i)$ adalah fungsi kernel, i adalah angka dari *support vector*. Fungsi kernel ini sangat penting dalam proses pelatihan dari support vector, yang secara efektif dapat mengatasi

masalah *Dimensionality Disaster*. Fungsi kernel yang tepat dapat meningkatkan akurasi prediksi model klasifikasi. Fungsi kernel yang umum digunakan adalah fungsi kernel linear, *Radial Basis Function*, fungsi kernel polynomial, fungsi kernel sigmoid, dan sebagainya.(Huo et al., 2020).

2.3.1 Radial Basis Function

Pendekatan *Radial Basis Function* (RBF) merupakan metode *meshfree* yang sangat kuat dan akurat secara spektral. RBF digunakan untuk menginterpolasi data tersebar multi dimensi, menyelesaikan persamaan integral, diferensial parsial, dan juga diferensial stokastik. RBF dinotasikan sebagai $\varphi(r)$, tergantung pada jarak *r* dari titik pusat, yaitu jarak antara titik sumber dan bidang (Karimi, Kazem, Ahmadian, Adibi, & Ballestra, 2020). Beberapa RBF yang paling umum digunakan dituliskan kedalam bentuk persamaan.

- Gaussian (GA):

$$f\varphi(r) = \exp(-(\epsilon r)^2), \qquad (2.3)$$

- Inverse Quadratic (IQ):

$$\varphi(r) = \frac{1}{(\epsilon r)^2+1}, \qquad (2.4)$$

- Multiquadrics (MQ):

$$\varphi(r) = \sqrt{(\epsilon r)^2 + 1}, \qquad (2.5)$$

- Inverse Multiquadrics (IMQ):

$$\varphi(r) = \frac{1}{\sqrt{(\epsilon r)^2 + 1}}, \qquad (2.6)$$

- Generalized inverse multiquadrics (GIMQ):

$$\varphi(r) = \frac{1}{((\epsilon r)^2 + 1)^{\frac{\beta}{2}}}, \beta = 1, 2, 3, \dots, \quad (2.7)$$

- Qubic:

$$\varphi(r) = r^3 \quad (2.8)$$

Dimana ϵ merupakan *shape parameter*.

2.4 *Cross-Validation*

Cross-Validation merupakan parameter penting sebagai bentuk evaluasi dari algoritma *Machine Learning* seberapa baik kemampuannya dalam memprediksi. Untuk menghindari *Overfitting* yang mana umumnya diamati dalam klasifikasi serta regresi menggunakan *machine learning*, untuk itu *Cross-Validation* dibutuhkan. Dalam melakukan *Cross-Validation* terhadap suatu dataset, jumlah lipatan perlu disebutkan (*K-fold*). Pada *K-fold Cross-Validation*, dataset dibagi secara acak menjadi K dengan ukuran sampel yang sama (Huo et al., 2020).

2.5 *Confusion Matrix*

Sebuah *Confusion Matrix* merepresentasikan informasi tentang seberapa sering perilaku tertentu dideteksi benar dan seberapa sering perilaku tersebut diklasifikasikan sebagai perilaku yang lain. Akurasi dari klasifikasi biasanya diringkas dari beberapa indikator kinerja seperti *precision*, *sensitivity*, *specificity* (Ruuska et al., 2018). Dalam *confusion matrix* biner hasil prediksi yang diklasifikasikan secara benar kelas positif disebut *True Positive* (TP) dan hasil pengklasifikasian dengan benar ke dalam kelas negatif disebut *True Negative* (TN). Contoh ketika kelas positif diklasifikasikan sebagai kelas negatif disebut sebagai *False Negative* (FN) dan ketika kelas negative diklasifikasikan sebagai kelas positif disebut dengan *False Positive* (FP). Nilai tersebut dapat digunakan dalam menghitung indikator kinerja dari *classfier* dengan mencerminkan bagaimana *classifier* dalam mendeteksi kelas yang diberikan. Indikator yang paling umum digunakan adalah *precision*, *specificity*, *sensitivity*, dan akurasi (Ruuska et al., 2018).

2.5.1 Akurasi

Akurasi merupakan indikator dari seberapa berhasil sebuah *classifier* dalam melakukan klasifikasi secara benar dari kelas negatif maupun positif. Proses perhitungan nilai akurasi dituliskan ke dalam persamaan (2.9).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.9)$$

2.5.2 *Sensitivity*

Sensitivity merupakan indikator yang menandakan seberapa berhasil sebuah *classifier* dalam melakukan klasifikasi secara benar terhadap kelas positif. Proses perhitungan nilai *sensitivity* dituliskan ke dalam persamaan (2.10).

$$Sensitivity = \frac{TP}{TP + FN} \quad (2.10)$$

2.5.3 *Specificity*

Specificity merupakan indikator yang menandakan seberapa berhasil sebuah *classifier* dalam melakukan klasifikasi secara benar terhadap kelas negatif. Proses perhitungan nilai *specificity* dituliskan ke dalam persamaan (2.11).

$$Specificity = \frac{TN}{TN + FP} \quad (2.11)$$

2.6 *Color Space* (ruang warna)

Color merupakan sebuah cara dari HVS (*Human Visual System*) dalam mengukur bagian dari spektrum elektromagnetik, dengan perkiraan antara 300 dan 830 nm. Karena sifat tertentu dari HVS kita tidak dapat melihat semua kemungkinan kombinasi spektrum yang terlihat tetapi kita cenderung mengelompokkan berbagai spektrum menjadi warna/*color*. *Color space* atau ruang warna adalah notasi yang dengannya kita dapat menentukan warna, yaitu persepsi manusia tentang spektrum elektromagnetik yang terlihat (Tkalčič & Tasič, 2003).

2.6.1 RGB Space

RGB (*Red, Green, Blue*) merupakan ruang warna yang paling banyak digunakan dan merupakan ruang warna yang secara *default* untuk merepresentasikan citra digital. Dari ruang warna RGB bisa diperoleh ruang warna lain dengan melakukan transformasi linear atau non-linear. Normalisasi RGB merupakan salah satu representasi yang sangat mudah untuk didapatkan melalui prosedur normalisasi sederharna dengan nilai RGB. Salah satu sifat yang patut diperhatikan pada representasi ini adalah pada permukaan *matte,* RGB secara sementara

mengabaikan cahaya sekitar, sehingga RGB yang dinormalisasi tidak berubah-ubah terhadap perubahan orientasi permukaan yang relatif dengan sumber cahaya. Nilai normalisasi dari RGB bisa didapatkan dengan menggunakan persamaan (2.12).

$$r = \frac{R}{R+G+B}$$
$$g = \frac{G}{R+G+B} \qquad (2.12)$$
$$b = \frac{B}{R+G+B}$$

Dimana:

r = normalisasi dari nilai komponen *Red*

g = normalisasi dari nilai komponen *Green*

b = normalisasi dari nilai komponen *Blue*

R = nilai komponen *Red*

G = nilai komponen *Green*

B = nilai komponen *Blue*

2.6.2 HSV Space

Ruang warna HSV(*Hue, Saturation, Value*) merupakan transformasi linear dari ruang warna RGB dan karenanya HSV mewarisi semua kekurangan seperti ketergantungan perangkat dan nonlinearitas (Tkalčič & Tasič, 2003). HSV adalah salah satu ruang warna yang baik dalam mengenali warna-warna dasar. Selain itu HSV juga memiliki kepekaan terhadap perubahan intensitas cahaya. Hal ini menjadikan HSV salah satu ruang warna yang unggul diantara ruang warna lainnya. HSV merupakan representasi ruang warna yang lain dari RGB. Yang mana ruang warna HSV memiliki basi *Cylindrical Coordinates*. HSV dikatakan lebih baik dari ruang warna RGB dikarenakan HSV mampu merepresentasikan warna dari perspektif penglihatan manusia. Intensitas warna yang dimiliki oleh HSV berkisar antara 0 sampai dengan 1 (Hardiyanto & Sartika, 2018), yang mana nilai tersebut diperoleh dari transformasi RGB dengan menggunakan persamaan transformasi nonlinier seperti yang terlihat pada persamaan

$$H = \begin{cases} \theta \; jika \; b \leq g \\ 360° \; jika \; b > g \end{cases} \qquad (2.13)$$

Dengan,

$$\theta = cos^{-1} \frac{\frac{1}{2}(r-g)+(r-b)}{[(r-g)^2+(r-b)(g-b)]^{1/2}} \quad (2.14)$$

$$v = \max(r, g, b) \quad (2.15)$$

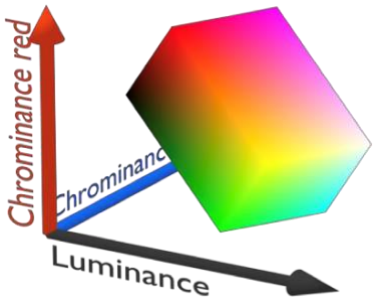
$$s = \frac{v - \min(r, g, b)}{v} \quad (2.16)$$

2.6.3 CIE Lab Space

Pada 1976 *CIE (Commision Internationale de l'Eclairage)* mengusulkan dua ruang warna (CIELuv & CIELab) yang mana tujuan utamanya adalah untuk menghasilkan persepsi ruang yang sama. Hal tersebut mengartikan bahwa jarak *Euclidian* antara dua warna dalam ruang warna CIELuv / CIELab sangat berkolerasi dengan persepsi visual/penglihatan manusia (Tkalčič & Tasič, 2003). Kepanjangan dari LAB adalah *Luminance* (kecerahan), A dan B (merupakan komponen berwarna). A memiliki *channel* warna yang berkisar dai warna hijau ke merah dan rentang B memliki *channel warna* dari biru ke kuning. *Luminance* sendiri memiliki nilai intensitas yang berkisar mulai dari 0 hingga 100, untuk komponen warna A nilai intensitas warnanya berkisar -120 hingga +120 (hijau ke merah) dan komponen warna B memiliki rentang nilai intensitas warnanya dari -120 hingga +120 (dari biru ke kuning) (Hapsari & Hidayattullah, 2013).

2.6.4 YCbCr

Ycbcr merupakan sinyal non-linear dari RGB yang di*encode*, umumnya digunakan pada studio televisi di Eropa dan untuk kompresi citra. Seperti yang diilustrasikan pada Gambar 2.1, dimana luma mewakili warna (pencahayaan yang dihitung dari RGB non-linear) yang dibangun sebagai jumlah bobot nilai-nilai RGB.



Gambar 2.1 Ilustrasi ruang warna Ycbcr

Sumber: (Kolkur, Kalbande, Shimpi, Bapat, & Jatakia, 2017)

YCbCr merupakan salah satu ruang warna yang umum digunakan dalam dunia digital. Karena representasi YCbCr mampu menghilangkan beberapa informasi warna yang berlebih. YCbCr digunakan dalam standar kompresi video dan citra seperti JPEG, MPEG, MPEG2 dan MPEG4. Nilai Y sebagai komponen tunggal yang menyimpan informasi lumina, dan informasi kroma disimpan ke dalam dua komponen warna yang berbeda yaitu Cb dan Cr. Cb merupakan representasi perbedaan antara komponen biru dan nilai referensi. Sedangkan Cr mewakili perbedaan antara komponen warna merah dan nilai referensi (Kolkur et al., 2017). Nilai YCbCr dapat diperoleh menggunakan ruang warna RGB yang dituliskan ke dalam persamaan.

$$Y = 0.299R + 0.287G + 0.11B \quad (2.17)$$

$$Cr = R - Y \quad (2.18)$$

$$Cb = B - y \quad (2.19)$$

2.7 CFS (Correlation-Based Feature Selection)

CFS (Correlation-based Feature Selection) merupakan salah satu seleksi fitur yang menggunakan heuristic berbasis korelasi untuk mengevaluasi seberapa baik nilai fitur. Inti dari algoritma CFS adalah *heuristic* untuk mengevaluasi nilai atau merit dari subset fitur. *Heuristic* ini memperhitungkan kegunaan individu fitur untuk memprediksi label kelas Bersama dengan tingkat interelasi antara mereka yang dituliskan ke dalam persamaan (2.20).

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{f\,f}}}} \qquad (2.20)$$

Dalam persamaan (2.20) menyatakan bahwa $Merit_s$ merupakan heuristic "merit" dari subset fitur *S* yang memuat fitur *k*, $\overline{r_{cf}}$ adalah rata-rata korelasi kelas fitur $(f \in S)$, dan $\overline{r_{f\,f}}$ adalah rata-rata interkorelasi masing-masing fitur (Hall & Smith, 1999).

2.8 Chi-Square

Teori statistika digunakan oleh fitur Chi-Square untuk menguji independensi dari sebuah *term* dengan kategorinya. Tujuan dari penggunaan seleksi fitur adalah untuk menghilangkan fitur yang mengganggu dalam proses klasifikasi. Berdasarkan teori statistika pada seleksi fitur Chi-Square, terdapat dua peristiwa di antaranya adalah, terjadinya kemunculan dari fitur dan kategori, yang mana kemudian setiap nilai *term* diurutkan mulai dari yang tertinggi berdasarkan persamaan (2.21).

$$X^2(D,t,c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \qquad (2.21)$$

Seleksi fitur Chi-Square dilakukan dengan cara setiap fitur diurutkan berdasarkan hasil seleksi fitur Chi-Square mulai dari nilai yang terbesar hingga nilai yang terkecil. Ketika nilai seleksi fitur Chi-Square lebih besar dari nilai signifikan maka hal tersebut menunjukkan penolakan hipotesis independensi. Jika dua peristiwa menunjukkan dependen, maka fitur tersebut sama dengan label kategori yang sesuai pada kategori (LING, N. KENCANA, & OKA, 2014).

BAB III
METODOLOGI PENELITIAN

3.1 Pengumpulan Data

Pada tahapan awal penulis melakukan pengumpulan data digunakan sebagai *Dataset*. *Dataset* ini diperoleh dari Badan Penyelidikan dan Penilitian Teknologi Kegempaan Geologi (BPPTKG) Yogyakarta. Sebanyak 100 citra Gunung Merapi didapatkan dari empat kamera DSLR pos pemantau dengan posisi sudut pandang yang berbeda-beda. Citra ini beresolusi sebesar 4274 × 2848, dengan ruang warna RGB dan format file ".jpeg". Rata-rata ukuran file dari citra Gunung Merapi ini sebesar 8MB. Dari total keseluruhan citra, citra dibagi berdasarkan dua kategori yaitu 50 citra untuk kategori siang jelas (gambar tidak dihapus) dan 50 citra untuk kategori siang kabut (gambar dihapus). Contoh citra dari Gunung Merapi berkategori siang jelas dapat dilihat pada Gambar 3.1 dan citra Gunung Merapi kategori siang kabut seperti terlihat pada Gambar 3.2. Dalam proses pengumpulan data, penulis dibantu oleh Bapak Agus Budi Santoso selaku pakar untuk menentukan citra mana saja yang masuk ke dalam kategori yang tepat.

Gambar 3.1 Contoh citra Gunung Merapi kategori siang jelas

Gambar 3.2 Contoh citra Gunung Merapi berkategori siang kabut

Berdasarkan citra Siang jelas (Gambar 3.1) dan Siang kabut (Gambar 3.2), terdapat perbedaan yang signifikan. Citra siang jelas terdapat berbagai objek di dalam nya yaitu langit dan permukaan gunung. Warna dari langit dan warna dari permukaan gunung memiliki karakter nya sendiri dimana jika cerah warna langit akan berwarna biru dan permukaan gunung yang berwarna coklat. Begitu juga dengan citra siang kabut, objek pada citra tersebut dominan dengan kabut/awan yang memiliki karakter warna putih dan abu-abu. Berdasarkan analisis tersebut maka pada penelitian ini akan dilakukan ekstraksi ciri berdasarkan warna. Ada beberapa faktor alasan dilakukannya ekstraksi ciri warna saja, diantaranya yang pertama adalah ekstraksi ciri warna cukup untuk menangani kasus ini, yang kedua komputasi yang dilakukan tidak cukup berat karena hanya menggunakan ciri warna saja.

3.2 Pengelompokkan Data

Proses pengelompokkan data ini dilakukan terhadap hasil dari *preprocessing* yang merupakan representasi data berupa model ruang vektor. Data tersebut akan dikelompokkan menjadi dua kelas yaitu siang jelas (tidak dihapus) dan siang kabut (dihapus). Untuk lebih jelas mengenai karakteristik pengelompokkan citra dapat dilihat pada Tabel 3.1.

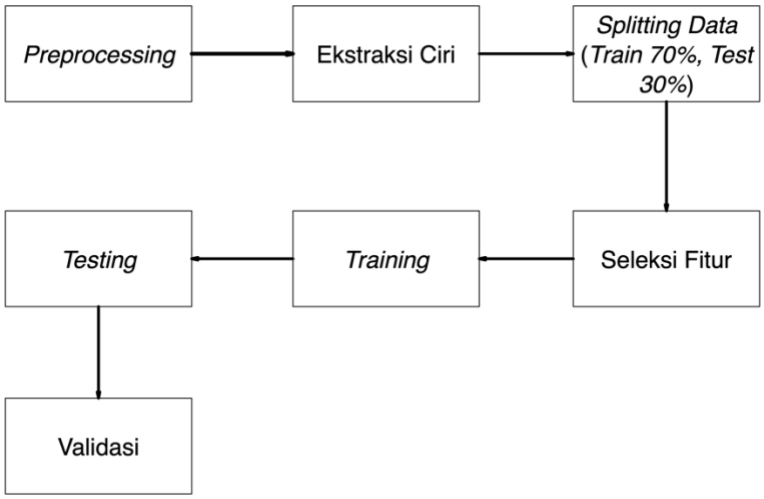
Tabel 3.1 Karakteristik citra berdasarkan kelompok

Citra	Kelompok	Keterangan

	Siang Jelas	Pada citra tersebut objek Gunung Merapi terlihat jelas / tidak terhalang oleh kabut/awan. Gambar tidak perlu dihapus/dipertahankan.
	Siang Kabut	Pada citra tersebut objek Gunung Merapi tidak terlihat jelas / terhalang oleh kabut / awan. Gambar perlu dihapus.

3.3 Rancangan Sistem

Secara umum rancangan sistem ini digambarkan ke dalam bentuk diagram tahapan. Tujuan dibuatnya diagram tahapan ini agar terlihat lebih terstruktur sehingga dapat membantu jalannya penelitian ini. Pada penelitian ini terdapat beberapa sub proses dari diagram tahapan yang akan dibahas pada masing-masing subbab. Untuk keseluruhan tahapan dari rancangan sistem ini digambarkan kedalam sebuah diagram yang dapat dilihat pada Gambar 3.3 Diagram Tahapan Rancangan Sistem.



Gambar 3.3 Diagram Tahapan Rancangan Sistem

3.4 *Preprocessing*

Dalam penelitian ini warna merupakan indikator dalam menentukan sebuah kelas dari citra, maka dari itu akan digunakan empat ruang warna yaitu RGB, CIELab, HSV, dan YCbCr. Keempat ruang warna tersebut memiliki masing-masing perspektif warna terhadap sebuah citra. Dari keempat ruang warna yang digunakan tersebut, akan dilihat ruang warna mana yang memiliki pengaruh kuat terhadap menentukan kelas pada citra. Berhubung data yang dikumpulkan memiliki ruang warna RGB, maka pada penilitian ini dilakukan tahap *preprocessing*, yang mana citra akan dikonversikan ke dalam tiga ruang warna lain yaitu HSV, CIELab, dan YCbCr.

3.5 Ekstraksi Ciri Warna

Setelah tahap *preprocessing* selanjutnya pada penelitian ini akan dilakukan ekstraksi ciri warna dari masing-masing ruang warna yang ada. Hal ini dilakukan agar data yang telah dikumpulkan dan dikonversi dapat diolah dengan model *Machine Learning* yang akan dirancang kemudian. Ekstraksi ciri warna merupakan sebuah proses dimana setiap citra dengan ruang warna yang sudah ada (RGB, HSV, CIELab, YCbCr), akan diambil cirinya yaitu berupa nilai minimum, maksimum, dan *mean* dari masing-masing *Channel* ruang warna (Contoh: *Red* minimum, *Red* maksimum). Penjelasan lebih detail terhadap ketiga ciri tersebut adalah sebagai berikut:

1. Nilai Minimum, merupakan salah satu ciri warna dengan mengambil nilai minimum dari sebuah *Channel* ruang warna dalam matriks warna citra. Sebagai contoh pada ruang warna RGB yang mana dalam ruang warna tersebut terdapat tiga *Channel* yaitu *Red, Green,* dan *Blue*. Dari masing-masing *Channel* diambil nilai minimum/terendah nya dalam matriks warna sebuah citra.
2. Nilai Maksimum, merupakan salah satu ciri warna dengan mengambil nilai maksimum dari sebuah *Channel* ruang warna dalam matriks warna citra. Dari masing-masing *Channel* ruang warna diambil nilai maksimum/tertinggi nya dalam matriks warna sebuah citra.
3. Nilai *Mean*, merupakan salah satu ciri warna dengan mengambil nilai *mean* dari sebuah *Channel* ruang warna dalam matriks warna citra. Dari masing-masing *Channel* ruang warna diambil nilai *mean*/rata-rata nya dalam matriks warna sebuah citra.

Sehingga hasil dari ektraksi ciri warna ini mendapatkan fitur sebanyak 36 dan 1 kolom kelas yang dapat dilihat pada Tabel 3.2.

Tabel 3.2 Atribut *Dataset*

Index	Nama Atribut	Color Space	Keterangan
0	rMin	RGB -> Red	rMin = nilai minimum dari red rMax = nilai maksimum dari red rAvg = nilai rata-rata dari red
1	rMax		
2	rAvg		
3	bMin	RGB -> Blue	bMin = nilai minimum dari blue bMax = nilai maksimum dari blue bAvg = nilai rata-rata dari blue
4	bMax		
5	bAvg		
6	gMin	RGB -> Green	gMin = nilai minimum dari green gMax = nilai maksimum dari green gAvg = nilai rata-rata dari green
7	gMax		
8	gAvg		
9	hMin	HSV -> Hue	hMin = nilai minimum dari hue hMax = nilai maksimum dari hue hAvg = nilai rata-rata dari hue
10	hMax		
11	hAvg		
12	sMin	HSV -> Saturation	sMin = nilai minimum dari saturation sMax = nilai maximum dari saturation sAvg = nilai rata-rata dari saturation
13	sMax		
14	sAvg		
15	vMin	HSV -> Value	vMin = nilai minimum dari value vMax = nilai maksimum dari value vAvg = nilai rata-rata dari value
16	vMax		
17	vAvg		
18	lMin	CieLAB -> Lightness	lMin = nilai minimum dari Lightness lMax = nilai maksimum dari Lightness lAvg = nilai rata-rata dari Lightness
19	lMax		
20	lAvg		
21	laMin	CieLAB -> a *component*	laMin = nilai minimum dari a component laMax = nilai maksimum dari a component laAvg = nilai rata-rata dari a component
22	laMax		
23	laAvg		
24	lbMin	CieLAB -> b *component*	lbMin = nilai minimum dari b component lbMax = nilai maksimum dari b component lbAvg = nilai rata-rata dari b component
25	lbMax		
26	lbAvg		
27	yMin	Y'CbCr -> Luma	yMin = nilai minimum dari Luma yMax = nilai maksimum dari Luma yAvg = nilai rata-rata dari Luma
28	yMax		
29	yAvg		
30	CbMin	Y'CbCr -> Kroma (B-Y)	CbMin = nilai minimum dari Kroma (B-Y) CbMax = nilai maksimum dari Kroma (B-Y) CbAvg = nilai rata-rata dari Kroma (B-Y)
31	CbMax		
32	CbAvg		
33	CrMin		

34	CrMax	Y'CbCr -> Kroma (R-Y)	CrMin = nilai minimum dari Kroma (R-Y) CrMax = nilai maksimum dari Kroma (R-Y) CrAvg = nilai rata-rata dari Kroma (R-Y)
35	CrAvg		
36	Target_name	-	Target *Class*

3.6 Penentuan Data *Training* dan *Testing*

Dalam penelitian ini, data *Training* dan *Testing* diperoleh dari citra Gunung Merapi BPPTKG Yogyakarta yang diambil pada tahun 2019 – 2020. Dari jumlah data yang diambil kemudian akan di *split* menjadi 70% data *training*, dan 30% data *testing* (Gambar 3.4). Data *Training* ini adalah sekumpulan data yang difungsikan dalam proses pelatihan model *Machine Learning*, sehingga jumlah data yang dimasukkan cenderung lebih banyak. Sedangkan Data *Testing* adalah sekumpulan data yang berfungsi dalam mengevaluasi model *Machine Learning* yang diukur dari tingkat keakuratan model tersebut.

Gambar 3.4 Ilustrasi Pembagian dataset

3.7 Seleksi Fitur

Seleksi fitur terhadap sebuah *dataset* dilakukan guna mendapatkan hasil yang lebih optimal dari segi akurasi maupun waktu komputasi. Dalam penelitian ini seleksi fitur yang digunakan adalah Chi-Square dan CFS. Masing-masing metode tersebut memiliki beberapa perbedaan dalam proses seleksi fitur diantarnya sebagai berikut:

a. Chi-Square

   Pada metode Chi-Square untuk melakukan proses seleksi fitur metode ini memiliki ketergantungan terhadap jenis model yang digunakan, selain itu pada Chi-Square juga dibutuhkan nilai k, yang mana nilai k ini merupakan jumlah sampel fitur yang digunakan. Setelah diberikan nilai k dan jenis model yang digunakan maka selanjutnya dilakukan proses *fitting* terhadap dataset dengan menginputkan nilai X (data independen) dan Y (data dependen). Hasil yang diberikan oleh metode ini berupa rekomendasi fitur apa saja yang digunakan sesuai dengan jumlah k (banyak fitur) yang diinputkan sebelumnya.

b. CFS

Berbeda halnya dengan metode Chi-Square, pada metode CFS secara langsung dilakukan proses *fitting* tanpa ada masukkan tambahan selain nilai X dan Y. Nilai X merupakan data indepeden yaitu berupa fitur (hasil ekstraksi ciri warna) dan nilai Y merupakan data dependen yaitu berupa target *output* (kelas). Metode CFS ini memberikan hasil rekomendasi fitur terbaik dengan cara melakukan *ranking* mulai dari fitur yang paling signifikan ke tidak signifikan. *Ranking* fitur pada CFS ini bertolok ukur pada seberapa bagus nilai meritnya.

Dari kedua metode seleksi fitur tersebut akan dilihat metode mana yang mampu menghasilkan kinerja yang lebih optimal.

3.8 *Training*

Untuk membangun sebuah model *Machine Learning* yang mampu memilah/klasifikasi data maka proses yang perlu dilakukan selanjutnya adalah *Training* model. Dalam penelitian ini metode *training* model dilakukan dengan metode *Supervised Learning.* Metode ini dilakukan dengan cara memberikan label kelas pada masing-masing data. Label merupakan sebuah tag dari data yang ditambahkan ke dalam model. Dengan adanya label ini maka proses *training* menjadi terawasi (*Supervised*).

Hasil kelas yang diberikan oleh *output* akan dikomparasikan dengan label dari *input* (label/kelas sebenarnya), sehingga model dapat dievaluasi ulang ketika hasil yang diberikan ternyata tidak sesuai. Untuk menerapkan metode *Supervised Learning*, selanjutnya dibutuhkan algoritma *Machine Learning* dalam melakukan proses klasifikasi data. Algoritma *Machine Learning* yang digunakan pada penelitian ini adalah *Support Vector Machine* (SVM) dan *K-Nearest Neighbour* (KNN). Selain itu dalam proses *Training* ini digunakan data *Training* sebanyak 70% dari total keseluruhan data citra. Dalam melakukan proses *training* model dilakukan beberapa skenario perancangan model sebagai berikut:

3.8.1 Perancangan Model SVM

Dalam perancangan model SVM dibutuhkan beberapa parameter masukkan. Pada penelitian ini parameter yang akan digunakan yaitu fungsi kernel, nilai C, dan Gamma. Penentuan parameter yang digunakan akan mempengaruhi hasil kinerja dari model tersebut. *Hypertuning* pada model SVM dilakukan dengan cara memberikan opsi nilai dari tiap parameter. Pada penelitian ini masing-masing parameter diberikan opsi nilai sebagai berikut:

a. Nilai C

   Opsi yang diberikan dalam melakukan pencarian nilai C yang tepat adalah 1, 100, dan 1000. Dari nilai tersebut akan dicari berapa nilai C yang tepat.

b. Fungsi Kernel

   Pada penelitian ini opsi fungsi kernel SVM yang akan dilakukan pencarian ialah RBF, polinominal, dan linear.

c. Nilai Gamma

   Opsi yang diberikan dalam melakukan pencarian nilai Gamma adalah 0.01, 0.001, dan 0.0001. Dari nilai yang diberikan tersebut akan dicari nilai mana yang tepat.

Pencarian Fitur Terbaik CFS

Pencarian fitur terbaik CFS dilakukan agar diketahui fitur mana saja yang tepat untuk digunakan sehingga mampu menciptakan model yang handal. Fitur yang terbaik dikatakan ketika semakin tinggi akurasi dan semakin kecil fiturnya maka fitur tersebut merupakan yang terbaik. Proses pencarian pada CFS dilakukan dengan cara sekuensial.

Pencarian Fitur Terbaik Chi-Square

Pencarian fitur terbaik Chi-Square dilakukan agar diketahui berapa jumlah k (banyak fitur) yang tepat untuk digunakan sehingga mampu menciptakan model yang handal. Fitur yang terbaik dikatakan ketika semakin tinggi akurasi dan semakin kecil fiturnya maka fitur tersebut merupakan yang terbaik. Proses pencarian pada CFS dilakukan dengan cara sekuensial dimana tiap nilai k akan diuji performanya.

3.8.2 Perancangan Model KNN

Dalam perancangan model KNN dibutuhkan parameter masukkan yaitu berupa nilai k. Nilai k ini merupakan jumlah tetangga terdekat. Penentuan berapa jumlah nilai k yang tepat akan mempengaruhi hasil kinerja dari model tersebut. Proses *Hypertuning* pada model KNN dilakukan dengan cara melakukan pencarian nilai k secara sekuensial menggunakan proses perulangan. Proses perulangan ini akan mencari jumlah nilai k mulai dari angka 1 sampai 36. Setiap proses perulangan akan memberikan nilai balikan berupa akurasi. Akurasi yang tertinggi akan menandakan bahwa nilai k tersebut merupakan jumlah yang tepat untuk digunakan.

Pencarian Fitur Terbaik CFS

Sama hal nya pada SVM pencarian fitur terbaik CFS dilakukan agar diketahui fitur mana saja yang tepat untuk digunakan sehingga mampu menciptakan model yang handal. Fitur yang terbaik dikatakan ketika semakin tinggi akurasi dan semakin kecil fiturnya maka fitur tersebut merupakan yang terbaik. Proses pencarian pada CFS dilakukan dengan cara sekuensial.

Pencarian Fitur Terbaik Chi-Square

Pencarian fitur terbaik Chi-Square dilakukan agar diketahui berapa jumlah k (banyak fitur) yang tepat untuk digunakan sehingga mampu menciptakan model yang handal. Fitur yang terbaik dikatakan ketika semakin tinggi akurasi dan semakin kecil fiturnya maka fitur tersebut merupakan yang terbaik. Proses pencarian pada CFS dilakukan dengan cara sekuensial dimana tiap nilai k akan diuji performanya.

3.9 *Testing*

Pada tahap pengujian model *Machine Learning*, dari model yang sudah dilatih sebelumnya akan dilakukan pengujian terhadap model tersebut dengan cara menggunakan data *testing* yang sudah dibagi sebelumnya yaitu sebanyak 30% dari keseluruhan data. Dari hasil pengujian ini akan dilihat seberapa baik performa yang dihasilkan ketika model dihadapkan dengan data yang baru. Jika hasil dari tahap *testing* tidak sesuai dengan harapan maka model tersebut perlu dilakukan optimasi lagi.

3.10 Validasi

Setelah model *Machine Learning* terbentuk maka tahapan selanjutnya adalah melakukan validasi model. Validasi model ini akan menyimpulkan seberapa baik performa yang dihasilkan oleh model *Machine Learning.* Validasi model diukur dengan menggunakan *Confusion Matrix. Confusion Matrix* mencakup nilai Akurasi, *Sensitivity,* dan *Specificity* dalam menunjukkan seberapa akurat model *Machine Learning* tersebut bekerja. Nilai akurasi dihasilkan dengan cara mengkomparasi hasil klasifikasi yang diberikan oleh model dengan label/kelas yang sebenarnya. Pada penelitian ini juga menerapkan *Cross-Validation* pada model *Machine Learning* tujuannya agar melihat seberapa baik model ini bekerja dalam menangani kasus di dunia nyata. *Cross-Validation* ini dilakukan dengan menggunakan *K-fold Cross-Validation,* dimana jumlah *fold* yang digunakan sebanyak lima *fold.*

3.11 Implementasi Model Pada *Website*

Model yang telah dibangun diharapkan dapat diimplementasikan secara langsung di kehidupan nyata. Oleh karena itu pada penelitian ini, model yang telah dibangun akan disematkan kedalam teknologi *website. Website* ini nantinya akan menjadi sebuah aplikasi *filtering* citra Gunung Merapi. Aplikasi *filtering* citra Gunung Merapi mampu memilah citra antara siang jelas dan siang kabut berdasarkan citra masukkan dari pengguna. Aplikasi ini dibangun dengan teknologi *micro web framework* yaitu bernama Flask. Flask sendiri berjalan dengan menggunakan Bahasa pemrograman Python sehingga cocok dengan model yang juga dibangun dengan Bahasa yang sama.

3.12 Metode Analisis Data

Untuk menunjang kegiatan analisis data pada penelitian ini digunakan beberapa *tools* dan *software* penunjang diantaranya adalah sebagai berikut:

a. Google Collab

   Google Collab merupakan *software* yang berbasis *online*, digunakan sebagai sebuah *platform* IDE dalam mengeksekusi kode program. Kelebihan dari *software* ini adalah komputasi yang digunakan sangat memadai dalam proses pengembangan *Machine Learning*. *Platform* ini dijalankan dengan menggunakan Bahasa pemrograman Python versi 3.7.

b. SkLearn

   SkLearn merupakan sebuah *library* dari python yang digunakan untuk melakukan pengujian model *Machine Learning*.

c. Pandas

   Pandas merupakan sebuah *library* dari python yang digunakan untuk memanipulasi data dan juga menganalisis data.

d. Matplotlib

   Matplotlib merupakan sebuah *library* dari python yang digunakan untuk melakukan visualisasi data.

e. Numpy

   Numpy adalah sebuah *library* dari python yang fungsinya membantu dalam melakukan proses perhitungan.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Penerapan Sistem

Dari hasil rancangan sistem yang sudah dilakukan sebelumnya maka selanjutnya adalah menerapkan rancangan sistem tersebut. Hasil penerapan sistem tersebut akan dibahas pada subbab selanjutnya sesuai dengan alur rancangan sistem. Sistem ini diterapkan dengan menggunakan Bahasa pemrograman Python dengan *platform* Google Collab dan juga *library* pendukung lainnya.

4.1.1 *Preprocessing Data*

Pada penelitian ini *preprocessing data* yang dilakukan adalah mengkonversi ruang warna dari sebuah citra. Untuk melakukan konversi warna citra dibutuhkan sebuah *library* Python yaitu adalah OpenCv. Pada dasarnya karena citra yang diperoleh memiliki ruang warna RGB, maka RGB akan dijadikan kondisi citra awal untuk kemudian dikonversi ke ruang warna lain. Pada OpenCv RGB dinotasikan sebagai BGR (*Blue, Green, Red*). BGR merupakan ruang warna yang sama dengan RGB perbedaanya hanya pada susunan *Channel* warna nya. Proses dalam mengkonversi ruang warna citra hal yang pertama dilakukan adalah membaca *file* citra yang disimpan ke dalam *variable* Gambar_merapi. Kemudian hasil dari konversi masing-masing ruang warna disimpan kedalam *variable* HSV_images, LAB_images, dan YCrCb_images.

4.1.2 Ekstraksi Ciri Warna

Untuk melakukan ekstraksi ciri warna tiap file gambar/citra, setiap *Channel* dari ruang warna dilakukan perhitungan. Dimana perhitungan yang dimaksud adalah dengan cara mengambil nilai minimum, maksimum, dan *mean* dari tiap *Channel* ruang warna. Masing-masing nilai tersebut disimpan ke dalam *variable.*

Mengubah Tipe Data dan *Array*

Hasil perhitungan dari *Channel* warna disimpan ke dalam masing-masing *variable.* Proses selanjutnya yaitu menggabungkan *variable* tersebut ke dalam bentuk *array.* Proses ini dapat diimplementasikan dengan menggunakan Numpy. Ketika *variable* ini digabungkan ke

dalam *array* tipe data nya akan menjadi *object*, tipe data tersebut tidak akan bisa di proses oleh *Machine Learning* sehingga perlu diubah tipe datanya menjadi *float.*

Mengubah *Array* menjadi *Dataframe*

Proses berikutnya, data yang telah tersimpan ke dalam *array* akan diubah menjadi sebuah *Dataframe*. Tujuan diubahnya *array* menjadi *Dataframe* agar memudahkan dalam melakukan proses analisis data dan juga presentasi data menjadi terlihat lebih menarik. Dari sini penulis telah mendapatkan sebuah *Dataframe* yang siap untuk dimasukkan ke dalam model *Machine Learning*. *Dataframe* tersebut menghasilkan total atribut sebanyak 37 atribut dimana terdiri dari 36 fitur dan satu target kelas.

4.1.3 *Splitting Data*

Setelah proses ekstraksi ciri dilakukan maka tahapan selanjutnya adalah melakukan proses *Splitting Data.* Pada pembahasan 3.6 telah ditentukan bahwa pembagian data *training* dan *testing* adalah 70%, dan 30%. Untuk melakukan *Splitting Data* digunakan sebuah *method* dari Sklearn yang bernama *train_test_split().* Yang mana secara kode program dapat diimplementasikan seperti yang terlihat pada Gambar 4.1 dan hasil pembagiannya dapat dilihat pada Gambar 4.2.

```
# Proses splitting data dengan test size sejumlah 0.3(30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=4)
# Cetak ukuran data train
print ('Train set:Server, X_train.shape,  y_train.shape)
# Cetak ukuran data test
print ('Test set:Server, X_test.shape,  y_test.shape)
```

Gambar 4.1 Sintaks program *Splitting data* dengan *train_test_split()*

```
Train set: (70, 36) (70,)
Test set: (30, 36) (30,)
time: 26 ms
```

Gambar 4.2 Hasil *Splitting data*

Berdasarkan Gambar 4.2, dari total 100 data yang ada sebelumnya, setelah dilakukan proses *Splitting data* hasilnya menjadi 70 data untuk *Train* set dan 30 data untuk *Test set.* Setelah mendapatkan data train_set dan juga test_set maka proses selanjutnya adalah membangun model sesuai dengan skenario yang dirancang.

4.1.4 Seleksi Fitur

Untuk menunjang performa yang dihasilkan dari model *Machine Learning* maka selanjutnya dilakukan proses seleksi fitur. Dalam penelitian ini metode seleksi fitur yang digunakan adalah CFS dan Chi-Square. Bagaimana proses pengimplementasian metode seleksi fitur ini dilakukan akan dibahas pada subbab berikut.

CFS

Dalam mengimplementasikan metode seleksi fitur CFS digunakan sebuah *library* bernama Skfeature. Metode CFS memerlukan sebuah masukkan parameter yaitu nilai X dan y. Nilai X dan y yang digunakan dalam penelitian ini adalah hasil dari *splitting data* yang dilakukan sebelumnya yaitu X_train dan y_train. Dari hasil rekomendasi tersebut perlu dilakukan pencarian lagi fitur mana saja yang tepat untuk digunakan pada model *Machine Learning.* Untuk proses pencarian fitur mana saja yang diambil akan dibahas pada bagian pelatihan (*training*) model.

Chi-Square

Pada metode Chi-square, untuk mengimplementasikannya ke dalam kode program dibutuhkan *library* dari Sklearn. Berhubung Chi-square memiliki ketergantungan terhadap algoritma *Machine Learning* yang digunakan maka pada penelitian ini metode Chi-Square akan diterapkan ke dalam dua algoritma yaitu SVM dan KNN. Sama halnya dengan metode CFS, nilai X dan y menggunakan X_train dan y_train. Sebelum menjalankan metode seleksi fitur ini dibutuhkan masukkan berupa nilai k (banyak fitur). Untuk menentukan jumlah nilai k yang terbaik dibutuhkan proses pencarian nilai k. Proses pencarian nilai k ini selanjutnya akan dibahas pada bagian pelatihan (*training*) model.

4.1.5 *Training* Model

Dalam proses *training* model akan dibangun sebuah model sesuai dengan skenario/perancangan model yang telah disebutkan pada subbab 3.8. Proses pelatihan ini akan melibatkan dua algoritma *Machine Learning* yaitu SVM dan KNN. Selama proses pelatihan, terdapat beberapa tahapan lainnya juga yang akan dibahas pada subbab ini yaitu proses pencarian fitur terbaik dari masing-masing metode seleksi fitur dan juga proses *Hypertuning* guna mengoptimalkan kinerja dari model *Machine Learning* yang dibangun. Untuk proses pembangunan model akan dibahas pada subbab selanjutnya.

Pembangunan Model SVM

Untuk membangun sebuah model dengan algoritma SVM, dibutuhkan sebuah *library* dari Sklearn. Untuk mengoptimasi model SVM, nilai dari parameter SVM menentukan performa dari model tersebut. Untuk itu selanjutnya dilakukan proses *Hypertuning* sehingga menghasilkan parameter yang tepat dalam menghasilkan model yang handal. Proses *Hypertuning* pada SVM dapat dibantu dengan menggunakan *tools* dari Sklearn yaitu bernama GridSearchCV. Untuk melakukan proses *Hypertuning* dengan GridSearchCV dibutuhkan masukkan berupa perkiraan nilai parameter yang cocok pada model yang akan dibangun. Pada penelitian ini terdapat tiga parameter yang akan dilakukan pencarian yaitu parameter C, Gamma, dan Kernel. Hasil dari pencarian parameter menggunakan GridSearchCV menghasilkan keluaran seperti yang terlihat pada Gambar 4.3.

```
GridSearchCV(cv=None, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid=[{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)time: 70.6 ms
```

Gambar 4.3 Hasil keluaran GridSearchCV

Berdasarkan Gambar 4.3, nilai parameter yang direkomendasikan oleh GridSearchCV adalah C=1.0, Gamma=’scale’, dan kernel=’rbf’. Setelah mendapatkan parameter yang tepat maka selanjutnya adalah melakukan proses *fitting* model dengan data *training.* Proses *fitting* dilakukan dengan memasukkan nilai X_train dan y_train dengan jumlah fitur sebanyak 36.

Dari hasil *fitting* model, model tersebut diuji dengan menggunakan data *train* itu sendiri. Hasil akurasi yang dihasilkan oleh model tersebut dengan menggunakan data *training* adalah sebesar 100%, yang menandakan model tersebut tidak mengalami kesalahan dalam melakukan proses klasifikasi. Setelah model dilakukan proses *Hypertuning* maka tahap selanjutnya adalah melakukan pencarian fitur yang tepat dari seleksi fitur.

Seleksi Fitur CFS Model SVM

Pada subbab 4.1.4, sebelumnya sudah dibahas terkait pengimplementasian kode program dari metode seleksi fitur CFS. Namun dari hasil rekomendasi yang diberikan oleh CFS perlu dilakukan pencarian fitur mana saja yang diambil guna mengoptimalkan performa dari model. Untuk melakukan pencarian fitur terbaik dari CFS, maka pada penelitian metode pencarian fitur ini dilakukan secara sekuensial. Pencarian secara sekuensial ini dilakukan dengan cara menerapkan algoritma perulangan. Perulangan dilakukan sebanyak 36 kali sebanyak total keseluruhan fitur yang dimiliki. Setiap perulangan ini akan menambahkan masing-masing fitur sesuai dengan urutan yang direkomendasikan oleh CFS sebelumnya. Dari setiap fitur yang diuji akan menghasilkan akurasi dan waktu komputasi. Pada kondisi ini dikatakan fitur yang tepat untuk digunakan ketika akurasi yang dihasilkan tinggi dengan jumlah fitur yang lebih kecil. Hasil pencarian fitur terbaik CFS pada model SVM mendapatkan akurasi dan waktu komputasi yang cukup bervariasi, yang mana bisa dilihat pada Tabel 4.1.

Tabel 4.1 Jumlah fitur beserta akurasi dan waktu komputasi CFS

Jumlah Fitur	Akurasi (%)	Waktu Komputasi (Detik)
1	81	0.000218
2	81	0.000201
3	86	0.000238
4	86	0.000239
5	86	0.000243
6	89	0.000196
7	86	0.000247
8	87	0.000312
9	90	0.000216
10	89	0.000212
11	89	0.000229
12	91	0.000225

Jumlah Fitur	Akurasi (%)	Waktu Komputasi (Detik)
19	100	0.000252
20	89	0.000322
21	100	0.000386
22	91	0.000278
23	100	0.000312
24	100	0.000325
25	100	0.000266
26	100	0.000322
27	100	0.000318
28	100	0.000296
29	100	0.000346
30	100	0.000355

13	91	0.000244	31	100	0.000485
14	93	0.000261	32	100	0.000366
15	99	0.000218	33	100	0.000385
16	97	0.000217	34	100	0.000351
17	99	0.000253	35	100	0.000368
18	100	0.000265	36	100	0.000361

Berdasarkan pada Tabel 4.1, ditunjukkan bahwa jumlah fitur yang terbaik dari segi akurasi dan waktu komputasi adalah berjumlah 18 fitur dengan akurasi senilai 100% dan waktu komputasi 0.000265 detik.

Seleksi Fitur Chi-Square Model SVM

Sama halnya dengan CFS, sebelumnya sudah dibahas juga terkait pengimplementasian kode program dari metode seleksi fitur Chi-Square. Dalam metode Chi-Square akan dilakukan pencari nilai k (banyak fitur) yang tepat untuk digunakan. Proses pencarian nilai k ini dilakukan dengan cara sekuensial dimana dibutuhkan algoritma perulangan. Sama halnya pada CFS perulangan dilakukan sebanyak 36 kali yang mana jumlah perulangan tersebut merupakan total keseluruhan jumlah fitur. Setiap proses iterasi nilai k akan terus bertambah dengan nilai satu. Dari nilai k tersebut akan dilihat nilai k mana yang mampu menghasilkan akurasi tetinggi dan waktu komputasi yang singkat. Dari hasil pencarian fitur terbaik Chi-Square pada model SVM ditunjukkan akurasi dan waktu komputasi dari masing-masing fitur, yang mana bisa dilihat pada Tabel 4.2.

Tabel 4.2 Jumlah fitur beserta akurasi dan waktu komputasi Chi-Square

Jumlah k	Akurasi (%)	Waktu Komputasi (Detik)	Jumlah k	Akurasi (%)	Waktu Komputasi (Detik)
1	77	0.000683	19	100	0.000659
2	79	0.001067	20	100	0.000739
3	79	0.000637	21	100	0.000723
4	97	0.000684	22	100	0.000694
5	96	0.000629	23	90	0.000836
6	93	0.000755	24	100	0.000729
7	94	0.000743	25	100	0.000800
8	96	0.000744	26	91	0.000873
9	90	0.000684	27	100	0.001171
10	91	0.000713	28	100	0.000866

11	98	0.000668	29	100	0.000822
12	96	0.000685	30	100	0.000786
13	99	0.000742	31	100	0.000871
14	99	0.000759	32	100	0.000816
15	99	0.000660	33	100	0.000979
16	99	0.000768	34	100	0.000850
17	100	0.000753	35	100	0.000813
18	100	0.001285	36	100	0.000911

Berdasarkan pada Tabel 4.2, ditunjukkan bahwa jumlah fitur yang terbaik dari segi akurasi dan waktu komputasi adalah berjumlah 17 fitur dengan akurasi senilai 100% dan waktu komputasi 0.000753 detik.

Pembangunan Model KNN

Untuk membangun sebuah model dengan algoritma KNN, dibutuhkan sebuah *library* dari Sklearn. Untuk mengoptimasi model KNN, nilai dari parameter KNN menentukan performa dari model tersebut. Tahapan selanjutnya dilakukan proses *Hypertuning* pada model KNN sehingga menghasilkan parameter yang tepat dalam mengoptimalkan performa model. Proses *Hypertuning* pada KNN dilakukan dengan cara melakukan pencarian nilai k (jumlah tetangga terdekat). Pencarian nilai k menggunakan proses pencarian sekuensial dengan algoritma perulangan. Pada penelitian ini proses perulangan dilakukan sebanyak 12 kali, jumlah perulangan ini juga merepresentasikan jumlah nilai k. Dihasilkan akurasi beserta waktu komputasi dari masing-masing nilai k, yang mana hasil tersebut dapat dilihat pada Tabel 4.3.

Tabel 4.3 Jumlah k beserta akurasi dan waktu komputasi model KNN

Jumlah k	Akurasi (%)	Waktu Komputasi (Detik)
1	100	0.005091
2	94	0.003432
3	90	0.002720
4	84	0.002511
5	89	0.003076
6	80	0.002629
7	81	0.002899
8	80	0.003024
9	77	0.002820

10	74	0.002803
11	73	0.002843
12	73	0.002722

Berdasarkan hasil akurasi dan waktu komputasi pada Tabel 4.3., maka jumlah k yang tepat untuk digunakan adalah bernilai satu, dimana jumlah k=1 memiliki tingkat akurasi tertinggi yaitu sebesar 100% dengan waktu komputasi 0.005091 detik. Kemudian setelah mendapatkan nilai k yang tepat pada model KNN, maka selanjutnya dilakukan proses pelatihan dengan metode seleksi fitur.

Seleksi Fitur CFS Model KNN

Sama halnya dengan metode seleksi fitur CFS pada model SVM, pada model KNN juga akan dilakukan pencarian fitur terbaik. Fitur mana saja yang mampu menghasilkan performa terbaik dengan menggunakan model KNN. Untuk melakukan pencarian fitur terbaik dari CFS sama dengan proses pencarian fitur pada SVM, yang mana metode pencarian fitur ini dilakukan secara sekuensial. Pencarian secara sekuensial ini dilakukan dengan cara menerapkan algoritma perulangan. Untuk pengimplementasian kode program pencarian fitur terbaik CFS dengan model KNN dapat dilihat pada Tabel 4.4.

Tabel 4.4 Jumlah fitur beserta akurasi dan waktu komputasi CFS KNN

Jumlah fitur	Akurasi (%)	Waktu Komputasi (Detik)
1	81	0.004983
2	97	0.002443
3	100	0.002061
4	100	0.002319
5	100	0.003201
6	100	0.001878
7	100	0.001997
8	100	0.002107
9	100	0.001943
10	100	0.002034
11	100	0.002933
12	100	0.002596
13	100	0.002888
14	100	0.002379

Jumlah fitur	Akurasi (%)	Waktu Komputasi (Detik)
19	100	0.002248
20	100	0.002666
21	100	0.005118
22	100	0.002248
23	100	0.002375
24	100	0.005757
25	100	0.004168
26	100	0.003468
27	100	0.002165
28	100	0.002163
29	100	0.002481
30	100	0.002530
31	100	0.002245
32	100	0.002188

15	100	0.002669	33	100	0.003843
16	100	0.002396	34	100	0.001996
17	100	0.003899	35	100	0.002194
18	100	0.003053	36	100	0.002195

Berdasarkan pada Tabel 4.4, ditunjukkan bahwa jumlah fitur yang terbaik dari segi akurasi dan waktu komputasi adalah berjumlah 3 fitur dengan akurasi senilai 100% dan waktu komputasi 0.002061 detik.

Seleksi Fitur Chi-Square Model KNN

Secara sistematis proses pencarian nilai k Chi-Square pada model KNN sama dengan pencarian nilai k pada model SVM. Pencarian nilai k pada model KNN juga menggunakan algoritma perulangan sebanyak 36 kali. Dari hasil pencarian fitur terbaik Chi-Square pada model KNN ditunjukkan akurasi dan waktu komputasi dari masing-masing fitur, yang mana bisa dilihat pada Tabel 4.5.

Tabel 4.5 Jumlah k beserta akurasi dan waktu komputasi Chi-Square KNN

Jumlah k	Akurasi (%)	Waktu Komputasi (Detik)	Jumlah k	Akurasi (%)	Waktu Komputasi (Detik)
1	70	0.002071	19	73	0.001843
2	70	0.001926	20	73	0.001944
3	66	0.001686	21	73	0.001894
4	73	0.001653	22	73	0.001858
5	77	0.002021	23	77	0.002078
6	77	0.002069	24	77	0.002025
7	77	0.002037	25	77	0.002338
8	77	0.001913	26	87	0.002006
9	77	0.002050	27	90	0.002136
10	77	0.001977	28	90	0.002066
11	70	0.001910	29	90	0.002515
12	70	0.002116	30	90	0.002039
13	70	0.001926	31	90	0.002082
14	70	0.003273	32	90	0.002209
15	70	0.002152	33	90	0.001983
16	77	0.001983	34	90	0.002036
17	73	0.002066	35	90	0.002034
18	77	0.002066	36	90	0.002013

Berdasarkan pada Tabel 4.5, ditunjukkan bahwa jumlah fitur yang terbaik dari segi akurasi dan waktu komputasi adalah berjumlah 27 fitur dengan akurasi senilai 90% dan waktu komputasi 0.002136 detik. Akurasi ini dihasilkan dengan menggunakan data *training* untuk selanjutnya akan dibuktikan pada tahap pengujian apakah jumlah fitur ini sama hasilnya ketika digunakan data *testing.*

4.1.6 *Testing* Model

Dalam proses *testing* model akan dilakukan pengujian terhadap model yang telah dilatih sebelumnya. Pengujian ini dilakukan dengan cara menguji model yang terlatih dengan menggunakan data *testing.* Model yang diuji tersebut akan dilihat performa yang dihasilkan apakah sama dengan performa saat melakukan *training.* Pengujian model dilakukan terhadap model yang tidak menggunakan metode seleksi fitur dan model yang menggunakan metode seleksi fitur. Secara detail, proses pengujian akan dijelaskan pada subbab berikutnya.

Pengujian Model SVM

Model SVM yang telah dioptimasi dengan *Hypertuning,* selanjutnya akan dilakukan pengujian dengan menggunakan data *testing/*data baru. Total fitur yang diujikan pada model ini sejumlah 36 fitur. Model SVM yang telah dioptimasi dengan *Hypertuning,* selanjutnya akan dilakukan pengujian dengan menggunakan data *testing/*data baru. Berdasarkan hasil pengujian model SVM dengan data *testing,* akurasi yang dihasilkan senilai 90% dengan waktu komputasi 0.002015 detik.

Pengujian Model SVM dengan CFS

Dari metode CFS yang diterapkan pada model SVM akan dilakukan pengujian dengan menggunakan data *testing.* Proses pengujian seleksi fitur CFS dilakukan dengan menggunakan hasil seleksi fitur pada tahap *training* yaitu sebanyak 18 fitur. Hasil pengujian metode CFS dengan 18 fitur pada model SVM mendapatkan akurasi sebesar 80% dan waktu komputasi 0.000162 detik.

Pengujian Chi-Square dengan Model SVM

Pengujian pada metode Chi-Square juga dilakukan dengan cara mengujikan data *testing.* Proses pengujian ini menggunakan hasil seleksi fitur Chi-Square pada tahap *training.* Dari hasil

pengujian Chi-Square berjumlah 17 fitur pada model SVM dihasilkan akurasi senilai 83% dan waktu komputasi 0.000686 detik.

Pengujian Model KNN

Sebelum nya pada tahap *training* model telah didapatkan nilai k yang tepat untuk model KNN ini. Untuk selanjutnya model KNN yang sudah dibangun akan diuji dengan menggunakan data *testing* dimana jumlah k yang digunakan adalah k=1. Berdasarkan kode program pada **Error! Reference source not found.**, dihasilkan akurasi beserta waktu komputasi dari nilai k=1 adalah sebesar 90% dan waktu selama 0.005749 detik.

Seleksi Fitur CFS Model KNN

Sama halnya dengan metode seleksi fitur CFS pada model SVM, pada model KNN juga akan dilakukan pengujian fitur terbaik. Untuk melakukan pengujian fitur terbaik dari CFS, model akan diuji dengan menggunakan data *testing* dimana jumlah fitur yang digunakan adalah sebanyak 3 fitur. Hasil akurasi dan waktu komputasi yang didapatkan oleh metode seleksi fitur CFS pada model KNN adalah sebesar 83% dengan lama komputasi 0.001187 detik.

Pengujian Chi-Square Model KNN

Pengujian metode Chi-Square akan dilakukan dengan menggunakan jumlah nilai k yang sudah didapatkan pada proses *training* yaitu k=27 (27 fitur). Model ini akan diuji dengan menggunakan data *testing.* Dari hasil pengujian fitur terbaik Chi-Square pada model KNN ditunjukkan akurasi dan waktu komputasi 27 fitur adalah senilai 90% dan waktu komputasi 0.001822 detik.

Hasil Pengujian

Setelah dilakukannya proses pengujian pada masing-masing model dengan *Hypertuning* dan Feature Selection maka selanjutnya penulis akan mencoba untuk membandingkan dari setiap proses pengujian yang telah dilakukan, pengujian mana yang memiliki performa terbaik dari segi akurasi dan waktu komputasi. Dari hasil semua skenario perancangan model bisa dilihat pada Tabel 4.6.

Tabel 4.6 Rangkuman keseluruhan pengujian model

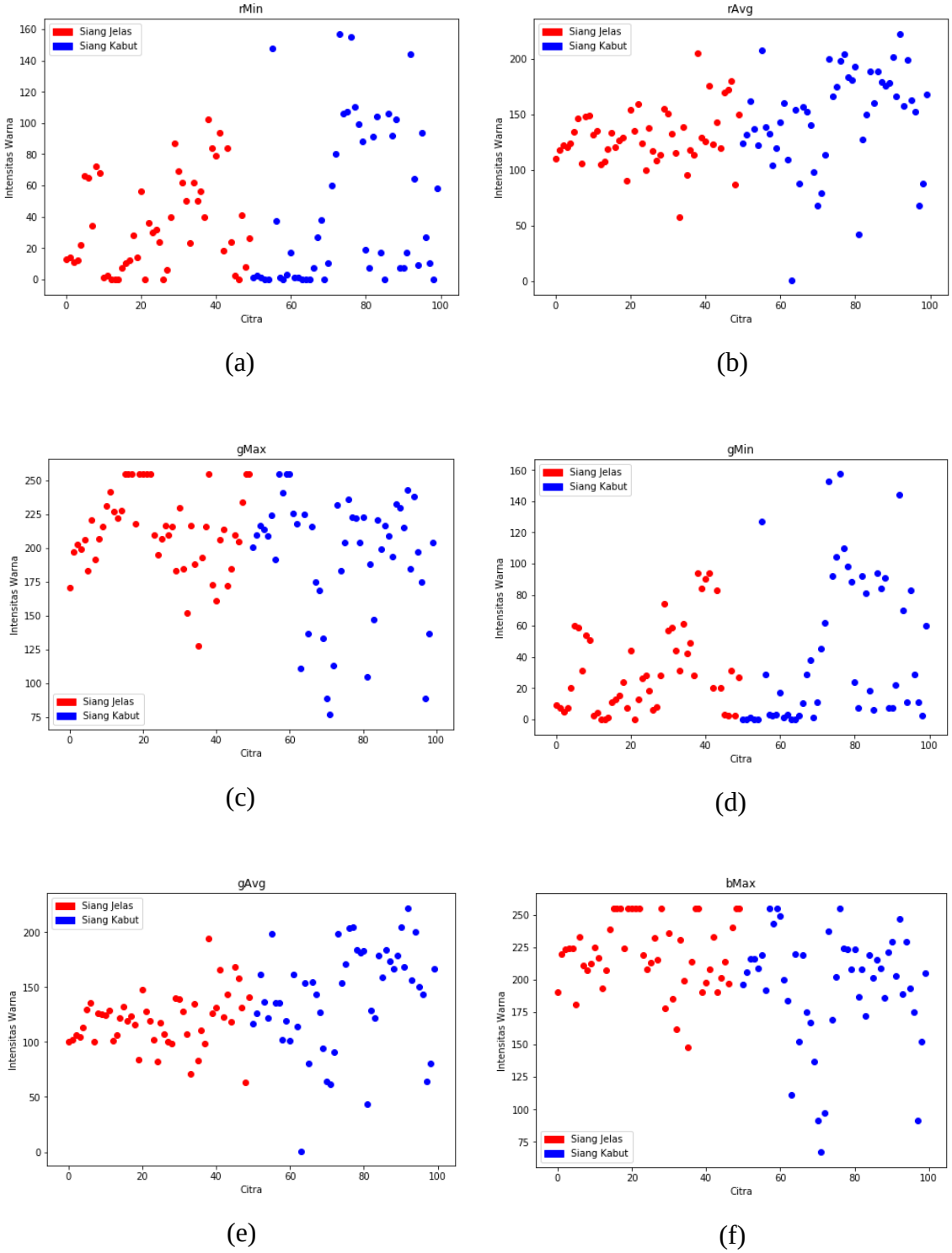
Classifier	Parameter	Jumlah Fitur	Waktu Komputasi Test Data Baru (1 sampel/Detik)	Akurasi (%)
KNN	(k = 1)	36	0.002238	90
KNN	(k = 1, Chi2)	27	0.001822	90
KNN	(k = 1, CFS)	3	0.001187	83
SVM	(kernel = 'rbf', C = 1.0, Gammal = ‘scale’)	36	0.002015	90
SVM	(kernel = 'rbf', C = 1.0, Gammal = ‘scale’), Chi2	17	0.000686	83
SVM	(kernel = 'rbf', C = 1.0, Gammal = ‘scale’), CFS	18	0.000162	80

Sementara untuk detail fitur apa saja yang digunakan secara berurut dapat dilihat pada Tabel 4.7.

Tabel 4.7 Detail atribut terpilih

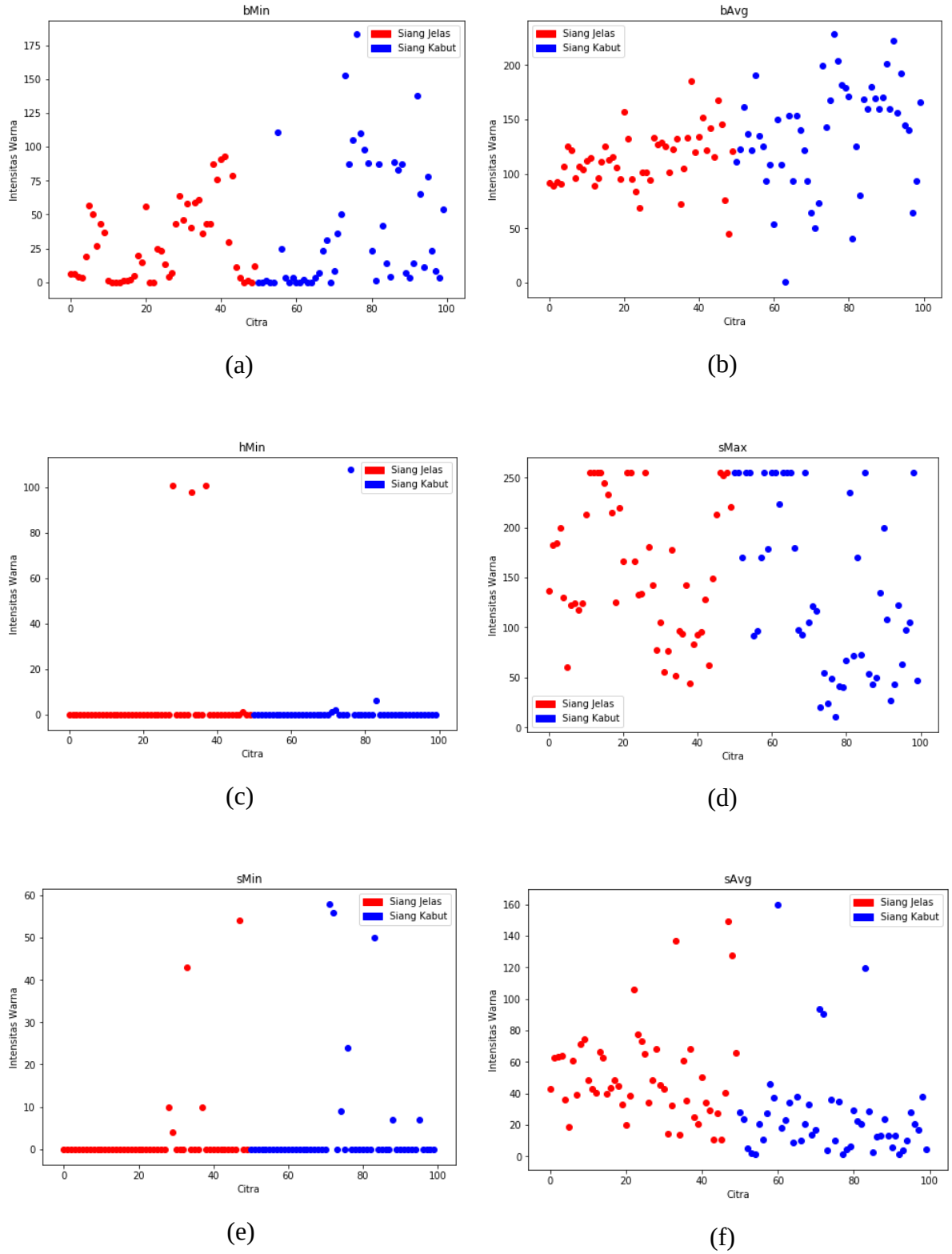
Classifier	Jumlah Fitur	Nama Fitur
KNN & SVM	36	'rMax', 'rMin', 'rAvg', 'gMax', 'gMin', 'gAvg', 'bMax', 'bMin', 'bAvg', 'hMax', 'hMin', 'hAvg', 'sMax', 'sMin', 'sAvg', 'vMax', 'vMin', 'vAvg', 'lMax', 'lMin', 'lAvg', 'laMin', 'laMax', 'laAvg', 'lbMin', 'lbMax', 'lbAvg', 'yMax', 'yMin', 'yAvg', 'cbMax', 'cbMin', 'cbAvg', 'crMax', 'crMin', 'crAvg'
KNN (Chi2)	27	rMin, rAvg, gMax, gMin, gAvg, bMax, bMin, bAvg, hMin, sMax, sMin, sAvg, vMax, vMin, vAvg, lMin, lAvg, laMax, lbMin, lbMax, yMax, yMin, yAvg, cbMax, cbMin, crMax, crMin
KNN (CFS)	3	'crMax', 'cbMax', 'yMax'
SVM (Chi2)	17	rMin, rAvg, gMin, gAvg, bMin, bAvg, hMin, sMax, sAvg, vMin, vAvg, lMin, lAvg, lbMin, yMin, yAvg, crMax
(CFS)	18	'crMax', 'cbMax', 'yMax', 'laMax', 'lbMin', 'laMin', 'lMax', 'vMax', 'sMax', 'lbMax', 'hMax', 'bMax', 'gMax', 'rMax', 'crMin', 'yAvg', 'lMin', 'gAvg', 'hMin'

Jika merujuk pada Tabel 4.6, model perancangan yang memiliki akurasi tinggi untuk digunakan adalah model KNN dengan menggunakan seleksi fitur Chi-Square. Hasil yang didapatkan oleh KNN dan Chi-Square bernilai akurasi = 90%, waktu komputasi = 0.001822 detik, dan jumlah fitur = 27. Chi-Square merekomendasikan fitur sebanyak 26, dengan fitur yang digunakan ialah 'rMin', 'rAvg', ‘gMax’, 'gMin', 'gAvg', 'bMax', 'bMin', 'bAvg', 'hMin', 'sMax', ‘sMin’, 'sAvg', 'vMax', 'vMin', ‘vAvg’, 'lMin', 'lAvg', ‘laMax’, 'lbMin', ‘lbMax’, ‘yMax’,'yMin', 'yAvg', ‘cbMax’, ‘cbMin’, ‘crMax’, ‘crMin’. Dari 27 fitur tersebut akan di visualisasikan data dari tiap fiturnya ke dalam bentuk Scatterplot untuk mengamati perseberan datanya. Untuk hasil visualisasi fitur 'rMin', 'rAvg', ‘gMax’, 'gMin', ‘gAvg’, ‘gMax’ dapat dilihat pada Gambar 4.4.



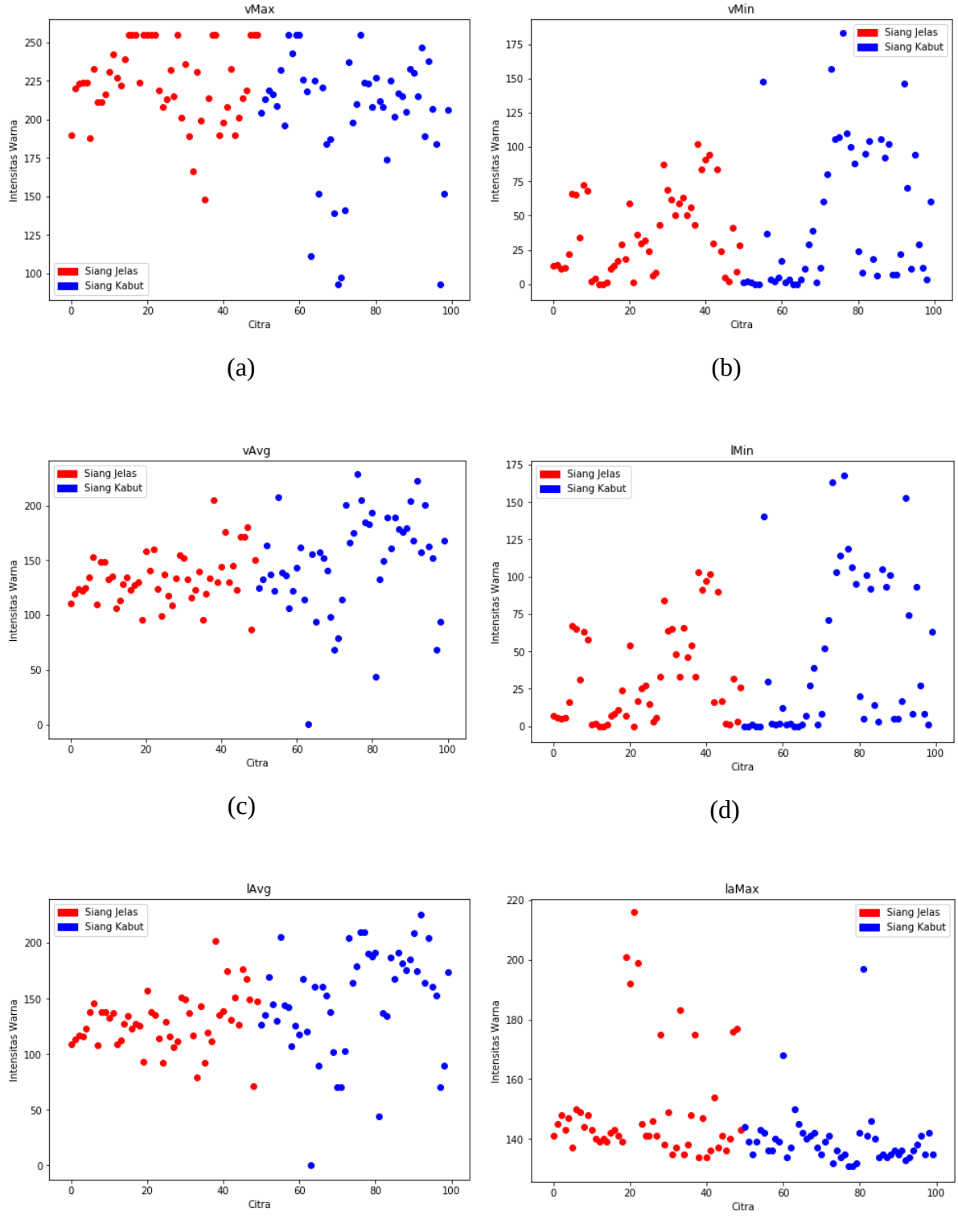
Gambar 4.4 Hasil visualisasi data : (a). rMin, (b). rAvg, (c). gMax, (d). gMin, (e). gAvg, (f). bMax

Hasil visualisasi fitur 'bMin', 'bAvg', ‘hMin’, 'sMax', ‘sMin’, ‘sAvg’ dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil visualisasi data : (a). bMin, (b). bAvg, (c). hMin, (d). sMax, (e). sMin, (f). sAvg

Hasil visualisasi fitur 'vMax', 'vMin', ‘vAvg’, 'lMin', ‘lAvg’, ‘laMax’ dapat dilihat pada Gambar 4.6.



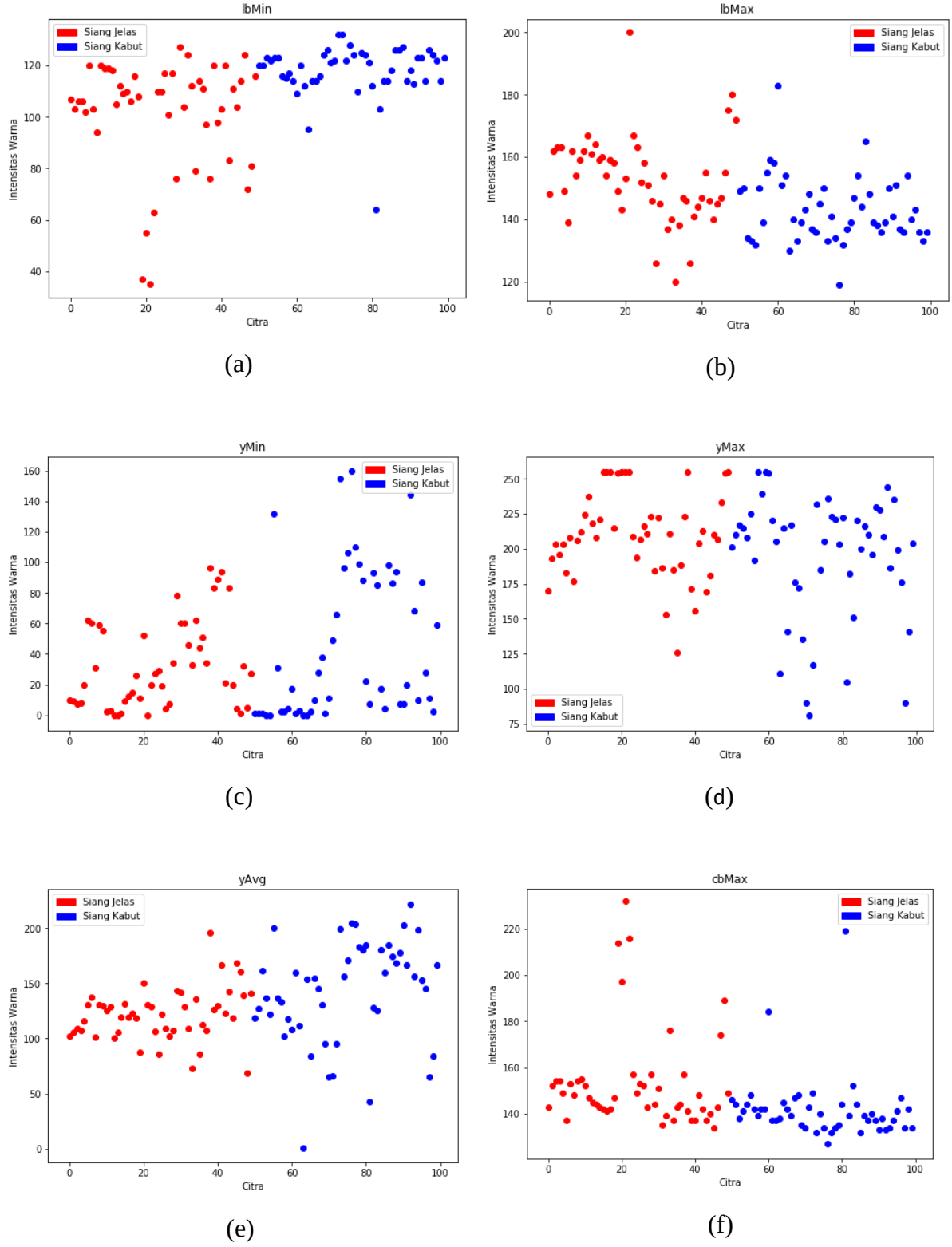
(a) (b)

(c) (d)

(e) (f)

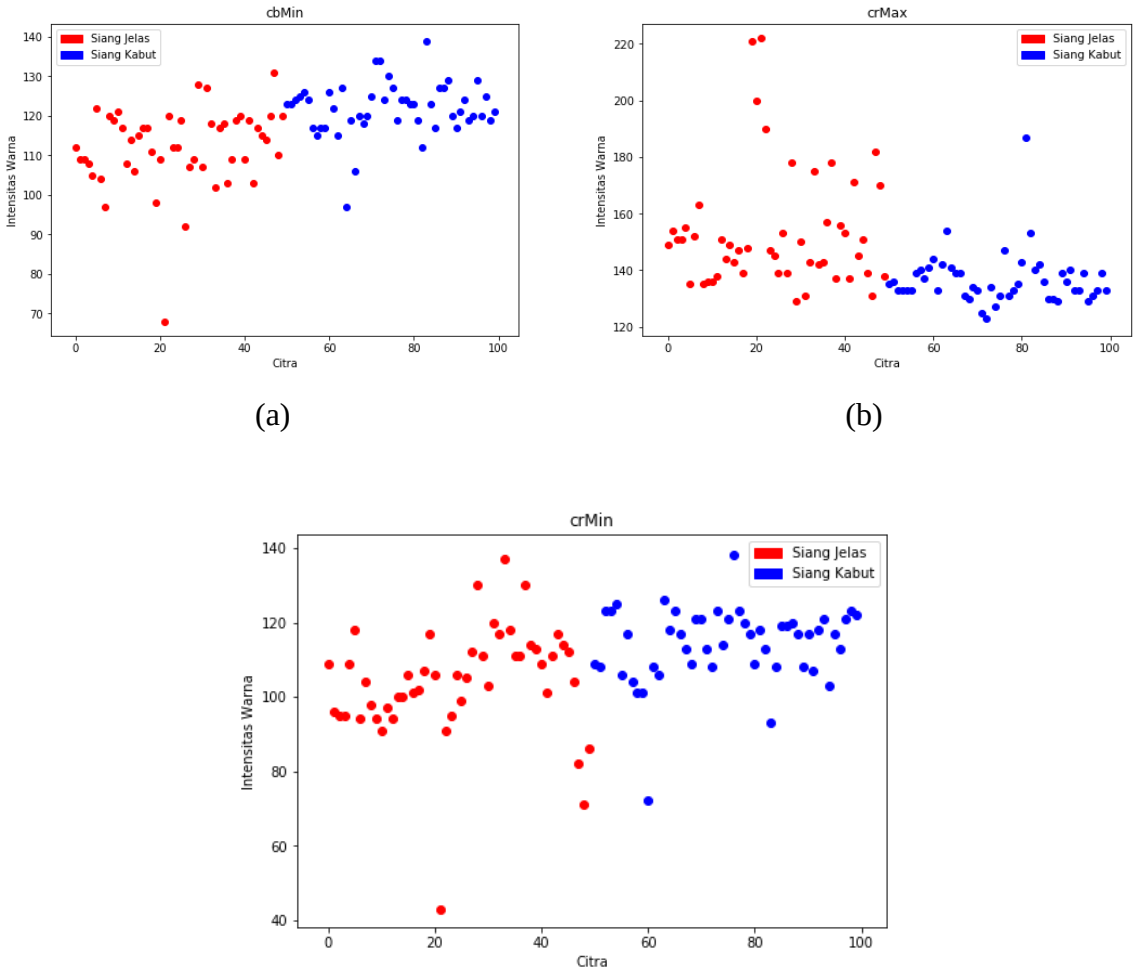
Gambar 4.6 Hasil visualisasi data : (a). vMax, (b). vMin, (c). vAvg, (d). lMin, (e). lAvg, (f). laMax

Hasil visualisasi fitur 'lbMin', 'lbMax', ‘yMin’, 'yMax', ‘yAvg’, ‘cbMax’ dapat dilihat pada Gambar 4.7.



Gambar 4.7 Hasil visualisasi data : (a). lbMin, (b). lbMax, (c). yMin, (d). yMax, (e). yAvg, (f). cbMax

Hasil visualisasi fitur 'cbMin', 'crMax', ‘crMin’ dapat dilihat pada Gambar 4.8.



(a) (b)

(c)

Gambar 4.8 Hasil visualisasi data : (a). cbMin, (b). crMax, (c). crMin

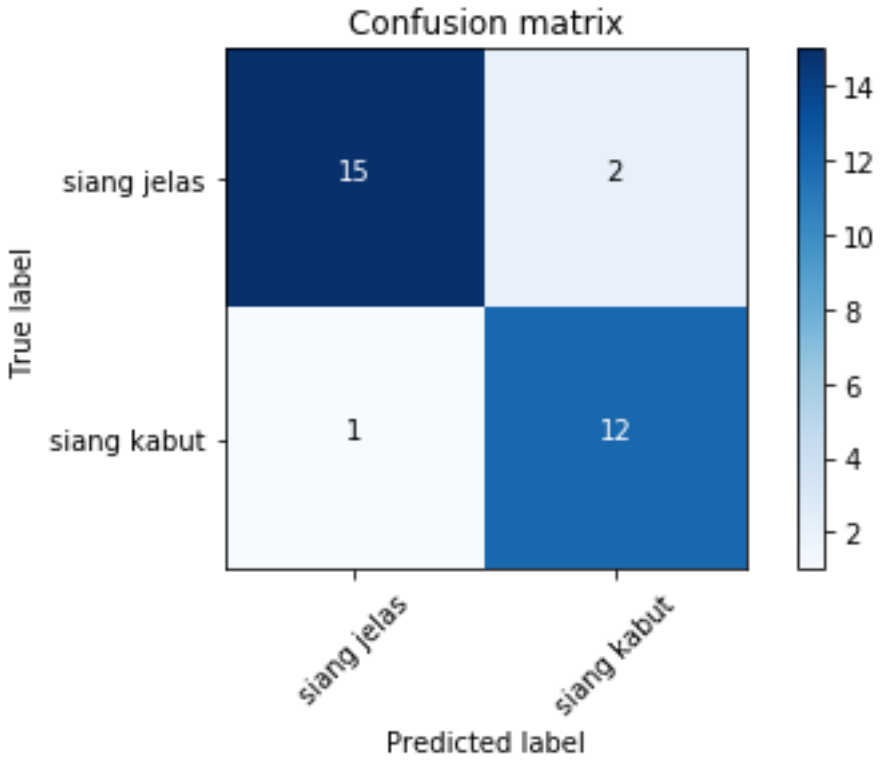
4.1.7 Validasi Model

Setelah mendapatkan model yang terbaik, model yang sudah dibangun sebelumnya perlu di evaluasi terlebih dahulu sehingga kedepannya model ini tetap stabil kinerjanya. Model yang akan divalidasi merupakan model KNN dengan metode Chi-Square. Untuk melakukan validasi model *Machine Learning* pada penelitian ini dilakukan analisis terhadap *Confussion Matrix*, nilai akurasi, *specificity*, dan juga *sensitivity.*

Confusion Matrix

Untuk mendapatkan *Confussion Matrix* dari model tersebut digunakan *library* dari Sklearn. *Confusion Matrix* ini akan diplot sehingga representasi data nya terlihat lebih mudah

dan menarik. Hasil *Confusion Matrix* dari model *Machine Learning* dapat dilihat pada Gambar 4.9.



Gambar 4.9 Hasil *Confusion Matrix*

Berdasarkan Gambar 4.9, dapat dilihat bahwa dari total sebanyak 30 data *test* terdapat sebanyak 15 data diprediksi sebagai *true positive*, 12 data diprediksi sebagai *true negative*, 1 data diprediksi sebagi *false positive*, dan 2 data diprediksi sebagai *false negative*. Dari total keseluruhan prediksi terdapat 3 buah data yang prediksinya tidak sesuai. Hal ini disebabkan oleh nilai atribut yang dimiliki oleh masing-masing data memilik kemiripan dengan kelas yang lain.

Akurasi

Untuk mendapatkan akurasi dari suatu model maka dilakukan perhitungan dengan cara menggunakan rumus akurasi, yang mana secara perhitungan dapat dilihat pada persamaan (4.1).

$$Akurasi = \frac{15 + 12}{30} = 0.90 = 90\% \qquad (4.1)$$

Hasil dari perhitungan yang didapatkan, nilai akurasi dari model tersebut adalah sebesar 90%.

Specificity

Untuk mendapatkan nilai *specificity* dari suatu model maka dilakukan perhitungan dengan cara menggunakan rumus *specificity*, yang mana secara perhitungan dapat dilihat pada persamaan (4.2).

$$Specificity = \frac{12}{12+1} = 0.92 = 92\% \qquad (4.2)$$

Hasil dari perhitungan yang didapatkan, nilai *specificity* dari model tersebut adalah sebesar 92%.

Sensitivity

Untuk mendapatkan nilai *sensitivity* dari suatu model maka dilakukan perhitungan dengan cara menggunakan rumus *sensitivity*, yang mana secara perhitungan dapat dilihat pada persamaan (4.3).

$$Sensitivity = \frac{15}{15+2} = 0.88 = 88\% \qquad (4.3)$$

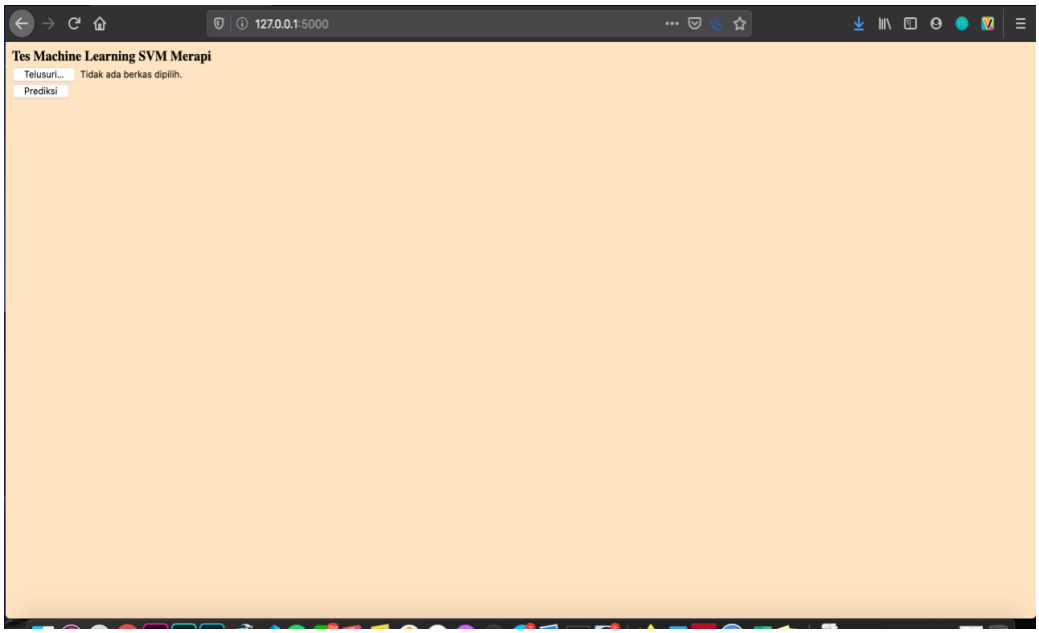
Hasil dari perhitungan yang didapatkan, nilai *sensitivity* dari model tersebut adalah sebesar 88%.

Cross-Validation

Pada penelitian ini digunakan metode Cross-Validation dengan nilai KFold = 5 (jumlah lipatan) dalam melakukan evaluasi terhadap model *Machine Learning*. Setiap *fold* akan menghasilkan berupa nilai rata-rata akurasi. Rata-rata yang dihasilkan oleh tiap *fold* berdasarkan eksekusi kode program pada **Error! Reference source not found.**. Pada *fold* ke-1 menghasilkan akurasi sebesar 85%, *fold* ke-2 menghasilkan akurasi sebesar 70%, *fold* ke-3 menghasilkan akurasi sebesar 90%, *fold* ke-4 menghasilkan akurasi sebesar 80%, dan *fold* ke-5 menghasilkan akurasi sebesar 75%.

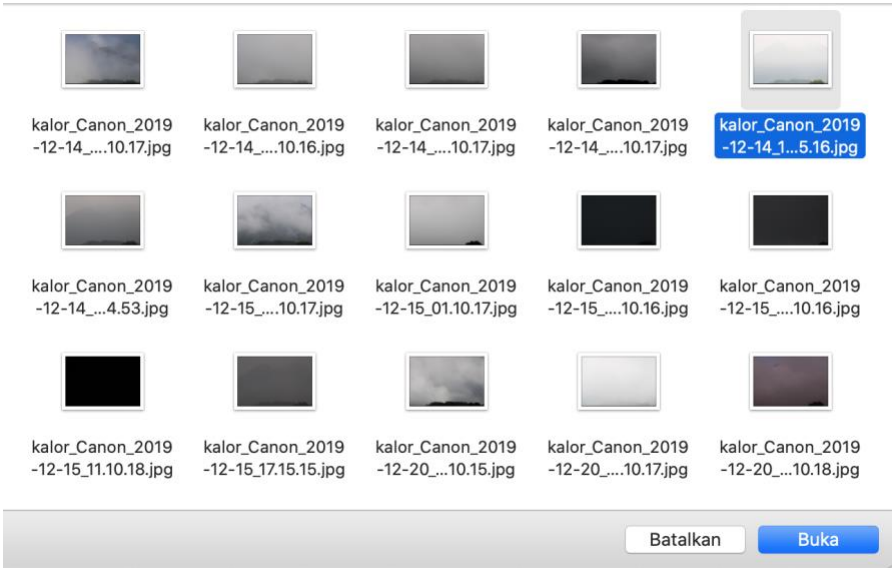
4.2 Pembuktian Model Pada *Website*

Model yang sudah dibangun dan diuji selanjutnya akan diterapkan secara langsung sebagai pembuktian bahwa model ini mampu diaplikasikan pada suatu sistem. Untuk itu pada tahap yang terakhir penulis akan melakukan pengujian model dengan menerapkannya ke dalam sebuah aplikasi *website*. Aplikasi *filtering* gambar ini berbasis *website* dengan menggunakan basis pemrograman Python. Model yang sudah dilatih dan diuji sebelumnya, diambil dan dimasukkan ke dalam website aplikasi *filtering*. Secara tampilan awal *website* terdapat dua tombol yaitu tombol prediksi dan telusuri (pilih gambar) seperti yang dapat dilihat pada Gambar 4.10.



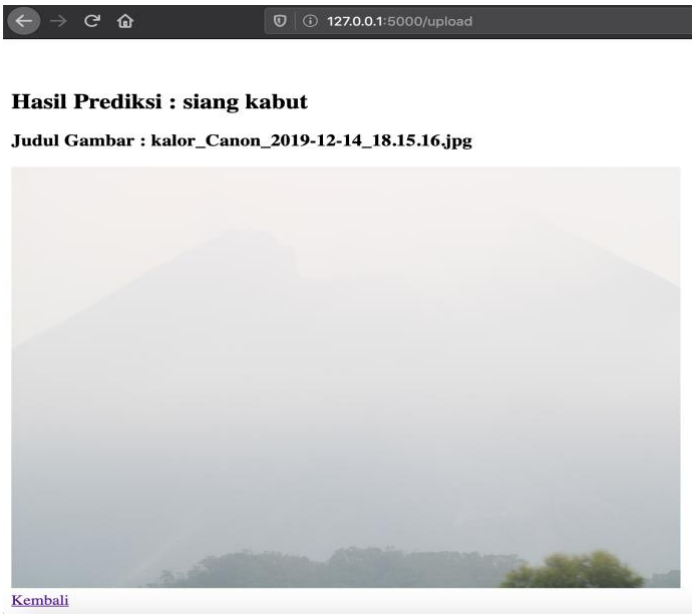
Gambar 4.10 Tampilan awal aplikasi pengujian model

Untuk memulai proses prediksi citra Merapi, pertama tekan tombol Telusuri untuk memilih citra yang ingin diprediksi. Sebuah window baru akan terbuka dan mengarahkan ke *file manager* pada PC seperti pada Gambar 4.11.



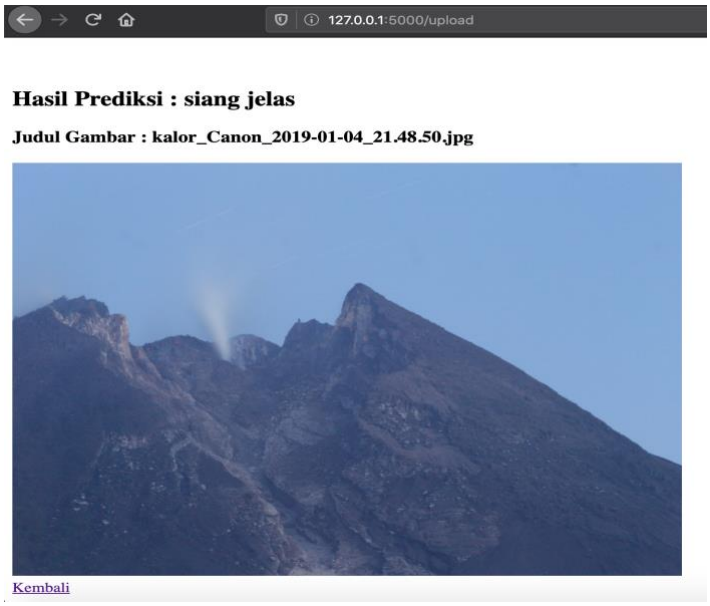
Gambar 4.11 Memilih citra yang akan diprediksi

Pada Gambar 4.11, pilih citra yang akan diprediksi kemudian tekan tombol Buka. Setelah itu tekan tombol Prediksi untuk mengeksekusi prediksi citra. Pada Gambar 4.12, terlihat hasil prediksi dari citra beserta nama file dan juga preview dari citra yang diprediksi. Hasil prediksi mengatakan kalau citra tersebut diklasifikasikan sebagai kelas siang kabut.



Gambar 4.12 Hasil prediksi citra siang kabut

Untuk Contoh citra yang diprediksi sebagai kelas siang jelas dapat dilihat pada Gambar 4.13.



Gambar 4.13 Hasil prediksi citra siang jelas

BAB V
KESIMPULAN & SARAN

5.1 Kesimpulan

Setelah dilakukannya pengembangan model klasifikasi citra Gunung Merapi, berdasarkan tahapan pengembangan hingga pengujian model, dinyatakan bahwa model KNN menggunakan seleksi fitur Chi-square mampu menghasilkan kinerja yang paling baik dengan total akurasi sebesar 90% dan waktu komputasi = 0.001187 detik. Untuk fitur mana saja yang paling berpengaruh menghasilkan model yang andal adalah sebanyak 27 fitur yaitu, 'rMin', 'rAvg', ‘gMax’, 'gMin', 'gAvg', 'bMax', 'bMin', 'bAvg', 'hMin', 'sMax', ‘sMin’, 'sAvg', 'vMax', 'vMin', ‘vAvg’, 'lMin', 'lAvg', ‘laMax’, 'lbMin', ‘lbMax’, ‘yMax’,'yMin', 'yAvg', ‘cbMax’, ‘cbMin’, ‘crMax’, ‘crMin’.

5.2 Saran

Dalam penelitian ini masih terdapat beberapa kekurangan. Adapun saran yang diberikan untuk penelitian selanjutnya pengembangan model dilakukan dengan metode *Reinforcement Learning,* hal ini dikarenakan kedepannya dari waktu ke waktu akan terdapat data-data baru yang bermunculan sehingga model ini harus mampu memperbarui pengetahuannya setiap saat. Pada penelitian ini masih berupa *early research,* yang mana diperlukan penelitian lebih lanjut untuk membahas mekanisme sistem penghapusan citra Gunung Merapi secara otomatis.

DAFTAR PUSTAKA

Hall, M., & Smith, L. a. (1999). Feature Selection for Machine Learning : Comparing a Correlation-based Filter Approach to the Wrapper CFS : Correlation-based Feature. *International FLAIRS Conference*, 5. https://doi.org/10.1.1.50.2192

Hapsari, Y., & Hidayattullah, M. F. (2013). *Deteksi Wajah Dari Berbagai Ras Manusia Menggunakan Warna Kulit Berbasis Ruang Warna L * A * B*. *2013*(November), 409–414.

Hardiyanto, D., & Sartika, D. (2018). EKSTRAKSI FITUR CITRA API BERBASIS EKSTRAKSI WARNA PADA RUANG WARNA HSV dan RGB. *FAHMA*, *16*(3), 1–12. Retrieved from http://stmikelrahma.e-journal.id/FAHMA/article/view/22

Huo, Y., Xin, L., Kang, C., Wang, M., Ma, Q., & Yu, B. (2020). SGL-SVM: A novel method for tumor classification via support vector machine with sparse group Lasso. *Journal of Theoretical Biology*, *486*. https://doi.org/10.1016/j.jtbi.2019.110098

IKHSAN, J., FUJITA, M., & TAKEBAYASHI, H. (2009). *Sustainable sand mining management in Merapi Area using groundsills*.

Karimi, N., Kazem, S., Ahmadian, D., Adibi, H., & Ballestra, L. V. (2020). On a generalized Gaussian radial basis function: Analysis and applications. *Engineering Analysis with Boundary Elements*, *112*(September 2019), 46–57. https://doi.org/10.1016/j.enganabound.2019.11.011

Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., & Jatakia, J. (2017). *Human Skin Detection Using RGB, HSV and YCbCr Color Models*. *137*, 324–332. https://doi.org/10.2991/iccasp-16.2017.51

Leidiyana, H. (2013). Penerapan Algoritma K-Nearest Neighbor Untuk Penentuan Resiko Kredit Kepemilikan Kendaraan Bermotor. *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic*, *1*(1), 65–76.

LING, J., N. KENCANA, I. P. E., & OKA, T. B. (2014). Analisis Sentimen Menggunakan Metode Naïve Bayes Classifier Dengan Seleksi Fitur Chi Square. *E-Jurnal Matematika*, *3*(3), 92. https://doi.org/10.24843/mtk.2014.v03.i03.p070

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Murphy, M. J. (2009). Intrafraction Geometric Uncertainties in Frameless Image-Guided Radiosurgery. *International Journal of Radiation Oncology*Biology*Physics*, *73*(5), 1364–1368. https://doi.org/https://doi.org/10.1016/j.ijrobp.2008.06.1921

PRATOMO, I. (2006). Klasifikasi gunung api aktif Indonesia, studi kasus dari beberapa letusan gunung api dalam sejarah. *Indonesian Journal on Geoscience*, *1*(4), 209–227. https://doi.org/10.17014/ijog.vol1no4.20065

Ruuska, S., Hämäläinen, W., Kajava, S., Mughal, M., Matilainen, P., & Mononen, J. (2018). Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behavioural Processes*, *148*(March 2017), 56–62. https://doi.org/10.1016/j.beproc.2018.01.004

Tkalčič, M., & Tasič, J. F. (2003). Colour spaces - Perceptual, historical and applicational background. *IEEE Region 8 EUROCON 2003: Computer as a Tool - Proceedings*, *A*, 304–308. https://doi.org/10.1109/EURCON.2003.1248032

LAMPIRAN

Kode program konversi ruang warna

```
Gambar_merapi = imread(file)#membaca file citra
HSV_images = cv2.cvtColor(Gambar_merapi, cv2.COLOR_BGR2HSV)#menyimpan citra hasil konversi ruang warna RGB ke HSV
LAB_images = cv2.cvtColor(Gambar_merapi, cv2.COLOR_BGR2LAB)#menyimpan citra hasil konversi ruang warna RGB ke CIELAB
YCrCb_images = cv2.cvtColor(Gambar_merapi, cv2.COLOR_BGR2YCrCb)#menyimpan citra hasil konversi ruang warna RGB ke YCbCr
```

Kode program perhitungan *Channel* warna

```
# RED
 #Simbol ":" berarti all(semua) sehingga nilai keseluruhan dari matriks dihitung
 #Angka 2 mengartikan posisi index dari Channel warna Red (BGR, B=0, G=1, R=2)
 rMax = np.amax(images[:, : , 2]) #mengambil nilai maksimum dari Channel Warna RED
 rMin = np.amin(images[:, :, 2]) #mengambil nilai minimum dari Channel Warna RED
 rAvg = np.mean(images[:, :, 2]) #mengambil nilai mean dari Channel Warna RED

 # Angka 1 mengartikan posisi index dari Channel warna Green
 # GREEN
 gMax = np.amax(images[:, : , 1]) #mengambil nilai minimum dari Channel Warna Green
 gMin = np.amin(images[:, :, 1]) #mengambil nilai maksimum dari Channel Warna Green
 gAvg = np.mean(images[:, :, 1]) #mengambil nilai mean dari Channel Warna Green
 # Angka 0 mengartikan posisi index dari Channel warna Blue
 # BLUE
 bMax = np.amax(images[:, : , 0]) #mengambil nilai minimum dari Channel Warna Blue
 bMin = np.amin(images[:, :, 0]) #mengambil nilai maksimum dari Channel Warna Blue
```

```
 bAvg = np.mean(images[:, :, 0]) #mengambil nilai mean dari Channel Warna
Blue

 # Angka 0 mengartikan posisi index dari Channel warna Hue (HSV, SERVER=0,
S=1, V=2)
 # HUE
 hMax = np.amax(HSV_images[:, :, 0]) #mengambil nilai minimum dari Channel
Warna Hue
 hMin = np.amin(HSV_images[:, :, 0]) #mengambil nilai maksimum dari Channel
Warna Hue
 hAvg = np.mean(HSV_images[:, :, 0]) #mengambil nilai mean dari Channel Warna
Hue

 # Angka 1 mengartikan posisi index dari channel warna Saturation
 # SATURATION
 sMax = np.amax(HSV_images[:, :, 1]) #mengambil nilai minimum dari Channel
Warna Saturation
 sMin = np.amin(HSV_images[:, :, 1]) #mengambil nilai maksimum dari Channel
Warna Saturation
 sAvg = np.mean(HSV_images[:, :, 1]) #mengambil nilai mean dari Channel Warna
Saturation

 # Angka 2 mengartikan posisi index dari channel warna Values
 # VALUES
 vMax = np.amax(HSV_images[:, :, 2]) #mengambil nilai minimum dari Channel
Warna Values
 vMin = np.amin(HSV_images[:, :, 2]) #mengambil nilai maksimum dari Channel
Warna Values
 vAvg = np.mean(HSV_images[:, :, 2]) #mengambil nilai mean dari Channel Warna
Values

 # Angka 0 mengartikan posisi index dari channel warna Lightness(LAB, l=0,
la=1, lb=2)
 # LIGHTNESS
 lMax = np.amax(LAB_images[:, :, 0]) #mengambil nilai minimum dari Channel
Warna Lightness
 lMin = np.amin(LAB_images[:, :, 0]) #mengambil nilai maksimum dari Channel
Warna Lightness
 lAvg = np.mean(LAB_images[:, :, 0]) #mengambil nilai mean dari Channel Warna
Lightness
```

```
 # Angka 1 mengartikan posisi index dari channel warna L'A
 # L'A
 laMax = np.amax(LAB_images[:, :, 1]) #mengambil nilai minimum dari Channel
Warna L'A
 laMin = np.amin(LAB_images[:, :, 1]) #mengambil nilai maksimum dari Channel
Warna L'A
 laAvg = np.mean(LAB_images[:, :, 1]) #mengambil nilai mean dari Channel
Warna L'A

 # Angka 2 mengartikan posisi index dari channel warna L'B
 # L'B
 lbMax = np.amax(LAB_images[:, :, 2]) #mengambil nilai minimum dari Channel
Warna L'B
 lbMin = np.amin(LAB_images[:, :, 2]) #mengambil nilai maksimum dari Channel
Warna L'B
 lbAvg = np.mean(LAB_images[:, :, 2]) #mengambil nilai mean dari Channel
Warna L'B

 # Angka 0 mengartikan posisi index dari channel warna Lumina(YCbCr,
SERVER=0, Cb=1, Cr=2)
 # Lumina
 yMax = np.amax(YCrCb_images[:, :, 0]) #mengambil nilai minimum dari Channel
Warna Lumina
 yMin = np.amin(YCrCb_images[:, :, 0]) #mengambil nilai maksimum dari Channel
Warna Lumina
 yAvg = np.mean(YCrCb_images[:, :, 0]) #mengambil nilai mean dari Channel
Warna Lumina

 # Angka 1 mengartikan posisi index dari channel warna Chroma Blue
 # Cb
 cbMax = np.amax(YCrCb_images[:, :, 1]) #mengambil nilai minimum dari Channel
Warna Chroma Blue
 cbMin = np.amin(YCrCb_images[:, :, 1]) #mengambil nilai maksimum dari
Channel Warna Chroma Blue
 cbAvg = np.mean(YCrCb_images[:, :, 1]) #mengambil nilai mean dari Channel
Warna Chroma Blue

 # Angka 2 mengartikan posisi index dari channel warna Chroma Red
 # Cr
```

```
 crMax = np.amax(YCrCb_images[:, :, 2]) #mengambil nilai minimum dari Channel
Warna Chroma Red
 crMin = np.amin(YCrCb_images[:, :, 2]) #mengambil nilai maksimum dari
Channel Warna Chroma Red
 crAvg = np.mean(YCrCb_images[:, :, 2]) #mengambil nilai mean dari Channel
Warna Chroma Red
```

Kode program mengubah tipe data

```
 images_feature = [] # Array menyimpan keseluruhan data dari semua citra
 # Memasukkan data kedalam array dan konversi tipe data menjadi float
 images_feature.append(np.array([target_name, images_name, rMax, rMin, rAvg,
 gMax, gMin, gAvg, bMax, bMin, bAvg, hMax, hMin, hAvg, sMax, sMin, sAvg,
 vMax, vMin, vAvg, lMax, lMin, lAvg, laMin, laMax, laAvg, lbMin, lbMax,
 lbAvg, yMax, yMin, yAvg, cbMax, cbMin, cbAvg, crMax, crMin,
crAvg,i].astype('float'))) #ubah tipe data
```

Kode program mengubah *array* menjadi *Dataframe*

```
#proses mengubah array ke dataframe dengan menggunakan
 pandas dimana pada kodingan ini digunakan dua parameter masukkan parameter
 pertama data yang ingin diubah menjadi dataframe dan kedua adalah pemberian
 nama kolom pada dataframe

df = pd.DataFrame(np.array(images_feature), columns = ['target_name',
 'rMax', 'rMin', 'rAvg', 'gMax', 'gMin', 'gAvg', 'bMax', 'bMin', 'bAvg',
 'hMax', 'hMin', 'hAvg', 'sMax', 'sMin', 'sAvg', 'vMax', 'vMin', 'vAvg',
 'lMax', 'lMin', 'lAvg', 'laMin', 'laMax', 'laAvg', 'lbMin', 'lbMax',
 'lbAvg', 'yMax', 'yMin', 'yAvg', 'cbMax', 'cbMin', 'cbAvg', 'crMax',
 'crMin', 'crAvg'])
```