MACHINE LEARNING PENGENALAN CITRA DIGITAL BERBASIS *SOFTWARE AS A SERVICE* (SaaS)

THE MACHINE LEARNING OF DIGITAL IMAGE RECOGNITION BASE ON SOFTWARE AS A SERVICE (SAAS)

ANDI LUKMAN

P2700211434

KONSENTRASI TEKNIK INFORMATIKA

PROGRAM STUDI S2 TEKNIK ELEKTRO

UNIVERSITAS HASANUDDIN

MAKASSAR

2013

***MACHINE LEARNING* PENGENALAN CITRA DIGITAL**

BERBASIS *SOFTWARE AS A SERVICE* (SaaS)

Tesis

Sebagai salah satu syarat untuk mencapai gelar Magister

Program Studi

Teknik Elektro

Disusun dan diajukan oleh

ANDI LUKMAN

P2700211434

Kepada

PROGRAM STUDI S2 TEKNIK ELEKTRO

UNIVERSITAS HASANUDDIN

MAKASSAR

2013

TESIS

MACHINE LEARNING PENGENALAN CITRA DIGITAL BERBASIS *SOFTWARE AS A SERVICE* (SaaS)

Disusun dan diajukan oleh

ANDI LUKMAN

Nomor Pokok P2700211434

Telah dipertahankan di depan Panitia Ujian Tesis

Pada Tanggal 28 Juni 2013

Dan dinyatakan telah memenuhi syarat

Menyetujui

Komisi Penasehat,

Dr.Eng. Syafaruddin, S.T, M.Eng	**Merna Baharuddin, S.T, M.Eng, Ph.D**
Ketua	**Sekretaris**
Ketua Program Studi Teknik Elektro,	Direktur Program Pascasarjana Universitas Hasanuddin,
Prof. Dr. Ir. Salama Manjang, MT	**Prof. Dr. Ir. Mursalim**

PERNYATAAN KEASLIAN TESIS

Yang betanda-tangan di bawah ini :

Nama	: Andi Lukman
NIM	: P2700211434
Program Studi	: Teknik Elektro
Konsentrasi	: Teknik Informatika

Menyatakan dengan sebenarnya bahwa tesis yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil-alihan tulisan atau pemikiran orang lain. Adapun kutipan atau rujukan sebagai sumber informasi yang saya gunakan dari penulis lain, telah saya sebutkan namanya pada daftar pustaka tesis ini.

Apabila dikemudian hari ada terbukti bahwa tesis ini adalah hasil karya orang lain maka saya bersedia menerima sanksi apapun sesuai peraturan yang berlaku.

Makassar, 28 Juni 2013

Penulis

(Andi Lukman)

PRAKATA

Puji syukur kehadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tesis berjudul : "*Machine Learning* Pengenalan Citra Digital Berbasis *Software as a Service* (SaaS)". Penyusunan tesis ini merupakan salah satu syarat memperoleh gelar Magister Teknik pada program studi Teknik Elektro, konsentrasi Teknik Informatika Pascasarjana Universitas Hasanuddin.

Dalam penyusunan tesis ini, berbagai pihak telah memberikan bantuan baik materil maupun moril, dorongan, saran dan kritik yang membangun agar menjadi lebih baik sehingga dalam kesempatan ini penulis menyampaikan terima kasih kepada :

1. Ayah, H. Syamsuddin Nur dan Ibu, Hj. Andi Syahraeni A.P. Meskipun telah meninggal dunia, namun cinta, nasehat, semangat perjuangan dan pengorbanan mereka masih tetap memotivasi penulis untuk terus berkarya.
2. Istriku dan Anak-anakku, Marwana, S.Kom, M.T, Arkani Al Mahdi, Dalilana Az Zahra dan Vidya Magitra. Kasih dan kebahagiaan mereka membuat penulis tetap semangat menyelesaikan tesis ini.
3. Bapak Dr.Eng. Syafaruddin, S.T, M.Eng dan Ibu Merna Baharuddin, S.T, M.Eng, Ph.D selaku pembimbing yang telah memberikan pengetahuan dan bimbingan yang sangat bermanfaat.

4. Ketua Program Studi Teknik Elektro Pascasarjana, Bapak Prof. DR. Ir. H. Salama Manjang, MT. beserta staff.
5. Direktur Program Pascasarjana Universitas Hasanuddin, Bapak Prof. DR. Ir. Mursalim beserta staff.
6. Bapak-Bapak Dosen Teknik Informatika yang telah memberikan ilmu dan pengalaman selama kuliah di Pascasarjana UNHAS.
7. Ketua Sekolah Tinggi Multi Media dan Informatika (STIMED) Nusa Palapa Makassar beserta staff.
8. Teman-teman PascaMelekTI'11. Diskusi, *sharing*, canda tawa dan semangat perjuangan mereka mampu menulari penulis dan menjadi sebuah dukungan yang luar biasa.
9. Serta semua pihak yang tidak dapat disebutkan satu persatu atas segala dukungan, bantuan dan saran sehingga tesis ini dapat terselesaikan dengan baik.

Penulis menyadari bahwa karya ilmiah ini masih mempunyai kekurangan, sehingga kritik dan saran yang membangun sangat diharapkan. Mudah-mudahan kekurangan-kekurangan tersebut dapat disempurnakan pada penelitian-penelitan penulis selanjutnya. Semoga tesis ini dapat bermanfaat bagi kita semua. Tetap berkarya demi nusa dan bangsa, Jayalah Indonesiaku, Merdeka!

Makassar, 28 Juni 2013

Penulis

ABSTRAK

ANDI LUKMAN. *Machine Learning* Pengenalan Citra Digital Berbasis *Software as A Service* (SaaS) (dibimbing oleh **Dr.Eng. Syafaruddin, ST., M.Eng** dan **Merna Baharuddin, ST., M.Tel.Eng, Ph.D**)

Aplikasi Pengenalan citra pada umumnya dirancang untuk kebutuhan khusus objek penelitian tertentu dan tidak dapat digunakan secara *multiuser*. Penelitian ini bertujuan membangun aplikasi *Machine Learning* pengenalan citra digital sebagai alat bantu bagi pengguna untuk mendapatkan algoritma klasifikasi terbaik dalam mengenali objek citra digital. Aplikasi dapat digunakan secara langsung di *cloud* tanpa proses instalasi dan bersifat *multiuser* menggunakan teknologi *Software as a Service* (SaaS). Algoritma yang digunakan dalam penelitian ini adalah algoritma klasifikasi *Machine Learning* WEKA yaitu :*Support Vector Machine*, *K-Nearest Neighbor, Naïve Bayes, C4.5 Decision Tree, Logistic Regression* dan *Random Forest*. Data set yang digunakan diambil dari *California Institute of Technology* bernama *Caltech* 101. Teknik dokumentasi menggunakan diagram *Unified Modeling Language*. Aplikasi dibangun menggunakan bahasa pemrograman java, untuk sisi *client* menggunakan *Google Web Toolkit* dan sisi *server* menggunakan *Java Servlet*. Aplikasi di-*deploy* ke *Google App Engine*. 4 bagian utama aplikasi yaitu Login, *Overview*, Pra-proses dan latih_uji_pengenalan. Pengguna dapat menggunakan aplikasi hanya bermodalkan *web browser* secara *online* tanpa perlu melakukan instalasi aplikasi. Aplikasi sukses dalam pengujian *blackbox* dan pengujian keberhasilan algoritma-algoritma *machine learning* dalam mengenali citra. Algoritma *Logistic Regression* mempunyai tingkat keberhasilan tertinggi yaitu 95%, 90% untuk kelas *yes* dan 100% untuk kelas *no* dalam mengenali wajah.

Kata Kunci : *Machine Learning*, Pengenalan, Citra digital, *Software As a Service*, Algoritma Klasifikasi *WEKA*, *Google App Engine*.

ABSTRACT

ANDI LUKMAN. Machine Learning of Digital Image Recognition Based on Software as a Service (SaaS) (Supervised by **Dr.Eng. Syafaruddin, ST., M.Eng** and **Merna Baharuddin, ST., M.Tel.Eng, Ph.D**)

Image recognition applications are usually designed for the special needs of a particular research object and can not be used in multiuser. This research aims to establish the application of digital image recognition machine learning as a tool for users to get the best classification algorithm to recognize objects in digital images. Application can be used directly in the cloud without the installation process and multiuser using technology Software as a Service (SaaS). The algorithm used in this research is WEKA Machine Learning classification algorithms, namely: Support Vector Machine, K-Nearest Neighbor, Naïve Bayes, C4.5 Decision Tree, Logistic Regression and Random Forest. The data set used is taken from the California Institute of Technology called Caltech 101. Documentation techniques using Unified Modeling Language diagrams. Applications built using the Java programming language, client side using Google Web Toolkit and server side using Java Servlet. Application is deployed to Google App Engine. 4 main parts, namely application login, Overview, Pre-processing and train_test_recognize. Users can use the application with web browser without the need to install the application. Applications Successfully testing with Blackbox and the testing for success of machine learning algorithms to recognize the image. Logistic Regression algorithm has the highest success rate of 95%, 90% for class yes and 100% for no classes in recognizing faces.

Keywords: Machine Learning, Recognizing, Digital Image, Software As a Service, WEKA Classification Algorithms, Google App Engine.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

A. Latar Belakang

Pengenalan citra digital merupakan salah satu tema penelitian *computer vision* khususnya *pattern recognition* dan *machine learning*. Metode *machine learning* digunakan agar mempermudah komputer untuk belajar mengenali pola citra digital dengan memanfaatkan teori-teori matematika dan statistik. Penelitian pengenalan citra sangat bermanfaat untuk kebutuhan klasifikasi objek di berbagai bidang, seperti: kedokteran, kelautan dan bisnis.

Software as a Service (SaaS) merupakan salah satu layanan Teknologi *Cloud Computing* yang memudahkan pengguna menggunakan aplikasi secara *online* dimanapun dan kapanpun (selama ada koneksi internet) tanpa harus melakukan instalasi (Allen, dkk., 2012). Aplikasi berbasis SaaS juga dapat digunakan oleh beberapa pengguna dalam waktu bersamaan (*multiuser*). Teknologi ini dapat membantu para pengguna agar lebih fleksibel dalam mengakses aplikasi.

Telah banyak penelitian sebelumnya yang mengangkat tema pengenalan citra digital, seperti pengenalan citra buku (Lukman, 2012), citra mamografi untuk mengenali kanker payudara (Hajare, dkk., 2012) dan citra kelenjar getah bening metastatis untuk mengenali kanker lambung (Li, dkk., 2012). Pada umumnya penelitian-penelitian

sebelumnya membuat aplikasi khusus untuk mengenal citra digital tertentu menggunakan algoritma *machine learning* tertentu dengan nilai-nilai parameter optimal sesuai hasil penelitannya. Selain itu, ada juga aplikasi umum ujicoba menggunakan algoritma-algoritma *machine learning*, misalnya WEKA, dimana aplikasi ini memberikan fleksibilitas dalam pemilihan algoritma dan nilai parameter sesuai keinginan peneliti (Desai, dkk., 2012).

Aplikasi-aplikasi khusus pengenalan citra digital hanya efektif digunakan untuk citra yang telah diteliti sebagai objek penelitiannya, sehingga setiap peneliti citra harus membangun aplikasi baru untuk objek penelitian baru. Hal ini membutuhkan keahlian dalam pemrograman, sementara para peneliti citra berasal dari berbagai bidang ilmu yang bukan *programmer*. Aplikasi tersebut juga tidak dapat digunakan secara *multiuser* dan harus melalui proses instalasi sebelum digunakan. Aplikasi *machine learning* popular seperti WEKA hanya diperuntukkan untuk analisis algoritma *machine learning* secara umum, sementara aplikasi pengenalan citra digital yang memberikan fleksibilitas dalam pemilihan algoritma belum tersedia.

Dari latar belakang masalah tersebut, penulis mencoba mengabungkan ide aplikasi khusus pengenalan citra digital dan *machine learning* WEKA serta melengkapinya dengan pemanfaatan teknologi SaaS. Penelitian ini mencoba membangun sebuah aplikasi pengenalan citra digital menggunakan algoritma *machine learning* untuk kebutuhan

klasifikasi yang berbasis *Software as a Service*. Hasil penelitian ini diharapkan dapat membantu para peneliti pengenalan citra tanpa harus membangun sendiri aplikasi setiap melakukan penelitian, dilengkapi dengan kemudahan akses dan fleksibilitas pemilihan metode / algoritma.

B. Rumusan Masalah

Deskripsi masalah yang akan dikemukakan dalam penelitian ini adalah sebagai berikut:

1. Bagaimana membangun aplikasi pengenalan citra digital yang fleksibel untuk kebutuhan klasifikasi, sehingga dapat digunakan untuk berbagai objek citra sesuai kebutuhan peneliti citra.
2. Bagaimana memanfaatkan teknologi *Software as a Service* agar aplikasi yang dibangun dapat digunakan oleh para peneliti citra secara *multiuser* dan tanpa proses instalasi aplikasi.

C. Tujuan Penelitian

Penelitian ini bertujuan membangun *machine learning* pengenalan citra digital untuk kebutuhan klasifikasi menggunakan algoritma *machine learning* dengan memanfaatkan teknologi *Software as a Service* (SaaS) agar dapat digunakan sebagai *tools* untuk mempermudah penelitian pengenalan citra.

D. Batasan Masalah

Permasalahan yang dibahas pada penelitian ini dibatasi pada :

1. Input Citra digital berupa *image-based* (menggunakan informasi mentah dari *pixel* citra digital) bertipe *file* PNG8 (8 bit) berwarna (*true color*).
2. Target Klasifikasi terdiri dari dua kelas, apakah citra digital sesuai (ya) atau tidak sesuai (tidak).
3. Algoritma *machine learning* yang digunakan dalam penelitan ini adalah algoritma klasifikasi *Machine Learning* WEKA yaitu: *Support Vector Machine*, *K-Nearest Neighbor*, *Naive Bayes*, *C4.5 Decision Tree*, *Logistic Regression* dan *Random Forest*.
4. Aplikasi dibangun menggunakan bahasa pemrograman *Java 2 Enterprise Edition* (J2EE) dan di-*deploy* ke *Platform as a Service* (PaaS) *Google App Engine* (GAE).

E. Manfaat Penelitian

Diharapkan dari penelitian akan diperoleh manfaat :

1. Peneliti pengenalan citra dapat langsung menggunakan aplikasi pengenalan citra digital yang dibangun untuk kebutuhan klasifikasi tanpa harus membangun sendiri aplikasi sesuai objek penelitan masing-masing.

2. Membantu pengguna untuk bereksperimen dalam menentukan algoritma dan hasil *learning* yang optimal sesuai citra digital yang diteliti kemudian menggunakannya untuk mengenali citra digital.
3. Memudahkan pengguna dalam menggunakan aplikasi secara *online* dimanapun dan kapanpun tanpa harus melakukan proses instalasi serta dapat digunakan secara *multiuser*.

BAB II

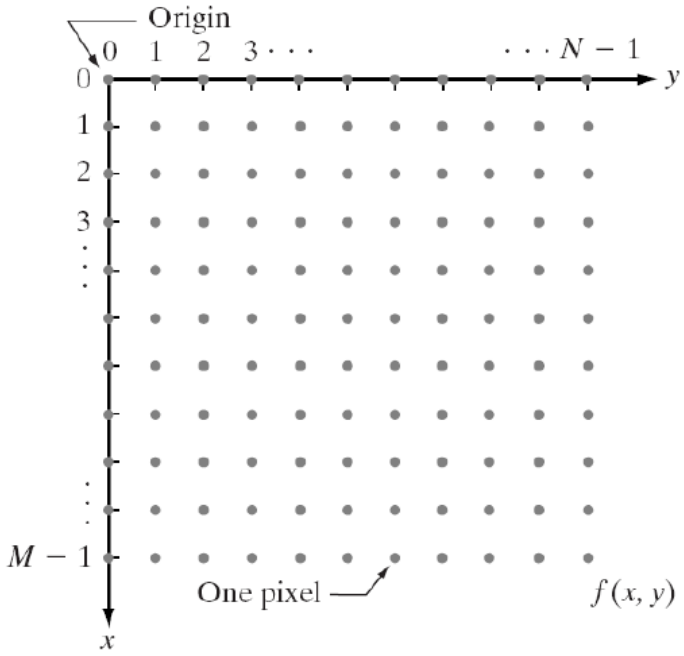
TINJAUAN PUSTAKA

Pada bab ini akan dibahas beberapa tinjauan pustaka dari berbagai sumber baik dari jurnal ilmiah penelitian-penelitan sebelumnya maupun dari buku sebagai landasan-landasan teori penunjang penelitian. Sesuai topik penelitian, tinjauan pustaka yang akan dibahas yaitu : citra digital, pengolahan citra, *machine learning*, *machine learning* WEKA, *support vector machine*, *k-nearest neighbor*, *naïve bayes*, *c4.5 decision tree, logistic regression, random forest, software as a service*, google app engine, *roadmap* penelitian dan kerangka pikir.

A. Citra Digital

Citra digital merupakan gambar pada bidang dua dimensi yang direkam oleh mesin digital, misalnya kamera digital dan *scanner*. Di dalam komputer, citra digital disimpan sebagai suatu file dengan format tertentu. Format citra tersebut menunjukan cara sebuah citra digital disimpan. Salah satu format format yang dapat digunakan adalah format PNG (*Portable Network Graphics*). PNG menggunakan metode pemadatan yang tidak menghilangkan bagian dari citra (berbeda dengan format JPG yang bersifat *lossy*) dan mempunyai faktor kompresi yang lebih baik dibandingkan dengan format GIF. Secara umum PNG dipakai untuk Citra Web yaitu PNG-8 (8 bit) dan PNG-24 (24 bit).

Ukuran citra digital dinyatakan dalam pixel (*picture element*) berupa matriks dua dimensi *M* X *N* yang membentuk sebuah fungsi *f(x,y)*. Ilustrasi tentang matriks citra digital dapat dilihat pada gambar 2.1. *M* mewakili banyaknya *pixel* perbaris, sementara *N* adalah banyaknya *pixel* perkolom.



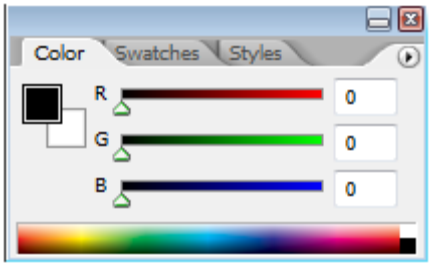
Gambar 2.1. Ilustrasi matriks citra digital

Berdasarkan jenis warnanya, citra digital dapat dibagi menjadi tiga yaitu:

1. **Citra warna (*true color*)**

   Setiap *pixel* pada citra warna mewakili warna kombinasi tiga warna dasar, yaitu merah, hijau, dan biru atau biasa disingkat dengan RGB (*Red, Green, Blue*) (Santi, 2011). Dengan kata lain satu *pixel* terdiri dari tiga *layer*: *R-layer*, *Green-layer* dan *Blue-Layer*. Setiap warna

dasar menggunakan penyimpanan 8 bit = 1 byte, jadi satu *pixel* pada citra warna diwakili oleh 3 bytes. Ilustrasi citra warna dapat dilihat pada gambar 2.2.



Gambar 2.2. Ilustrasi jenis citra warna terdiri dari R, G dan B

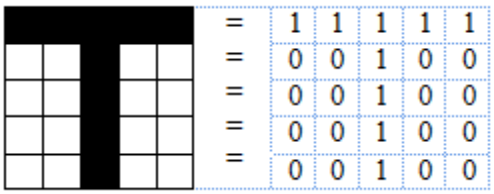
2. Citra *grayscale* (tingkat keabu-abuan)

Citra *grayscale* merupakan jenis citra yang mempunyai tingkat keabu-abuan. Tingkatan keabu-abuan di sini merupakan warna abu-abu dengan berbagai tingkatan dari hitam hingga putih. Satu *pixel* pada citra jenis ini hanya terdiri dari satu *layer* yang berisi intensitas cahaya dari 0 sampai dengan 255 (Santi, 2011). Ilustrasi jenis citra *grayscale* dapat dilihat pada gambar 2.3.

Gambar 2.3. Ilustrasi jenis citra *grayscale*

3. Citra Biner (*Monocrhome*)

Citra biner atau biasa juga disebut *B/W* memiliki 2 warna, yaitu hitam dan putih. Warna hitam bernilai 1 dan warna putih bernilai 0 (Lukman, 2012). Untuk menyimpan kedua warna ini dibutuhkan 1 bit di memori. Ilustrasi jenis citra ini dapat dilihat pada gambar 2.4.



Gambar 2.4. Ilustrasi jenis citra biner atau *B/W*

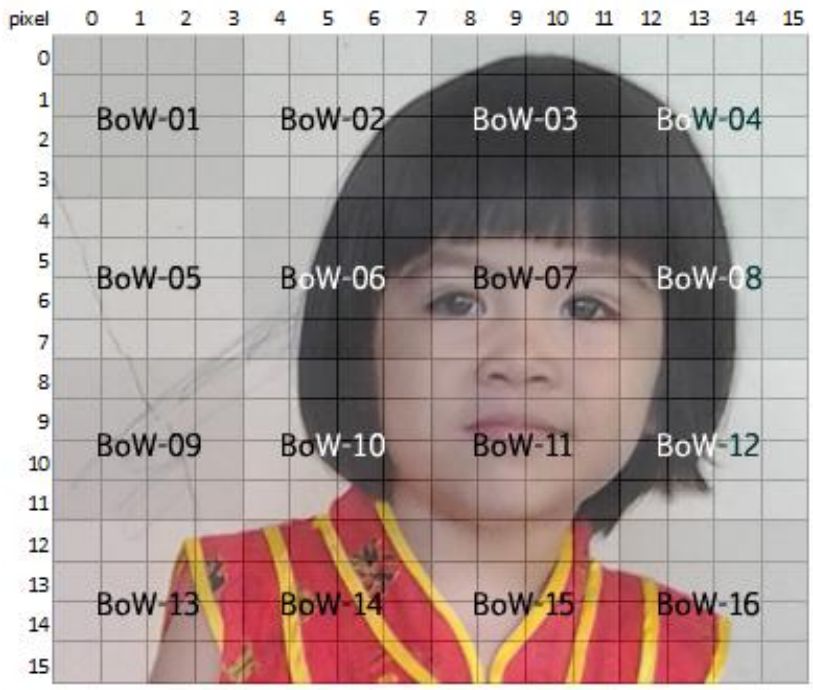
Citra digital merupakan data yang dapat diekplorasi sesuai kebutuhan. Untuk melakukan hal tersebut gunakan teknik-teknik pengolahan citra.

B. Pengolahan Citra

Pengolahan citra merupakan proses memanipulasi dan menganalisis citra dengan bantuan komputer, dalam hal ini mengolah informasi yang terdapat pada suatu citra digital untuk keperluan pengenalan objek secara otomatis. Ada berbagai teknik pengolahan citra tergantung kebutuhan dan keluaran yang diinginkan. Berikut beberapa teknik yang dapat digunakan yaitu: normalisasi, *Bag of Words* dan *grayscaling* dan.

Normalisasi merupakan proses menyeragamkan ukuran citra digital inputan menjadi matriks ukuran *M* X *N*. Hal ini dilakukan karena setiap citra yang diolah belum tentu mempunyai ukuran yang sama. Normalisasi juga digunakan untuk memperkecil citra digital agar jumlah pixel yang akan diolah tidak terlalu banyak. Semakin banyak jumlah pixel, semakin banyak *data set* yang menyebabkan semakin lama waktu komputasi.

Bag of Words (BoW) atau biasa juga disebut *bag of feature* merupakan teknik yang diadopsi dari *text mining* untuk kebutuhan memperkecil data set dengan mempertahankan posisi setiap bagiannya. Sebagai ilustrasi, misalnya gambar 5.2 merupakan citra berukuran 16X16 pixel dibagi menjadi 16 BoW. Setiap bagian terdiri dari 16 pixel (4X4 pixel).



Gambar 2.5. Ilustrasi *Bag of Words* 16 bagian

Grayscaling adalah teknik yang digunakan untuk mengubah citra berwarna menjadi bentuk *grayscale*. Setiap pixel pada citra berwarna terbagi atas tiga bagian yaitu R (*red*), G (*green*) dan B (*blue*). Aturan yang digunakan sebagai berikut (Lukman, 2012):

$$(x,y,f(x,y)) \rightarrow (x,y,I(x,y))$$

$$I(i,j) = \frac{R(i,j)+G(i,j)+B(i,j)}{3} \quad \text{.....(1)}$$

Keterangan :

$f(x,y)$ = nilai intensitas lama

$I(x,y)$ = nilai intensitas *pixel grayscale*

Setelah melalui pengolahan citra, citra digital siap digunakan sebagai data set yang dapat diolah oleh *machine learning* untuk kebutuhan pelatihan, pengujian dan pengenalan..

C. Machine Learning

Pada umumnya, Kecerdasan manusia didapatkan dari proses belajar dari pengalaman. Semakin banyak pengalaman manusia dalam memecahkan masalah maka semakin bijak dan semakin tepat dalam pengambilan keputusan. Kemampuan tersebut menginsiprasi dunia komputer sehingga melahirkan bidang keilmuan *Machine learning* yang mempelajari bagaimana sebuah mesin atau komputer dapat belajar dari pengalaman atau bagaimana cara memprogram mesin untuk dapat belajar. *Machine learning* membutuhkan data untuk belajar sehingga biasa juga diistilahkan dengan *learn from data* (Alpaydin, 2010).

Metode belajar *machine learning* menggunakan algoritma-algoritma *learning*. Secara garis besar ada 3 jenis metode belajar yang digunakan yaitu:

1. ***Supervised Learning***

   Supervised learning merupakan metode belajar terawasi. Program diberikan beberapa contoh data yang telah diketahui jenis/ klasifikasinya sebagai bahan pembelajaran atau pelatihan (Harrington, 2012). Misalnya sebuah program diberikan benda berupa bangku dan meja, maka setelah beberapa contoh, program tersebut harus dapat memilah-milah objek ke dalam klasifikasi yang cocok.

   Terdapat kemungkinan program akan salah dalam mengklasifikasi sebuah objek setelah dilatih. Oleh karena itu, selain menggunakan *training set,* juga memberikan *test set*. Dari situ akan terukur persentase keberhasilannya.

2. ***Unsupervised Learning***

   Unsupervised Learning merupakan metode belajar tidak terawasi. Metode ini menggunakan prosedur yang berusaha untuk mencari partisi dari sebuah pola. *Unsupervised learning* mempelajari bagaimana sebuah sistem dapat belajar untuk merepresentasikan pola input dalam cara yang menggambarkan struktur statistikal dari keseluruhan pola input (Harrington, 2012). Berbeda dari *supervised learning*, metode ini tidak memiliki target output yang eksplisit sehingga biasa digunakan untuk kebutuhan pengelompokan (*clustering*).

3. Reinforcement Learning

Reinforcement learning adalah sebuah metode *learning* yang mempelajari aturan kontrol dengan cara berinteraksi dengan lingkungan yang masih asing (Alpaydin, 2010). Program akan mendapatkan hukuman jika salah dalam pengambilan keputusan dan hadiah jika benar. Pengalaman interaksi tersebut terakumulasi sehingga program dapat mengambil kesimpulan dikemudian hari menggunakan pola-pola yang telah dipelajarinya.

Terdapat beberapa aplikasi *machine learning* yang telah dikembangkan oleh universitas-universitas ternama di dunia. Salah satu aplikasi *machine learning* yang popular adalah *machine learning* WEKA.

D. Machine Learning WEKA

WEKA (*Waikato Environment for Knowledge Analysis)* merupakan perangkat lunak *Data Mining* yang memiliki sekumpulan algoritma standar *Machine Learning* untuk kebutuhan praproses, klasifikasi, pengelompokan, regresi, *Association Rules Mining* (ARM) dan visualisasi (Desai dkk., 2012). Weka dikembangkan oleh Univeritas Waikato di Selandia Baru menggunakan bahasa pemrograman Java dan merupakan perangkat lunak gratis dibawah lisensi *GNU General Public*. Perangkat Lunak ini dapat dijalankan berbasis *GUI* dan *command line*. Selain itu, WEKA juga menyediakan *Library* yang dapat langsung digunakan dalam pemrograman Java (Witten dkk., 2011).

Setiap data yang akan diolah menggunakan *Machine Learning* WEKA, harus memenuhi standar *dataset* dari kelas *weka.core.Instance*. Setiap instan memiliki beberapa atribut. Domain dari atribut dapat berupa Nominal (Misalnya : ya dan tidak), Numerik (bilangan bulat dan pecahan), String, Date (tanggal) dan Relasional. Dataset tersebut biasanya disimpan dalam format file ARFF (*Attribute-Relation File Format*) yang terdiri dari dua bagian yaitu header (menjelaskan tipe atribut) dan bagian data (meliputi data yang dipisah dengan koma). Gambar 2.6 memperlihatkan contoh dataset *weather* (cuaca) dalam format ARFF (Witten dkk., 2011).

```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

Gambar 2.6. Contoh File ARFF untuk data cuaca

Berikut cara menggunakan algoritma klasifikasi *machine learning* WEKA pada pemrograman java:

1. ***Proses Instansiasi file ARFF***

   Data set dalam format ARFF melalui proses instansiasi atau dijadikan instan sebelum dilakukan klasifikasi. Gambar 2.7 merupakan contoh kode program java dalam membaca file ARFF kemudian diinstansiasi.

```
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
...
BufferedReader reader = new BufferedReader(
                            new FileReader("/some/where/data.arff"));
Instances data = new Instances(reader);
reader.close();
// setting class attribute
data.setClassIndex(data.numAttributes() - 1);
```

Gambar 2.7. Contoh kode program proses instansiasi file ARFF

   Index *class* merupakan atribut target yang digunakan untuk klasifikasi. Secara *default*, dalam file ARFF berada pada atribut terakhir, makanya menggunakan perintah *numAttributes-1*. Kita harus set instansiasi jika ingin melakukan klasifikasi, misalnya dengan cara *weka.classifiers.Classifier.buildClassifier(data)*.

2. ***Membangun Klasifikasi***

   Algoritma klasifikasi WEKA cukup mudah digunakan untuk pelatihan data set yang diberikan. sebagai contoh , kita dapat melatih data set

menggunakan algoritma *C4.5 Decision Tree* menggunakan metode *buildClassifier(Instances)*. Hal ini dapat dilihat pada Gambar 2.8.

```
import weka.classifiers.trees.J48;
...
String[] options = new String[1];
J48 tree = new J48();         // new instance of tree
tree.buildClassifier(data);   // build classifier
```

Gambar 2.8 contoh kode program membangun klasifikasi WEKA

3. *Pelatihan dan Pengujian*

```
import weka.core.Instances;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.J48;
...
Instances train = ...   // from somewhere
Instances test = ...    // from somewhere
// train classifier
Classifier cls = new J48();
cls.buildClassifier(train);
// evaluate classifier and print some statistics
Evaluation eval = new Evaluation(train);
eval.evaluateModel(cls, test);
System.out.println(eval.toSummaryString("\nResults\n======\n", false));
```

Gambar 2.9. Contoh kode program pelatihan dan pengujian algoritma

Kita dapat melatih dan kemudian menguji algoritma *machine learning* sesuai data set yang diberikan menggunakan WEKA pada kode program. Sebagai contoh dapat dilihat pada Gambar 2.9, J48 yang merupakan algoritma C4.5 Decision Tree pada WEKA diinstansiasi, dilatih, diuji dan kemudian hasil pengujian ditampilkan.

4. *Mengklasifikasikan Instan*

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import weka.core.Instances;
...
// load unlabeled data
Instances unlabeled = new Instances(
                      new BufferedReader(
                        new FileReader("/some/where/unlabeled.arff")));

// set class attribute
unlabeled.setClassIndex(unlabeled.numAttributes() - 1);

// create copy
Instances labeled = new Instances(unlabeled);

// label instances
for (int i = 0; i < unlabeled.numInstances(); i++) {
  double clsLabel = tree.classifyInstance(unlabeled.instance(i));
  labeled.instance(i).setClassValue(clsLabel);
}
// save labeled data
BufferedWriter writer = new BufferedWriter(
                        new FileWriter("/some/where/labeled.arff"));
writer.write(labeled.toString());
writer.newLine();
writer.flush();
writer.close();
```

Gambar 2.10. Contoh Klasifikasi instan tanpa target klasifikasi

Selain proses pelatihan dan pengujian, kita dapat menggunakan library algoritma klasifikasi WEKA untuk mengklasifikasi atau mengenali data set yang belum mempunyai target klasifikasi (*unlabled dataset*) atau belum dikenali. Gambar 2.10 memperlihatkan contoh kode program mengambil file */some/where/unlabeled.arff*, kemudian

menggunakan model pelatihan dan menyimpannya kembali sebagai data set yang telah mempunyai target klasifikasi (*labeled dataset)* ke */some/where/labeled.arff*.

Library Algoritma klasifikasi pada WEKA yang dapat digunakan dalam kode pemrograman Java terdapat pada kelas *weka.classifiers.* Beberapa algoritma *machine learning* yang terdapat pada kelas tersebut antara lain : *Support Vector Machine*, *K-Nearest Neighbor*, *Naive Bayes*, *C4.5 Decision Tree*, *Logistic Regression* dan *Random Forest*. Yang akan dibahas pada sub-sub bab berikutnya.

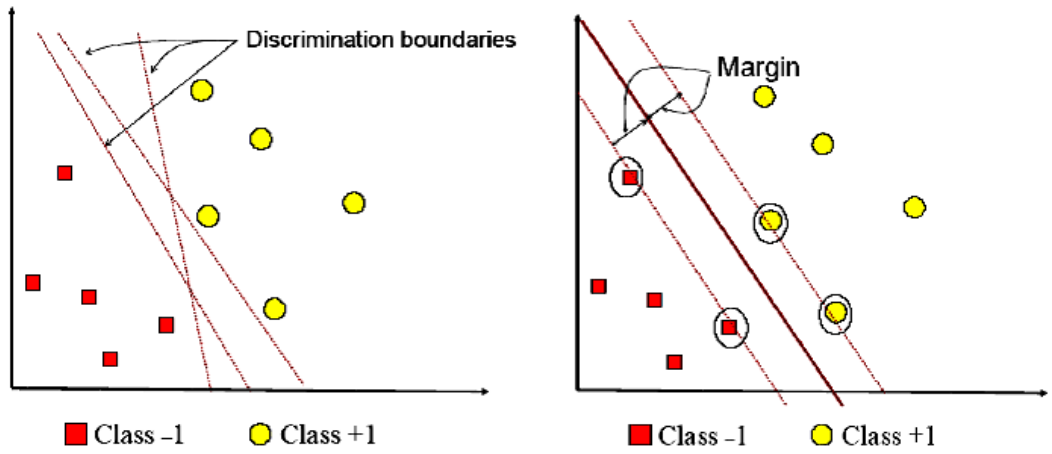
E. Support Vector Machine

*Support Vector Machine (*SVM*)* merupakan algoritma dengan metode *supervised learning* yang digunakan sebagai alat bantu pengenalan pola (*pattern recognition*) di berbagai bidang seperti bioinformatika, diagnose kanker, klasifikasi citra, dan sebagainya (Hajare dkk., 2012). Konsep SVM secara sederhana merupakan usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah class pada ruang input. *Hyperplane* dalam ruang vektor berdimensi d adalah *affine subspace* berdimensi d-1 yang membagi ruang vektor tersebut ke dalam dua bagian, yang masing-masing berkorespondensi pada kelas yang berbeda.

Gambar 2.11 memperlihatkan beberapa pola yang merupakan anggota dari dua buah kelas : +1 dan –1. Pola yang tergabung pada class

–1 disimbolkan dengan warna merah (kotak), sedangkan pola pada class +1, disimbolkan dengan warna kuning (lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.11 (a).

Hyperplane pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tersebut. dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan pola terdekat dari masing-masing kelas. Pola yang paling dekat ini disebut sebagai *support vector*. Garis solid pada Gambar 2.11 (b) menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua kelas, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM.



(a) (b)

Gambar 2.11. *Hyperplane* terbaik yang memisahkan class-1 dan class +1

Tujuan dari klasifikasi SVM untuk menghasilkan model berdasarkan data pelatihan yang akan memprediksi kelas dari data yang di ujikan Diberikan *training set* (*xi, yi*), *i* = 1, ..,*l*, dimana xi ∈ R^n dan y ∈ {1,-1}, SVM dapat ditemukan dari solusi masalah optimasi berikut (Kim dkk., 2012) :

$$\min_{w,b,\xi} \frac{1}{2} W^T W + C \sum_{i=1}^{i} \xi i \quad \ldots\ldots\ldots (2)$$

Dimana $yi(W^T ø(xi) + b) \geq 1 - \xi_i, \xi_i \geq 0$. Disini vektor pelatihan *xi* dipetakan ke dimensi yang lebih tinggi dengan suatu fungsi *ø*. Kemudian SVM akan mencari sebuah *hyperplane* yang memisahkan secara linier dengan margin maksimal dalam dimensi yang lebih tinggi tersebut. $K(xi,xj) = ø(xi)^T ø(xj)$ disebut fungsi *kernel*. Fungsi *kernel* ini ada beberapa macam, diantaranya fungsi *linier*, *polynomial*, *radial basis function* (RBF) dan *sigmoid*.

Selain SVM, ada beberapa algoritma yang dapat digunakan untuk tujuan klasifikasi. Salah satunya adalah algoritma *machine learning K-Nearest Neighbor*.

F. K-Nearest Neighbor

K-Nearest Neighbor (KNN) merupakan metode yang menggunakan algoritma *supervised,* dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN (Li dkk., 2012). Tujuan algoritma ini adalah mengklasifikasi objek baru berdasakan atribut dan sampel latih. Pengklasifikasian tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik uji, akan ditemukan sejumlah K objek (titik training) yang paling dekat

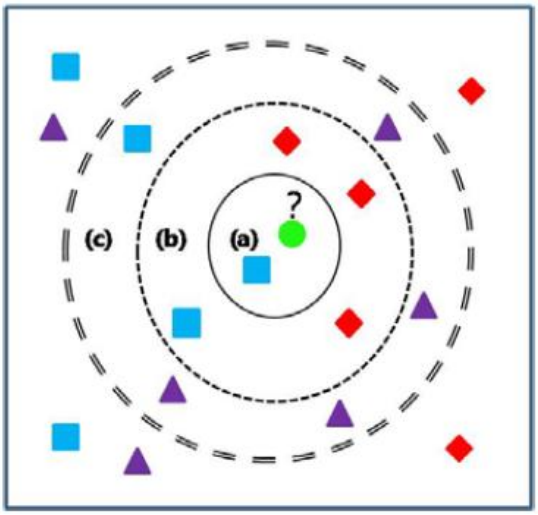
dengan titik uji. Klasifikasi menggunakan *voting* terbanyak di antara klasifikasi dari K objek. Algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *sample* uji yang baru. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak *Eucledian* seperti pada rumus (3) (Kim dkk., 2012).

$$d(x,y) = \left||x-y|\right| = \sqrt{(x-y).(x-y)}$$

$$= (\sum_{i=1}^{m}((xi-yi)^2)^{1/2} \quad \text{..... (3)}$$

Dimana *x* dan *y* adalah *histogram* dalam $X = R^m$. Gambar 2.12 memperlihatkan visualisasi dari proses klasifikasi KNN. Pada gambar tersebut, titik *query* dari lingkaran bergantung pada nilai *k* yaitu 1, 5 atau 10, titk *query* dapat berupa persegipanjang pada (a), berlian pada (b) dan segitiga pada (c). Gambar 2.12 memperlihatkan visualisasi dari proses klasifikasi KNN. Pada gambar tersebut, titik *query* dari lingkaran bergantung pada nilai *k* yaitu 1, 5 atau 10, titk *query* dapat berupa persegipanjang pada (a), berlian pada (b) dan segitiga pada (c) (Kim dkk., 2012).

.



Gambar 2.12 Klasifikasi KNN

Metode KNN sangatlah sederhana, bekerja dengan berdasarkan pada jarak terpendek dari sample uji ke sample latih untuk menentukan KNN-nya. Setelah mengumpulkan KNN, kemudian diambil mayoritas dari KNN untuk dijadikan prediksi dari *sample* uji.

Selain KNN, ada beberapa algoritma yang dapat digunakan untuk tujuan yang sama yaitu klasifikasi. Salah satunya adalah algoritma *machine learning Naïve Bayes*.

G. Naïve Bayes

Algoritma Naïve Bayes termasuk *supervised learning* yang memanfaatkan metode probabilitas dan statistik untuk memprediksi probabilitas di masa depan berdasarkan pengalaman di masa sebelumnya (Witten dkk., 2011). Algoritma ini berlandaskan teorema bayes dengan rumus:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \text{ .....(4)}$$

Dimana peluang kejadian A sebagai B ditentukan dari peluang B saat A, peluang A dan peluang B. Karena asumsi setiap atribut saling bebas (tidak ada ketergantungan antara atribut-attribut), sehingga rumusnya sebagai berikut :

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) \text{ .....(5)}$$

Bila *P(X|Ci)* dapat diketahui melalui perhitungan pada rumus (5), maka kelas dari data sampel X adalah kelas yang memiliki *P(X|Ci)*P(Ci)* maksimum. Hal inilah yang digunakan sebagai dasar dalam proses klasifikasi menggunakan algoritma *Naïve Bayes* (Witten dkk., 2011).

Selain *Naïve Bayes*, ada beberapa algoritma yang dapat digunakan untuk tujuan yang sama yaitu klasifikasi. Salah satunya adalah algoritma *machine learning C4.5 Decision Tree*.

H. C4.5 Decision Tree

Algoritma C4.5. mengkonsturksi *decision tree* (pohon keputusan) menggunakan strategi *divide and conquer* dari sebuah set data latih berupa kasus-kasus (*records*), dimana setiap kasus memiliki atribut kontinyu atau diskrit (Witten dkk., 2011). Secara umum, algoritma C4.5 dalam membangun pohon keputusan sebagai berikut:

- Pilih atribut sebagai akar
- Buat cabang untuk masing-masing nilai
- Bagi kasus dalam cabang
- Ulangi proses untuk masing-masing cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Untuk memilih atribut sebagai akar, didasarkan pada nilai *gain* tertinggi dari atribut-atribut yang ada. Untuk menghitung gain digunakan rumus (6):

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(Si) \quad \text{.....(6)}$$

Dengan :

S : Himpunan kasus

A : Atribut

n : Jumlah partisi Atribut A

|Si| : Jumlah kasus pada partisi ke-i

|S| : Jumlah kasus dalam S

Sedangkan nilai *entropy* dapat dihitung menggunakan rumus (7) :

$$Entropy(S) = \sum_{i=1}^{n} - pi * \log_2 pi \quad \text{.....(7)}$$

Dimana *pi* merupakan proporsi dari *Si* terhadap *S*.

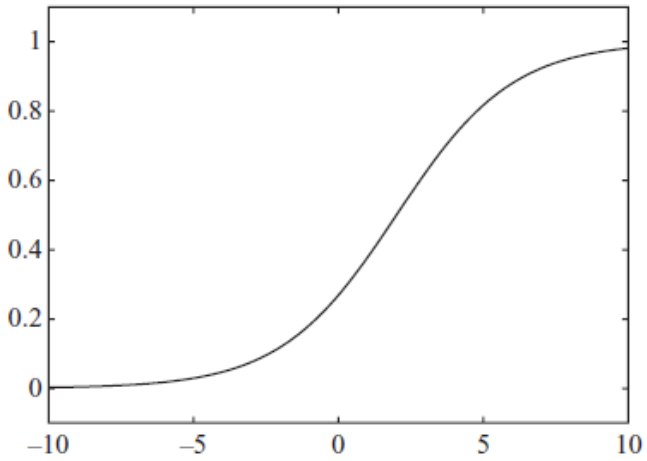
Selain *C4.5 Decision Tree*, ada beberapa algoritma yang dapat digunakan untuk tujuan yang sama yaitu klasifikasi. Salah satunya adalah algoritma *machine learning Logistic Regression*.

I. Logistic Regression

Logistic Regression (regresi logistik) merupakan model untuk mendapatkan persamaan regresi yang dapat memprediksi dua atau lebih kelompok objek yang dapat ditempatkan dalam dua kelas (Hastuti, 2012). Regresi logistik menggunakan transformasi *logit* dimana yang ditransformasi adalah peluang variabel respon sama dengan 1. Variable hasil transformasi tersebut ditebak menggunakan fungsi linier sama seperti yang dibangkitkan oleh regresi linier seperti pada rumus (6) (Witten, 2011) :

$$\Pr[1 \mid a_1, a_2, \ldots, a_k] = 1/(1 + \exp(-w_0 - w_1 a_1 - \ldots - w_k a_k)) \quad \ldots\ldots\ (6)$$

Dimana *w* adalah bobot (*weight*). Pada gambar 2.13 memperlihatkan contoh dari fungsi ini dalam satu dimensi, dengan bobot w0 = -1,25 dan w1 = 0,5.



Gambar 2.13. Contoh fungsi regresi logistik

Pada regresi linier, bobot yang ditemukan harus memenuhi kebutuhan data latih, dalam statistik dikenal dengan istilah *Goodness of fit*. Hal ini dihitung menggunakan *squared error*. Pada regresi logistik menggunakan *log-likelihood* (rasion kemungkinan) dengan rumus (7) :

$$\sum_{i=1}^{n}(1-x^{(i)})\log(1-\Pr[1 \mid a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}]) + x^{(i)}\log(\Pr[1 \mid a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}]) \quad \ldots\ldots (7)$$

Dimana $x^{(i)}$ bernilai 0 atau 1. Bobot *Wi* perlu dipilih untuk memaksimalkan *log-likelihood*. Ada beberapa metode untuk memecahkan masalah maksimisasi ini, salah cara sederhana yaitu mengatasi secara berulang setiap bobot yang masih tidak sesuai sampai *log-likehood* mencapai titik maksimum, biasanya dapat dicapai dalam beberapa iterasi (Witten, 2011).

Selain *Logistic Regression*, ada beberapa algoritma yang dapat digunakan untuk tujuan yang sama yaitu klasifikasi. Salah satunya adalah algoritma *machine learning Random Forest*.

J. Random Forest

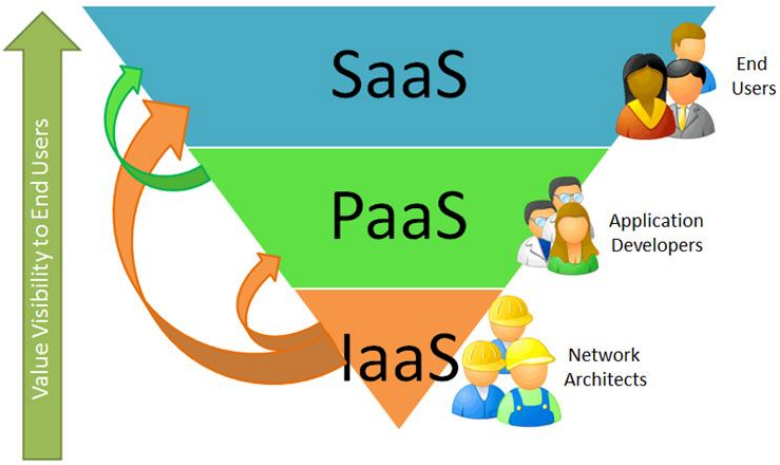
Random Forest merupakan pengembangan dari *Decision Tree* yang menggunakan beberapa *Decision Tree* yang telah dilakukan pelatihan menggunakan sampel individu. Setiap attribut dipecah pada *tree* (pohon) yang dipilih antara atribut subset yang bersifat acak (*random*). Proses klasifikasi individu didasari pada *vote* dari suara terbanyak pada kumpulan populasi pohon (Saputra dkk., 2011) .

Random Forest yang dihasilkan memiliki banyak pohon yang ditanam dengan cara yang sama. Pohon dengan variabel *x* akan ditanam sejauh mungkin dengan pohon dengan variabel *y*. Sejalan dengan bertambahnya data set, maka pohon pun ikut berkembang. Penempatan pohon yang saling berjauhan memberikan asumsi bahwa apabila terdapat pohon disekitar pohon *x*, itu berarti pohon tersebut merupakan perkembangan dari pohon *x*. Beberapa fungsi *learning* yang dihasilkan *random forest* menggunakan strategi *ensemble* ”*bagging*” untuk mengatasi masalah *overfitting* jika dihadapkan pada data set yang kecil.

Algoritma-algoritma yang telah dibahas dapat diimplementasikan dalam membangun aplikasi *machine learning*. Salah satu jenis aplikasi yang populer saat ini adalah jenis *Software as a Service* yang akan dibahas pada sub bab berikutnya.

K. Software as a Service

Software as a Service (SaaS) merupakan salah satu bentuk layanan *Cloud Computing*. SaaS menyediakan *service* dalam bentuk *software* yang sudah jadi dan dapat digunakan dimanapun dan kapanpun (selama terdapat akses ke internet) dengan syarat-syarat tertentu yang sudah di tentukan oleh penyedia jasa. Posisi atau *layer* SaaS dalam teknologi *Cloud Computing* (Furht dkk., 2012) dapat dilihat pada gambar 2.14.



Gambar 2.14. Arsitektur Layer *Cloud Computing*

Seperti yang terlihat pada gambar 2.14, layer aplikasi SaaS berada paling atas, artinya untuk menggunakan SaaS, *user* tidak lagi berinvestasi pada infrastruktur *hardware* maupun *software*, cukup menyediakan akses ke internet, aplikasi SaaS langsung dapat digunakan. Hal sangat memudahkan user karena *instalasi* software tidak lagi dibutuhkan dan *maintenance* serta *support* dapat ditangani langsung oleh *provider*.

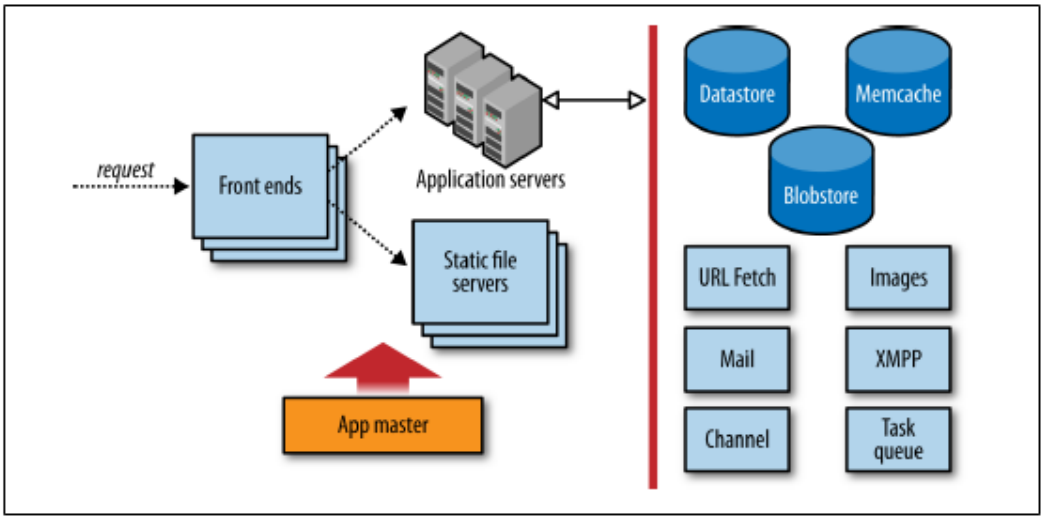
Aplikasi berbasis SaaS dibangun dengan konsep *multiuser*, dimana setiap *user* yang membutuhkan aplikasi tersebut dapat menggunakannya secara bersamaan. Hal ini akan tergantung pada kemampuan pengolahan data dari *server*, kecepatan dan kepadatan lalu lintas jaringan komputer. Sehingga dalam membangun aplikasi, perlu juga dipikirkan batasan-batasan atau aturan-aturan yang mengikat, agar aplikasi dapat digunakan secara optimal oleh setiap *user*.

Pada Gambar 2.14 juga memperlihatkan bahwa pihak *developer* tidak perlu berinvestrasi pada lapisan-lapisan yang ada di bawahnya untuk membangun aplikasi berbasis SaaS, cukup menggunakan *Platform as a Service* (PaaS) yang telah tersedia di internet. Salah satu PaaS yang dapat digunakan adalah layanan *Google App Engine* (GAE) dari google.

L. Google App Engine

Google App Engine (GAE) merupakan salah satu contoh PaaS. GAE menyediakan sekumpulan API (*Application Programming Interface*) dalam bentuk SDK (*Software Development Kit*) yang bisa diunduh secara bebas. Selain itu, produk GAE sendiri juga bebas atau gratis untuk digunakan dengan batasan-batasan *resources* yang diberikan, namun dalam beberapa hal tertentu, *resources* yang tersedia sudah mencukupi. Jika membutuhkan tambahan sumber daya sesuai keinginan, *user* dapat menggunakan versi berbayar (Sanderson, 2013).

Jenis sumber daya yang dibatasi di dalam GAE adalah pada *storage*, *bandwidth* dan waktu operasional dari aplikasi tersebut (dihitung per jam). Saat ini, untuk menggunakan GAE, anda dapat menggunakan dua macam bahasa pemrograman, yaitu JAVA dan PHYTON. Teknologi Java yang umum dipakai di GAE adalah Servlet dan JSP. Arsitektur GAE khususnya penanganan permintaan (*request handling*) secara umum dapat dilihat pada gambar 2.15 ((Sanderson, 2013).



Gambar 2.15. Arsitektur penanganan permintaan pada GAE

Apps Engine Request yang dimiliki oleh GAE menentukan jumlah *request* yang bisa ditangani oleh aplikasi. Jika menggunakan fasilitas berbayar, aplikasi dapat menerima 500 permintaan perdetik. Demikian juga dengan *DataStore* yang disediakan. Jika menggunakan fasilitas gratis, untuk melakukan penyimpanan data terbatas sebanyak 1 GB/hari, jika berbayar dapat menggunakan sesuai keinginan.

Apps Engine Request dengan komponen *URL Fetch Service*-nya, dapat berkomunikasi dengan aplikasi-aplikasi lainnya atau mengakses sumberdaya lainnya yang ada di Web dengan menggunakan URL (*Uniform Resource Locator*) yang dimiliki aplikasi/sumberdaya tersebut. Suatu aplikasi dapat menggunakan layanan ini untuk mengirim/menerima permintaan HTTP (*Hypertext Transport Protocol*) dan HTTPS (*Hypertext Transport Protocol Secure*).

Mengembangkan aplikasi menggunakan *Google App Engine for Java (GAE/J)* mirip dengan pengembangan aplikasi *Enterprise* menggunakan Java, kecuali *developer* tidak perlu terlalu memikirkan tentang jaringan komputer yang digunakan, perangkat keras, sistem operasi, basis data, atau server aplikasinya. Menggunakan GAE/J, *programmer* bisa langsung berinovasi dan mengembangkan aplikasi, tanpa harus melakukan hal-hal yang terlalu teknis seperti melakukan *setting* sistem operasi dan melakukan konfigurasi-konfigurasi basis data. GAE/J menyediakan *interface*s yang serupa dengan JVM (*Java Virtual Machine*) versi 6 dan Java Servlet, serta mendukung teknologi-teknologi Java yang umum seperti JDO (*Java Database Object*), JPA (*Java Persistence API*), *JavaMail*, dan sebagainya.

Aplikasi-aplikasi GAE/J dapat dikembangkan menggunakan IDE *Eclipse* dan *Google Plugin for Eclipse* yang menyediakan *server* pengembangan bersifat lokal, sehingga pengembang bisa mengembangkan aplikasi dari awal hingga akhir, sebelum men-*deploy*-nya menjadi *Java Google App Engine***.** Agar dapat digunakan secara *online* di internet, aplikasi di daftarkan ke https://appengine.google.com/ dengan syarat mempunyai *google account* dan telah teregitrasi pada GAE.

Setelah membahas beberapa teori pendukung, perlu dibahas penelitian-penelitian sebelumya yang menggunakan teori-teori tersebut.

Hal ini akan memperjelas posisi peneltian yang akan dilakukan melalui *roadmap* penelitian.

M. *Roadmap* Penelitian

Beberapa penelitian yang telah dilakukan menjadi ide dasar dalam penelitan ini, diantaranya sebagai berikut:

1. *Software as a Service for Data Scientists*. Penelitian ini menawarkan solusi teknologi SaaS untuk manajemen data penelitan, seperti kemudahan akses data antar para peniliti atau fasilitas-fasilitas penelitian (Allen dkk., 2012).
2. Implementasi Pengolahan Citra dan Algoritma LVQ untuk Pengenalan Pola Buku. Penelitian ini menggunakan Algoritma LVQ untuk klasifikasi buku sesuai sampulnya. Sebelum diolah menggunakan LVQ, citra terlebih dahulu dinormalisasi dan dikonversi menjadi citra *biner* (Lukman, 2012).
3. *Breast Tissue Classification Using Gabor Filter, PCA and Support Vector Machine* (Hajare dkk., 2012). Penelitian ini menggunakan algoritma SVM untuk klasifikasi citra mamografi untuk mengenali kanker payudara. Sebelum diklasifikasi, Citra terlebih dahulu melalui proses normalisasi, ekstraksi fitur menggunakan *filter gabor* dan reduksi dimensi citra menggunakan PCA.
4. *Using the K-Nearest Neighbor Algorithm for the Classification of Lymph Node Metastasis in Gastric Cancer* (Li dkk., 2012). Citra

terlebih dahulu melalui proses seleksi fitur untuk mereduksi dimesinya, kemudian digunakan KNN untuk klasifikasi. Penelitian ini membuktikan kelayakan dan efektivitas dari metode *machine learning* untuk diagnosa kelenjar getah bening metastatis pada kanker lambung menggunakan data GSI.

5. *Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines*" (Kim dkk., 2012). Penelitian ini mencoba membandingkan kemampuan klasifikasi citra metode KNN dan SVM menggunakan model *Bag of Word* (BoW). Dengan menggunakan model BoW, SVM lebih unggul dari KNN.
6. Analisis Komparasi Algoritma Klasifikasi Data Mining Untuk Prediksi Mahasiswa Non Aktif (Hastuti, 2012). Penelitian ini membandingkan 4 algoritma *data maining* yaitu *logistic regression*, *decision tree*, *naïve bayes* dan *neural network* untuk mendapatkan algoritma yang paling akurat dalam memprediksi mahasiswa non-aktif pada perguruan tinggi. Data set yang digunakan sebanyak 3861 mahasiswa Universitas Dian Nuswantoro terdiri dari data demografi dan akademik.
7. Seleksi Fitur Menggunakan Random Forest dan Neural Network (Saputra dkk., 2011). Penelitian ini
8. *Analysis of Machine Learning Algorithms using* WEKA (Hastuti, 2012). Penelitian ini mengimplementasikan aplikasi WEKA untuk

klasifikasi dan membandingkan performa waktu *learning* dan ketepatan klasifikasi beberapa Algoritma *Machine Learning*.

Penelitian no. 1 sebagai referensi dalam membangun aplikasi berbasis SaaS. 2, 3, 4 dan 6 digunakan sebagai referensi dalam membangun *machine learning* untuk kebutuhan klasifikasi dan pengenalan citra digital. Penelitian no 2 juga sebagai referensi teknik pra proses sistem yaitu normalisasi dan *grayscaling*.

Penelitian no. 5, 6 dan 7 sebagai referensi dalam membandingkan algoritma *machine learning*. Peneliian nomor 5 juga dijadikan referensi untuk teknik *bag of words* dalam pra-proses sistem. Penelitian no. 7 juga sebagai referensi dalam membangun aplikasi *machine learning* yang fleksibel dalam pemilihan algoritma dan parameternya menggunakan algoritma-algoritma yang telah tersedia pada *Machine Learning WEKA*.

Setelah *roadmap* penelitian, peneliti perlu memaparkan gambaran umum alur logika penelitiannya. Hal ini dapat dipaparkan melalui kerangka pikir penelitian.

N. Kerangka Pikir

Alur logika berjalannya penelitian dapat digambarkan dalam sebuah diagram kerangka pikir. Gambar 2.16 memperlihatkan kerangka pikir penelitian ini.

Gambar 2.16. Kerangka pikir penelitian