TUGAS AKHIR – SS 141501

PERBANDINGAN METODE *RANDOM FOREST CLASSIFICATION* DAN *SUPPORT VECTOR MACHINE* UNTUK DETEKSI EPILEPSI MENGGUNAKAN DATA REKAMAN *ELECTROENCEPHALOGRAPH* (EEG)

Muhammad Idrus Fachruddin
NRP 1311 100 104

Dosen Pembimbing
Dr.rer. pol. Heri Kuswanto, S.Si., M.Si.

Program Studi S1 Statistika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – SS 141501

COMPARISON BETWEEN RANDOM FOREST METHOD AND SUPPORT VECTOR MACHINE METHOD FOR EPILEPTIC SEIZURE DETECTION USING ELECTROENCEPHALOGRAPH DATA RECORD

Muhammad Idrus Fachruddin
NRP 1311 100 104

Supervisor
Dr.rer. pol. Heri Kuswanto, S.Si., M.Si.

Undergraduate Programme of Statistics
Faculty of Mathematics and Natural Sciences
Institut Teknologi Sepuluh Nopember
Surabaya 2015

LEMBAR PENGESAHAN

PERBANDINGAN METODE *RANDOM FOREST CLASSIFICATION* DAN *SUPPORT VECTOR MACHINE* UNTUK DETEKSI EPILEPSI MENGGUNAKAN DATA REKAMAN *ELECTROENCEPHALOGRAPH* (EEG)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Studi S-1 Jurusan Statistika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember

Oleh :
MUHAMMAD IDRUS FACHRUDDIN
NRP. 1311 100 104

Disetujui oleh Pembimbing Tugas Akhir :

Dr.rer.pol. Heri Kuswanto, S.Si., M.Si.
NIP. 19820326 200312 1 004

Mengetahui
Ketua Jurusan Statistika FMIPA-ITS

Dr. Muhammad Mashuri, MT.
NIP. 19620408 198701 1 001

SURABAYA, JULI 2015

Perbandingan Metode *Random Forest Classification* Dan *Support Vector Machine* Untuk Deteksi Epilepsi Menggunakan Data Rekaman *Electroencephalograph* (EEG)

Nama	:	Muhammad Idrus Fachruddin
NRP	:	1311 100 104
Jurusan	:	Statistika FMIPA – ITS
Pembimbing	:	Dr. rer. Pol. Heri Kuswanto, S.Si, M.Si

ABSTRAK

Diagnosa terhadap penderita epilepsi dilakukan dengan cara menyaksikan langsung terjadinya serangan sehingga menyulitkan dokter untuk melakukan diagnosa. Cara paling mutakhir untuk mendiagnosa penyakit epilepsi adalah dengan rekaman *electroencephalograph* (EEG).Namun, dengan menggunakan EEG dibutuhkan waktu lama untuk menentukan seseorang menderita epilepsi atau tidak.Oleh karena itu, dibutuhkan metode yang cepat dan akurat untukmendiagnosa penyakit epilepsi.Metode statistika klasifikasi bisa digunakan untuk mengatasi masalah tersebut.Pada penelitian ini, digunakan dua metode klasifikasi yakni *random forest* dan *support vector machine* (SVM) kemudian dibandingkan performanya. Sebelum dilakukan klasifikasi data di *preprocessing* terlebih dahulu menggunakan *discrete wavelet transform* (DWT) dan *line length feature* (LLF). Hasil analisa menunjukkan bahwa metode *random forest* memiliki rata-rata akurasi yang lebih baik dari pada SVM untuk data *training*.Untuk data *testing*, SVM memiliki rata-rata akurasi yang lebih baik dari pada *random forest*.

***Kata Kunci**:Epilepsi, EEG, Discrete Wavelet Transform, Line Length Feature, Random Forest, Support Vector Machine*

(Halaman ini sengaja dikosongkan)

Comparison Between Random Forest Method And Support Vector Machine Method For Epileptic Seizure Detection Using Electroencephalograph Data Record

Name of Student : Muhammad Idrus Fachruddin
NRP : 1311 100 104
Departement : Statistics
Supervisor : Dr. rer. pol. Heri Kuswanto, S.Si., M.Si

ABSTRACT

Diagnosis patient with epileptic seizure do by direct witnessed the attack. It making difficult for the doctor to diagnose. The recent method use for epileptic seizure diagnose is using elctroencephalograph (EEG) record. However, diagnose using EEG record takes a long time to determine a person suffering from epileptic or not. Therefore, it takes quick and accurate method to diagnose epileptic seizure. Statistics classification method can be used to resolve this problem. This study used two classification method that is random forest (RF) and support vector machine (SVM). The two method used to predict epileptic seizure and then compare its performance. Before classification, the EEG data record preprocessed using discrete wavelet transform (DWT) and line length feature (LLF) extraction. The result shows that RF method has better average accuration than SVM method for training data. However, for the testing data SVM has better average accuration.

Keywords : *Epileptic, EEG, Discrete Wavelet Transform, Line Length Feature, Random Forest, Support Vector Machine*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada Allah SWT, yang telah memberikan rahmat dan rahim Nya sehingga penyusunan Tugas Akhir ini dapat terselesaikan tepat waktu. Tugas Akhir yang berjudul "**Perbandingan Metode *Random Forest Classification* dan *Support Vector Machine*untuk Deteksi Epilepsi Menggunakan Data Rekaman *Electroencephalograph* (EEG)**" ini disusun untuk memenuhi salah satu syarat kelulusan Program Studi S1 Jurusan Statistika FMIPA ITS.

Dengan terselesaikannya penyusunan Tugas Akhir ini, penulis mengucapkan terima kasih kepada:

1. Allah swt yang telah memberikan kemudahan dan kelancaran dalam menjalankan Tugas Akhir sampai dengan penyusunan laporan.
2. Dr. rer. pol. Heri Kuswanto, S.Si., M.Si. selaku dosen pembimbing yang selalu memberikan pengarahan dan bimbingan kepada penulis selama penyusunan laporan Tugas Akhir.
3. Dr. Muhammad Mashuri, MT selaku Ketua Jurusan Statistika ITS.
4. Dra. Lucia Aridinanti, MS selaku Kaprodi S1 Jurusan Statistika ITS.
5. Prof. Drs. Nur Iriawan, MI.Kom, Ph.D dan Dr. Kartika Fitriasari selaku dosen penguji yang senantiasa memberikan kritik dan saran demi kesempurnaan Tugas Akhir ini.
6. Sjahid Akbar, S.Si M.Si dan Dr. Santi Putri Rahayu, M.Si selaku dosen wali yang telah memberikan pengarahan selama proses perkuliahan.
7. Seluruh dosen Statistika ITS dan dosen non Statistika ITS yang telah memberikan ilmu-ilmu yang tiada ternilai harganya dan segenap karyawan jurusan Statistika ITS.
8. Bapak dan Ibu yang selalu memberikan dukungan dan kasih sayangserta doa yang tiada henti.

9. Adik-adikku Irfa, Thoriq dan Althaf yang menjadi motivasi bagi penulis untuk menjadi semakin baik.
10. Seluruh teman-teman sigma 22 Statistika ITS angkatan 2011 yang selalu memberikan banyak pelajaran berharga bagi penulis
11. *All* pengurus FORSIS-ITS KHK atas bantuan dan dukungannya.
12. PH KKH JMMI yang menginspirasi dan memotivasi.
13. GGS dan Srikandi Syiar JMMI atas ukhuwah dan kontribusinya yang luar biasa.
14. Sindu, Epa, Hayam, Fidah, Ita dan Mbak Dian atas *sharing* ilmu tentang topik tugas akhir ini

Penulis menyadari sepenuhnya bahwa laporan Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis menerima kritik dan saran yang membangun bagi perbaikan di masa yang akan datang. Semoga laporan ini bermanfaat bagi penelitian selanjutnya.

Surabaya,8 Juni 2015

Penulis

DAFTAR ISI

Halaman

(Halaman ini sengja dikosongkan)

DAFTAR TABEL

Halaman

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Halaman

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jumlah penderita penyakit epilepsi atau dikenal juga dengan sebutan ayan di Indonesia masih cukup tinggi. Persepsi yang muncul di masyarakat terhadap penyakit epilepsi pun masih buruk. Kebanyakan orang beranggapan bahwa penyakit epilepsi adalah penyakit menular bahkan kutukan. Hal tersebut tentu saja semakin menambah penderitaan penderita epilepsi. Selain harus menahan beratnya penderitaan saat epilepsi menyerang, penderita juga seringkali dikucilkan di tengah masyarakat. Ada sebagian yang menganggap bahwa epilepsi adalah penyakit yang tidak bisa disembuhkan sehingga membiarkan penderita epilepsi tanpa memberikan terapi khusus yang dapat memberikan harapan kepada penderita untuk sembuh total sebelum dewasa.

Epilepsi adalah salah jenis penyakit yang menyerang syaraf. Epilepsi didefinisikan sebagai kumpulan gejala dan tanda-tanda klinis yang muncul disebabkan oleh gangguan fungsi otak secara intermiten. Gejala tersebut terjadi akibat terlepasnya muatan listrik abnormal atau berlebihan dari neuron-neuron secara paroksismal dengan berbagai macam etiologi. Sedangkan serangan atau bangkitan epilepsi yang dikenal dengan istilah *epileptic seizure* adalah manifestasi klinis yang serupa dan berulang secara paroksismal yang diakibatkan oleh hiperaktivitas listrik sekelompok sel saraf di otak yang spontan dan bukan disebabkan oleh suatu penyakit otak akut (Shovron, 2000).Gejala yang biasa timbul saat epilepsi menyerang umumnya adalah tatapan mata kosong, kejang total (*totalconvulsions*), kedutan (*twitching*) dan aura (doktersehat.com). Gejala – gejala tersebut muncul sesuai dengan tingkat keparahan penderita.

Diagnosa terhadap pasien yang menderita epilepsi dilakukan dengan cara menyaksikan langsung terjadinya serangan. Namun, permasalahannya adalah dokter kebanyakan tidak bisa menyaksikan langsung serangan epilepsi pada penderita.

Penderita pun tidak sadar dan tidak tahu sama sekali bahwa dirinya baru saja terkena serangan epilepsi. Diagnosa epilepsi hampir selalu dilakukan berdasarkan alloanamnesis yang bersumber dari keterangan yang diceritakan oleh orang sekitar, cara tersebut tentu saja tidak khas dan kurang akurat. Cara paling mutakhir yang dilakukan untuk mendiagnosa penyakit epilepsi adalah dengan rekaman *electroencephalograph* (EEG).

EEG adalah rekaman aktivitas listrik neuron otak. Fluktuasi arus listrik tersebut didapatkan dari perbedaan voltase yang diukur dari elektrode yang ditempel di kulit kepala (*skalp*), langsung dipermukaan kortek serebri atau di dalam jaringan otak (Hughes, 1994). EEG menjadi sebuah hal penting bagi para praktisi kesehatan untuk membantu mereka dalam melakukan diagnosa dan monitoring di bidang neurologi.

Pada penerapannya untuk monitoring pasien epilepsi, pemeriksaan menggunakan EEG dilakukan secara terus menerus selama beberapa hari. Akibatnya, sebagian besar data harus diamati secara visual oleh para ahli agar penyakit epilepsi bisa diidentifikasi. Cara tersebut tentunya kurang efisien karena membutuhkan waktu yang lama dan biaya yang besar. Oleh sebab itu, perlu adanya pembuatan sistem pembuat keputusan yang cepat dan akurat. Pada penelitian ini akan dilakukan diagnosa penyakit epilepsi melalui rekaman EEG dengan menggunakan *wavelet transform discrete*dan pendekatan *Random Forest Classification* (RF)serta *Support Vector Machine* (SVM).

RF dan SVM adalah metode statistika berbasis komputasi. RF digunakan untuk kasus klasifikasi. Sedangkan SVM bisa digunakan untuk kasus prediksi dan klasifikasi. Memasuki era *big data* penggunaan metode statistika berbasis komputasi sangat banyak digunakan. Hal tersebut dikarenakan metode berbasis komputasi mampu memberikan hasil keputusan yang cepat, akurat dan tidak membutuhkan asumsi yang ketat seperti metode statistika klasik.

Discrete Wavelet tramsform (DWT*)* digunakan untuk menguraikan sinyal EEG menjadi frekuensi yang berbeda. DWT

adalah metode yang sangat direkomendasikan untuk analisis waktu dan frekuensi. Dengan menguraikan sinyal menjadi elemen yang terlokalisasi secara baik untuk frekuensi maupun waktu, DWT mampu mengarakterisasi keteraturan lokal (Mallat dan Hwang, 1992).

Sebelumnya, telah dilakukan beberapa penelitian terkait *signal processing* dengan menggunakan *wavelet transform*. Beberapa diantaranya adalah Cuiwei, Chongxun dan Changfeng (1995) yang mendeteksi karakteristik poin pada sinyal ECG. Pada penelitiaan tersebut diketahui bahwa WT bisa digunakan untuk membedakan sinyal ECG walaupun dengan gangguan (*noise*) yang serius. Sementara itu, Pablo, Almeida dan Salvador (2004) juga menggunakan WT untuk mengevaluasi standar database sinyal ECG delineator. Sementara penelitian tentang diagnosa penyakit epilepsi telah dilakukan oleh Tawade (2011), Aris (2012) dan Ardilla (2014). Tawade menggunakan metode *Neural Network* untuk klasifikasi. Sedangkan Ardilla menggunakan *K-Means clustering* dan *Multilayer perceptron* untuk klasifikasi. Sementara itu, Subasi dan Gorsay (2010) menggunakan SVM dengan kombinasi *factor reduction* PCA, ICA dan LDA untuk mengklasifikasikan sinyal EEG. Kemudian Rahman (2012) menggunakan SVM untuk mengklasifikasikan tingkat keganasan *breast cancer*. Dan Darmawan (2014) menggunakan SVM untuk klasifikasi penderita kanker serviks.

Pada penelitian ini, klasifikasi akan dilakukan dengan menggunakan metode RF dan SVM pada data pasien epilepsi yang telah direkam dengan menggunakan EEG. Kemudian, hasil kedua metode yang digunakan akan dibandingkan untuk mengetahui metode mana yang lebih baik performansinya.

1.2 Rumusan Masalah

Diagnosa penyakit epilepsi selama ini masih menggunakan cara yang kurang efisien dari segi waktu dan tenaga. Perlu dirumuskan sebuah metode yang mampu membanti para dokter untuk mendiagnosa penderita epilpse secara cepat dan akurat

berdasarkan rekaman EEG. Dua metode statistika yang sedang berkembang saat ini yakni SVM dan *random forest* bisa digunakan untuk menjawab masalah tersebut. Sehingga dalam penelitian ini dirumuskan masalah yang akan dibahas, yakni :

1. Bagaimana *preprocessing* yang dilakukan pada data sinyal *Electroencephalograph*?
2. Bagaimana deteksi penyakit epilepsi dengan menggunakan metode *Random Forest* dan *Support Vector Machine* ?
3. Bagaimana ketepatan klasifikasi dari metode *Random Forest* dan *Support Vector Machine*?

1.3 Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah.

1. Mengetahui transformasi sinyal *Electroencephalograph* dengan menggunakan *Discrete Wavelet Transform*
2. Mengetahui deteksi penyakit epilepsi dengan menggunakan metode *Random Forest* dan *Support Vector Machine*
3. Mengetahui ketepatan klasifikasi dari metode *Random Forest* dan *Support Vector Machine*

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah dapat mengetahui penerapan metode DWT untuk memonitor dan mengevaluasi sinyal EEG. Kemudian dapat mengetahui perbandingan metode *Random Forest* dan *Support Vector Machine* dalam mengklasifikasikan penderita epilepsi melalui EEG.Kombinasi metode tersebut dapat digunakan sebagai akternatif dalam mendiagnosis penyakit epilepsi secara cepat dan akurat.

BAB II
TINJAUAN PUSTAKA

2.1 Transformasi Sinyal

Transformasi matematik digunaan untuk memperoleh informasi lebih lanjut dari sebuah sinyal. Pada praktiknya, sinyal kebanyakan tersedia dalam bentuk domain waktu. Atau dengan kata lain, saat di plot sumbu *x* merupakan waktu dan sumbu *y* adalah amplitudo sehingga akan dihasilkan representasi waktu-amplitudo. Representasi tersebut belum memberikan informasi yang terbaik dalam beberapa penerapan. Justru, dalam banyak kasus, informasi penting dari sebuah sinyal terdapat di konten frekuensi dari sinyal tersebut. Atau dengan kata lain, dibutuhkan sinyal dalam bentuk domain frekuensi untuk melakukan analisis yang mendalam pada suatu sinyal. Salah satu metode yang digunakan untuk merubah sinyal dari domain waktu menjadi domain frekuensi adalah *fourier transform* (FT). Namun, FT hanya bisa digunakan saat suatu sinyal besifat stasioner atau tidak berubah terhadap waktu. Padahal, sangat banyak sinyal di dunia nyata yang bersifat nonstasioner misalnya adalah sinyal EEG. Untuk menganalisa sinyal yang tidak stasioner dibutuhkan informasi mengenai domain frekuensi sekaligus juga domain waktu.Metode transformasi yang mampu menyediakan domain frekuensi dan waktu sekaligus adalah *wavelet transform.*

2.1.1 Wavelet Transform

Sinyal WT didefinisikan sebagai :

$$\psi^{a,b}(t) = |a|^{-\frac{1}{2}} \psi\left(\frac{t-b}{a}\right). \qquad (2.1)$$

Parameter *b* dalam persamaan (2.1) menyatakan posisi dari *wavelet.*Sedangkan parameter *a* menyatakan *scale* (timbal balik dari frekuensinya). Untuk $|a| < 1$, *wavelet* terkonsentrasi sangat tinggi dengan konten frekuensi mayoritas berada di dalam jarak

frekuensi yang tinggi. Begitu juga sebaliknya, jika $|a| > 1$, *wavelet* menyebar dan memiliki frekuensi yang rendah (Antonini dan Barlaud, 1992).

WT adalah metode yang digunakan untuk mentransformasi data dalam bentuk sinyal. Misalnya adalah data sinyal akustik seperti musik ataupun suara. WT memberikan hasil yang lebih baik untuk analisis tipe data sinyal dari pada metode *Short-time Fourier Transform* (Daubechies, 1990).

Ide dasar dari transformasi *wavelet* adalah menyajikan beberapa fungsi f yang terpisah-pisah dalam bentuk fungsi gabungan. Kemudian fungsi gabungan f diuraikan ke dalam level skala yang berbeda, dimana masing-masing level kemudian diuraikan lebih lanjut dengan resolusi yang telah disesuaikan. Fungsi f ditulis sebagai integral dari a sampai b dari $\psi^{a,b}$ dengan pendekatan koefisien terboboti. Berikut adalah persamaan untuk *continuous wavelet transform*.

$$CWT = \frac{1}{\sqrt{|b|}} \int x(t) \psi * \left(\frac{t-a}{b} \right) dt$$

Pada praktiknya, lebih dipilih fungsi diskrit lebih mudah dalam perhitungan tanpa menghilangkan informasi pentingnya. Bentuk diskrit didefinisikan sebagai $a = a_0^m, b = nb_0 a_0^m$ dengan $m, n \in Z$ dan $a_0 > 1$, $b_0 > 1$. Dekomposisi w*avelet* kemudian dinyatakan dalam persamaan (2.2) di bawah ini:

$$f = \sum c_{m,n}(f) \psi_{m,n} \qquad (2.2)$$

dengan $\psi_{m,n}(t) = \psi^{a_0^m, nb_0 a_0^m}(t) = a_0^{-m/2} \psi(a_0^{-m} t - nb_0)$.

2.1.2 Disrete Wavelet Transform

Prosedur implementasi *Dicrete Wavelet Transform* (DWT) secara efisien dilakukan dengan melewatkan sinyal melalui *low pass* dan *high pass filter*. Proses tersebut dikenal dengan istilah *subband coding.*

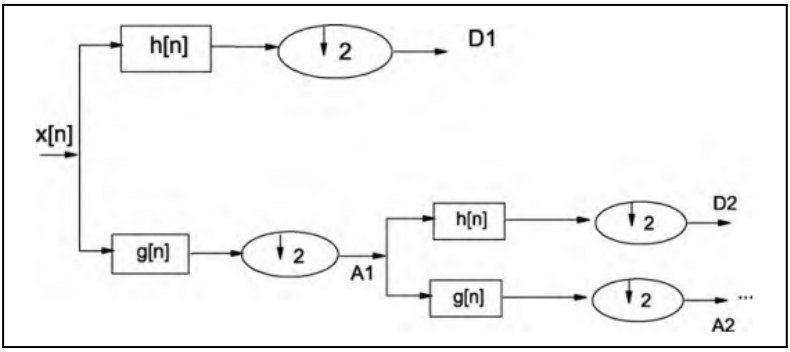
Proses dekomposisi sinyal yang dilakukan menggunakan metode *subband coding* seperti di tunjukkan pada Gambar 2.1. *Subband coding* adalah proses dekomposisi sinyal menjadi frekuensi yang berbeda dan diperoleh dari proses filter *high pass* dan *low pass* secara terus-menerus. Sinyal asli yang dinotasikan dengan x[n] pertama-tama harus melewati filter *high pass* g[n] dan *low pass* h[n]. setelah proses penyaringan, setengah dari jumlah sampel dieliminasi berdasarkan aturan Nyquist. Kemudian sinyal yang tersisa dibagi lagi menjadi dua. Proses tersebut merupakan proses dekomposisi satu level dan secara matematis dinotasikan dengan:

$$y_{high}[k] = \sum_{n} x[n].g[2k-n], \qquad (2.3)$$

$$y_{low}[k] = \sum_{n} x[n].h[2k-n], \qquad (2.4)$$

dimana $y_{high}[k]$ pada persamaan (2.3) dan $y_{low}[k]$ pada persamaan (2.4) adalah *output* dari filter *high pass* dan *low pass.* Proses *filtering* menggunakan metode linier convolution dimana $x[n]$ adalah fungsi sinyal yang menjadi *inputfiltering*, fungsi $g[2k-n]$ pada persamaan (2.3) dan $h[2k-n]$ pada persamaan 2.4 adalah fungsi koefisien wavelet untuk *high pass* dan *low pass*. *n* adalah panjang sinyal yang menjadi *input*, *k* adalah panjang koefisien wavelet yang digunakan seperti yang ditunjukkan Lampiran 13. Outputdari *high pass filter* adalah *details* (*D*) dan

output dari *low pass filter* adalah *aproximation* (*A*). Dekomposisi satu level akan menghasilkan *output*$D1$ dan $A1$. Pada level dua, *output* $A1$ akan difilter lagi hingga menghasilkan *output D2* dan *A2*. Proses tersebut berulang hingga *l*-level yang diinginkan. *D1*,*D2*,…,*Dl* dan *Al* akan menjadi varibel prediktor dalam proses klasifikasi.



Gambar 2. 1*Subband Coding Decomposition*

2.2 Line Length Feature Extraction (LLF)

Line length adalah ukuran kompleksitas sinyal atau *waveform fractal dimension*. Esteller et al. (2001) menyatakan bahwa *Line length* sensitif terhadap amplitudo dan variasi frekuensi sinyal sehingga berlaku juga sebagai ukuran kekejangan atau patologi dari karakteristik kombinasi amplitudo dan frekuensi pada sinyal EEG. Esteller et al. (2001) mendefinisikan *Line length*, *L* sebagai berikut:

$$L = \frac{1}{N-1}\sum_{i=1}^{N-1} abs(x_{i+1} - x_i) \qquad (2.5)$$

dimana *x* pada persamaan (2.5) adalah sinyal, *i* merepresentasikan indeks dari sampel sinyal dan *N* adalah banyaknya sampel yang digunakan. LLF mampu menyajikan karakteristik amplitudo dan frekuensi dari EEG dengan baik sehingga baik untuk deteksi eplipesi (Esteller et al., 2004). Alasan lainnya adalah LLF tidak

membutuhkan perhitungan yang rumit sehingga memungkinkan untuk mengembangkan *real time seizure detection system* di masa mendatang.

2.3 Random Forest

Random Forest adalah metode klasifikasi terbaru berbasis komputasi yang diperkenalkan oleh Breiman tahun 2001. *Random Forest* juga merupakan pengembangan dari metode CART sehingga mempunyai beberapa kelebihan yang tidak dimiliki metode CART.

2.3.1 Pengertian *Random Forests*

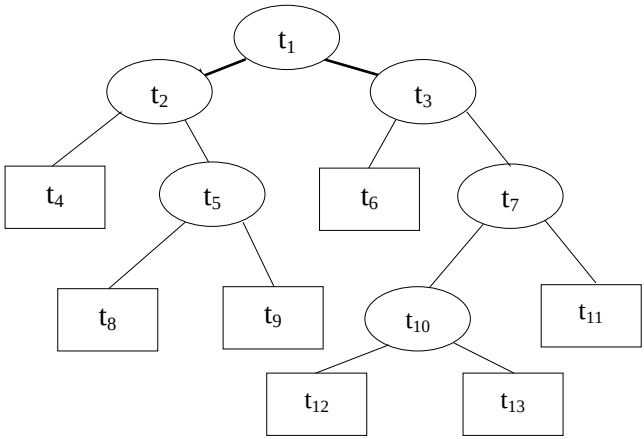
Random forests adalah suatu metode klasifikasi yang terdiri dari gabungan pohon klasifikasi (CART) yang saling independen. Prediksi klasifikasi diperoleh melalui proses *voting* (jumlah terbanyak) dari pohon-pohon klasifikasi yang terbentuk.*Random forests* merupakan pengembangan dari metode *ensemble* yang pertama kali dikembangkan oleh Leo Breiman (2001) dan digunakan untuk meningkatkan ketepatan klasifikasi. Bila dalam proses *bagging* digunakan *resampling bootstrap* untuk membangkitkan pohon klasifikasi dengan banyak versi yang kemudian mengkombinasikannya untuk memperoleh prediksi akhir, maka dalam *random forests* proses pengacakan untuk membentuk pohon klasifikasi tidak hanya dilakukan untuk data sampel saja melainkan juga pada pengambilan variabel prediktor. Sehingga, proses ini akan menghasilkan kumpulan pohon klasifikasi dengan ukuran dan bentuk yang berbeda-beda. Hasil yang diharapkan adalah suatu kumpulan pohon klasifikasi yang memiliki korelasi kecil antar pohon. Korelasi yang kecil akan menurunkan hasil kesalahan prediksi *Random Forests* (Breiman, 2001).

2.3.2 Pohon Klasifikasi

Sistem pohon keputusan (*decision trees*) berbeda proses yang terjadi pada metode regresi logistik, pohon keputusan (*decision trees*) bukanlah model parametrik sehingga tidak

memiliki parameter. Pohon disajikan berupa rangkaian yang terdiri dari pilihan dan keputusan. Keputusan digambarkan seperti pohon asli namun ditampilkan secara terbalik. Ada dua macam pohon keputusan yakni pohon klasifikasi dan pohon regresi. Pada penelitian ini akan digunakan pohon klasifikasi.

Pohon klasifikasi seperti yang diilustrasikan pada Gambar 2.2 dibuat dari kumpulan simpul yang saling terhubung. Masing-masing simpul mewakili sebuah keputusan, dimana keputusan didasarkan pada salah satu variabel prediktor. Untuk setiap keputusan, proses akan turun satu level dari pohon ke simpul lainnya hingga mencapai simpul terminal yang dinamakan daun (*leaf*). Daun berisi level dari respon dan diakhiri dengan menentukan kelas.



Gambar 2.2 Ilustrasi Pohon Klasifikasi

2.3.3 Pembentukan Pohon Klasifikasi

Saat membentuk sebuah pohon klasifikasi, pendekatan yang dilakukan adalah dengan memisahkan sebuah masalah menjadi sub masalah. Keputusan pada simpul teratas akan dipisahkan menjadi dua simpul dimana kedua simpul tersebut terdiri dari pernyataan benar dan salah.

Ada beberapa algoritma yang digunakan untuk memilih keputusan misalnya adalah indeks Gini yang didefinisikan sebagai :

$$G = \sum_{i=1}^{I} p_{mi}(1 - p_{mi}), \tag{2.6}$$

dimana p_{mi} adalah proporsi pengamatan pada kelas *i* dan simpul ke *m* (Kalmar, 2014). Persamaan (2.6) merupakan ukuran keheterogenan simpul. Himpunan bagian yang dihasilkan dari proses pemilihan harus lebih homogen dibandingkan simpul induknya. Kemudian langkah selanjutnya adalah menentukan kriteria *goodness of split.* Cara yang dilakukan adalah dengan mengoptimumkan fungsi indeks Gini.

Persamaan (2.6) adalah pernyataan derajat kedua sehingga akan memiliki satu titik ekstrim. Dengan menurunkan persamaan (2.6) hingga turunan kedua, titik ekstrim akan bisa diperoleh. Turunan pertama dari persamaan (2.5) adalah

$$\frac{d}{dp_{mi}} p_{mi}(1 - p_{mi}) = 1 - 2p_{mi} \tag{2.7}$$

saat persamaan (2.7) sama dengan 0, maka akan diperoleh titik ekstrim, yakni :

$$1 - 2p_{mi} = 0$$

$$2p_{mi} = 1$$

$$p_{mi} = 0{,}5$$

sehingga 0,5 adalah posisi dari titik ekstrim. Turunan kedua dari persamaan (2.6) adalah

$$\frac{d}{dp_{mi}} 1 - 2p_{mi} = -2.$$

sehingga 0,5 adalah posisi dari titik ekstrim. Turunan kedua dari persamaan (2.6) adalah

$$\frac{d}{dp_{mi}} 1 - 2p_{mi} = -2 .$$

Nilai dari turunan kedua adalah negatif sehingga titik p_{mi} = 0,5 adalah titik maksimum. Nilai rasio harus berada pada interval 0 sampai 1 sehingga persamaan diatas akan minimum dengan nilai yang mendekati 0 atau 1. Hal ini menunjukkan bahwa dengan rasio yang tinggi atau rendah, indeks gini akan bernilai paling kecil. Simpul dengan nilai rasio yang tinggi atau yang rendah akan memiliki indeks Gini yang kecil. Memilih pemisah dengan indeks Gini yang kecil akan menjadikan simpul yang memisahkan pengamatan menjadi simpul yang homogen.

2.3.4 Algoritma *Random Forests*

Secara umum, pengembangan *random forests* yang dilakukan dari proses *bagging* yaitu terletak pada proses pemilihan pemilah. Pada *random forests*, pemilihan pemilah hanya melibatkan beberapa variabel prediktor yang terambil secara acak. Algoritma *Random Forests* dijelaskan sebagai berikut:

Langkah a. Mengambil *n* data sampel dari *dataset* awal dengan menggunakan teknik *resampling bootstrap* dengan pengembalian.

Langkah b. Menyusun pohon klasifikasi dari setiap *dataset* hasil *resampling bootstrap*, dengan penentuan pemilah terbaik didasarkan pada variabel prediktor yang diambil secara acak. Jumlah variabel yang diambil secara acak dapat ditentukan melalui perhitungan $\log_2(M+1)$, dimana M adalah banyak variabel prediktor (Breiman, 2001) atau menggunakan $\sqrt{p}$, dimana p adalah banyak variabel prediktor (Genuer et al., 2009).

Langkah c. Melakukan prediksi klasifikasi data sampel berdasarkan pohon klasifikasi yang terbentuk.

Langkah d. Mengulangi langkah a sampai langkah c hingga diperoleh sejumlah pohon klasifikasi yang diinginkan. Perulangan dilakukan sebanyak *K* kali.

Langkah e. Melakukan prediksi klasifikasi data sampel akhir dengan mengombinasikan hasil prediksi pohon klasifikasi yang diperoleh berdasarkan aturan *majority vote.*

Analisis dengan menggunakan metode *random forests* dimulai dari pengambilan data dengan teknik *resampling bootstrap*. *Bootstrap* adalah suatu metode yang dapat bekerja tanpa membutuhkan asumsi distribusi karena sampel asli digunakan sebagai populasi. *Bootstrap* digunakan untuk mencari distribusi sampling dari suatu estimator dengan prosedur *resampling* dengan pengembalian dari data asli (Sungkono, 2013). Algoritma dari *resampling bootstrap* diilustrasikan pada Gambar 2.3 dengan langkah-langkah sebagai berikut:

Langkah a. Mengkonstruksi distribusi empiris $\hat{F}_n$ dari suatu sampel dengan memberikan probabilitas 1/*n* pada setiap X_i dimana *i*=1, 2, ..., *n.*

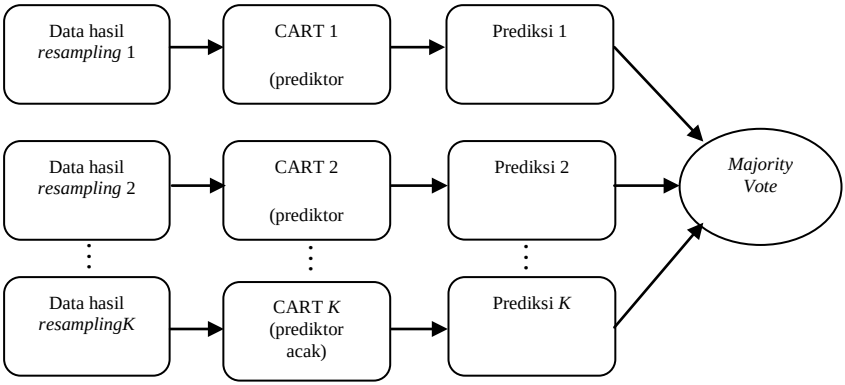
Langkah b. Mengambil sampel *bootstrap* berukuran *n* secara random dengan pengembalian dari distribusi empiris $\hat{F}_n$ sebut sebagai sampel *bootstrap* pertama X^{*1}.

Langkah c. Menghitung stastistik $\hat{\theta}$ yang diinginkan dari sampel *bootstrap*X^{*1} sebut sebagai $\hat{\theta}_1{}^*$.

Langkah d. Mengulangi langkah b dan langkahc hingga *B* kali diperoleh $\hat{\theta}_1{}^*, \hat{\theta}_2{}^*, \dots, \hat{\theta}_B{}^*$.

Langkah e. Mengkontruksi suatu distribusi probabilitas dari $\hat{\theta}_B{}^*$ dengan memberikan probabilitas *1/B* pada setiap $\hat{\theta}_1{}^*, \hat{\theta}_2{}^*, \dots, \hat{\theta}_B{}^*$. Distribusi tersebut merupakan estimator *bootstrap* untuk distribusi sampling $\hat{\theta}$ dan dinotasikan dengan $\hat{F}^*$.

Langkah f. Pendekatan estimasi *bootstrap* adalah $\hat{\theta}^* = \sum_{b=1}^{B} \hat{\theta}_b{}^* \frac{1}{B}$



Gambar 2.3*Random Forest*

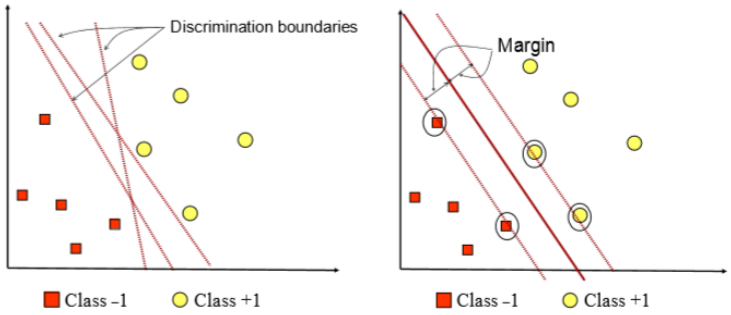
2.4 *Support Vector Machine*

Support Vector Machine (SVM) adalah salah satu metode statistika mutakhir yang sedang berkembang dengan pesat. SVM pertama kali diperkenalkan oleh Vapnik tahun 1992 di *Annual Workshop on Computational Learning Theory.*

SVM adalah salah satu dari beberapa metode yang dikembangkan untuk mengatasi permasalah yang tidak bisa diselesaikan dengan metode statistika klasik, terutama pada kasus klasifikasi dan prediksi. SVM dikembangkan dengan prinsip *linier classifier*. Namun dalam kasus nyata sering dijumpai data yang tidak linier sehingga dikembangkan SVM untuk kasus nonlinier dengan memasukkan konsep kernel. Dengan begitu, ada jaminan bahwa klasifikasi menggunakan SVM akan menghasilkan pemetaaan yang sangat akurat (Hsu, Lin dan Chang, 2003).

2.4.1 *Support Vector Concept*

Konsep dari SVM adalah berusaha menemukan *hyperplane* yang optimum pada *input space*. Fungsi dari *hyperplane* itu digunakan sebagai pemisah dua buah kelas pada *input space*. Kelas sering disimbolkan dengan -1 dan +1. Gambar 2.4 mengilustrasikan *hyperplane*pada metode SVM.*Pattern* pada kelas -1 ditandai dengan warna merah (kotak). Sedangkan *pattern* pada kelas +1, ditandai dengan warna kuning (lingkaran).



(a) (b)

Gambar 2. 4 Ilustrasi Hyperplane pada Metode SVM

Gambar 2.4.(a) menunjukkan alternatif garis pemisah antara dua kelas (*discriminant boundaries*). Garis pemisah yang terbaik adalah yang memiliki *margin hyperplane* maksimum. *Margin*adalah jarak antara *hyperplane* dengan *pattern* terdekat pada masing-masing kelas. *Pattern* yang paling dekat disebut sebagai *support vector*. Pada Gambar 2.4.(b),*pattern* yang dilingkari adalah *support vector* untuk tiap kelas. Sedangkan, garis tebal dalam Gambar 2.4.(b) adalah *hyperplane* terbaik karena berada di tengah-tengah kedua kelas. Proses mencari letak *hyperplane* ini adalah inti dari metode SVM.

2.4.2 *Support Vector Classification*

Data yang tersedia dinotasikan dengan $\vec{x}_i \in \Re^d$. Sedangkan untuk label masing-masing dinotasikan sebagai

$y_i \in \{-1,+1\}, i = 1,2,...,l$, dimana *l* adalah banyaknya data yang digunakan. Misalkan diketahui ***X*** memiliki pola dimana $\mathbf{x}_i$ termasuk kedalam kelas maka $\mathbf{x}_i$ diberi label (target) $y_i = +1$ dan $y_i = -1$. Diasumsikan kelas -1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi *d*, yang didefinisikan sebagai:

$$\vec{w}.\vec{x} + b = 0.$$

Pattern $\mathbf{x}_i$ masuk ke dalam kelas -1 (sampel negatif) jika memenuhi pertidaksamaan berikut:

$$\vec{w}.\vec{x} + b \leq -1.$$

Sedangkan *pattern* $\mathbf{x}_i$ masuk ke dalam kelas +1 (sampel positif) jika memenuhi pertidaksamaan berikut:

$$\vec{w}.\vec{x} + b \geq +1.$$

Margin terbesar bisa diperoleh dengan memaksimumkan nilai jarak antara *hyperplane*dengan titik terdekatnya, yaitu $1/\|\vec{w}\|$. Hal ini dapat dirumuskan sebagai *Quadratic Programming* (QP) *problem*, yaitu mencari titik minimal persamaan (2.9) dengan batasan persamaan (2.10)

$$\min_{\vec{w}} \tau(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 \qquad (2.9)$$

$$y_i[\vec{w}.\vec{x} + b] \geq 0, i = 1,2,...,l. \qquad (2.10)$$

Problem ini dapat diatasi dengan menggunakan teknik komputasi, diantaranya adalah Lagrange Multiplier.

$$L(\vec{w}, b, \alpha) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{l} \alpha_i \{y_i[\vec{w}.\vec{x}_i + b] - 1\} \qquad (2.11)$$

dengan $i=1,2,\ldots,l$.

α_i adalah *Lagrange Multiplier* yang nilainya nol atau positif ($\alpha_i \geq 0$).Persamaan (2.11) dapat dioptimumkan dengan meminimalkan

L terhadap $\vec{w}$ dan *b*, dan memaksimalkan *L* terhadap α_i. dengan menggunakan sifat bahwa pada titik optimum gradien *L* = 0, persamaan (2.11) dapat ditulis sebagai berikut.
Memaksimumkan:

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j \,, \qquad (2.12)$$

dimana, $\alpha_i \geq 0$ $(i=1,2,..l)$ $\sum_{i=1}^{l} \alpha_i y_i = 0.$

Dari persamaan (2.12) akan diperoleh α_l yang kebanyakan bernilai positif. Data yang berkorelasi dengan α_i yang positif inilah yang disebut sebagai ***support vector***.

2.4.3 *Soft Margin*

Teorema yang dijelaskan diatas dikembangkan berdasarkan asumsi bahwa kedua kelas dapat dipisahkan secara sempurna oleh *hyperplane*. Namun, dalam kasus nyata, umumnya kelas tidak dapat dipisahkan secara sempurna. Hal tersebut mengakibatkan batasan pada persamaan (2.8) tidak terpenuhi sehingga optimasi tidak dapat dilakukan. Untuk mengatasi masalah ini, SVM dirumuskan ulang dengan memperkenalkan teknik *soft margin*. Dalam teknik *soft margin*, persamaan (2.8) dimodifikasi dengan memasukkan slack variabel ξ, dengan (ξ>0).

$$y_i \left[\vec{w}.\vec{x}_i + b \right] \geq 1 - \xi_i$$

sehingga persamaan (2.7) menjadi.

$$\min_{\vec{w}} \tau(\vec{w}, \xi) = \frac{1}{2} \left\| \vec{w} \right\|^2 + C \sum_{i=1}^{l} \xi_i \,. \qquad (2.11)$$

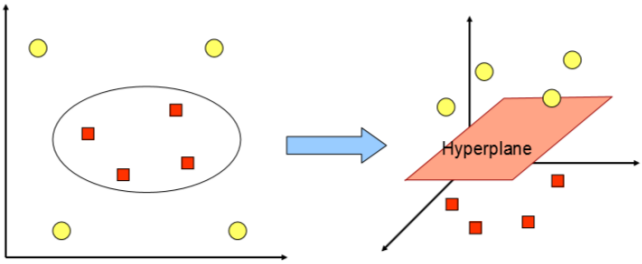
Paramater *C* pada persamaan (2.11) dipilih untuk mengontrol *trade off* antara margin dan error klasifikasi ξ. Nilai

*C*yang besar akan memberikan penalti yang lebih besar terhadap error klasifikasi tersebut.

2.4.4 Fungsi Kernel pada SVM

Pada kasus nyata, sangat jarang dijumpai masalah yang bersifat *linier separable*. Sebagai metode yang dikembangkan untuk kasus linier, SVM membutuhkan sebuah fungsi yang mampu membuat pemisah yang tidak linier. Fungsi yang sering digunakanuntuk mengatasi hal tersebut adalah fungsi kernel.

Pada kasus nonlinier SVM, data $\vec{x}$ terlebih dahulu dipetakan oleh fungsi $\Phi(\vec{x})$ ke dalam ruang vektor yang berdimensi lebih tinggi. Setelah mendapat ruang vektor yang baru, kemudian *hyperplane* bisa dikonstruksikan untuk memisahkan kedua kelas.



(a) (b)

Gambar 2. 5 Fungsi Φ memetakan data ke ruang vektor yang berdimensi lebih tinggi, sehingga kedua kelas dapat dipisahkan secara linear oleh sebuah hyperplane

Gambar 2.5.(a) mengilustrasikancontoh kasus data yang berada di dimensi dua dan tidak dapat dipisahkan dengan menggunakan *hyperplane* yang linier. Kemudian Gambar 2.5.(b) adalah pemetaan data pada *input space* ke dimensi yang lebih tinggi (dimensi tiga) melalui fungsi Φ. Setelah dipetakan ke

dimensi yang lebih tinggi kelas merah dan kelas kuning dapat dipisahkan dengan *hyperplane*. Notasi dari pemetaan ini ditunjukkan oleh persamaan berikut:

$$\Phi : \Re^d \rightarrow \Re^q ; d < q.$$

Pemetaan yang dilakukan tidak mengubah karakteristik atau dalam kata lain tetap menjaga tipologi data. Artinya, dua data yang berjarak dekat pada *input space* akan tetap berjarak dekat juga pada *feature space*. Sebaliknya, dua data yang berjarak jauh pada *input space* akan tetap berjarak jauh pada *feature space*.

Setelah melakukan pemetaan, langkah selanjutnya dalam proses SVM adalah menemukan titik-titik *support vector*. Untuk menemukan titik-titik *support vector*, digunakan *dot product* dari data yang sudah ditransformasi pada ruang yang berdimensi lebih tinggi, yaitu $\Phi(\vec{x}_i).\Phi(\vec{x}_j)$. Akan tetapi, transformasi Φ tidak dapat diketahui dan sangat sulit dipahami sehingga perhitungan *dot product* tersebut dapat diganti dengan fungsi kernel $K(\vec{x}_i, \vec{x}_j)$ yang mendefinisikan secara implisit transformasi Φ. Gunn tahun 1998 merumuskan fungsi kernel sebagai berikut:

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i).\Phi(\vec{x}_j).$$

Dengan menggunakan fungsi kernel proses menentukan *support vector* menjadi lebih mudah karena cukup dengan mengetahui fungsi kernel yang dipakai, tanpa perlu mengetahui wujud dari fungsi nonlinier Φ. Beberapa fungsi kernel yang biasa digunakan ditunjukkan pada Tabel 2.1.

Tabel 2. 1 Fungsi Kernel yang umum pada SVM

Jenis Kernel	**Fungsi**
Polynomial	$K(\vec{x}_i, \vec{x}_j) = \left(\left(\vec{x}_i, \vec{x}_j\right)+1\right)^p$ dimana$p$=1,…
Gaussian Radial Basis Function (RBF)	$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\left\|\vec{x}_i - \vec{x}_j\right\|^2}{2\sigma^2}\right)$
Sigmoid	$K(\vec{x}_i, \vec{x}_j) = \tanh(\alpha.\vec{x}_i\vec{x}_j + \beta)$

Selanjutnya hasil klasifikasi dari data $\vec{x}$ diperoleh dari persamaan berikut :

$$f[\Phi(\vec{x})] = \vec{w}\,\Phi(\vec{x}) + \vec{b}$$

$$f[\Phi(\vec{x})] = \sum_{i=1,SVs} \alpha_i y_i \Phi(\vec{x}).\Phi(\vec{x}_i) + \vec{b}$$

$$= \sum_{i=1,SVs} \alpha_i y_i \; K(\vec{x}_i, \vec{x}_j) + \vec{b}\,.$$

Support vector pada persamaan di atas adalah subset dari training set yang terpilih sebagai *support vector*, dengan kata lain data $\vec{x}_i$ yang berkorespondensi pada $\alpha_i \geq 0$.

2.5 Ukuran Ketepatan Klasifikasi

Ketepatan klasifikasi digunakan untuk menilai seberapa baik sebuah model yang diperoleh mampu merepresentasikan kejadian yang sesungguhnya. Kesalahan klasifikasi bisa diukur melalui *Apparent Error Rate* (APER) dan juga *total accuracy rate* (1-APER) untuk mengukur ketepatan klasifikasinya. Tabel 2.2 di bawah ini menunjukkan perhitungan nilai APER.

Tabel 2. 2*Cross Tabulation* Hasil Klasifikasi

Observasi	Prediksi		Total
	1	2	
1	n_{11}	n_{12}	$n_{1.}$
2	n_{21}	n_{22}	$n_{2.}$
Total	$n_{.1}$	$n_{.2}$	N

Keterangan :

n_{11}: jumlah anggota kelas 1 yang diprediksi benar

n_{12}: jumlah anggota kelas 1 yang diprediksi salah

n_{21}: jumlah anggota kelas 2 yang diprediksi salah

n_{22} : jumlah anggota kelas 2 yang diprediksi benar

N : jumlah pengamatan total

sehingga nilai *APER* nya adalah

$$APER = \frac{n_{12} + n_{21}}{N} \times 100\%$$

dan akurasi totalnya adalah

total accuracy rate = 1- *APER*

2.6 Epilepi

Epilepsi adalah salah jenis penyakit yang menyerang syaraf. Epilepsi didefinisikan sebagai kumpulan gejala dan tanda-tanda klinis yang muncul disebabkan oleh gangguan fungsi otak secara intermiten. Gejala tersebut terjadi akibat terlepasnya muatan listrik abnormal atau berlebihan dari neuron-neuron secara paroksismal dengan berbagai macam etiologi. Sedangkan serangan atau bangkitan epilepsi yang dikenal dengan istilah *epileptic seizure* adalah manifestasi klinis yang serupa dan berulang secara paroksismal yang diakibatkan oleh hiperaktivitas listrik sekelompok sel saraf di otak yang spontan dan bukan disebabkan oleh suatu penyakit otak akut (Shorvon, 2000). Manifestasi serangan atau bangkitan epilepsi secara klinis dapat

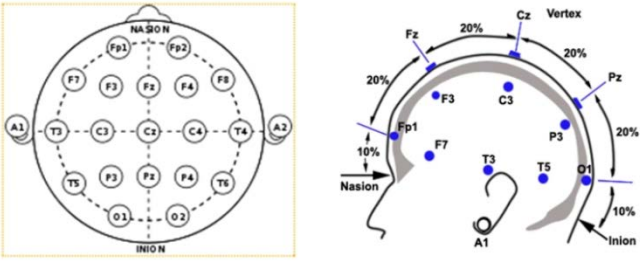
dicirikan sebagai gejala yang timbul secara mendadak, hilang spontan dan cenderung berulang.

2.7 Electroencephalograph

EEG adalah rekaman aktivitas listrik neuron otak. Fluktuasi arus listrik tersebut didapatkan dari perbedaan voltase yang diukur dari elektrode yang ditempel di kulit kepala (*skalp*), langsung dipermukaan kortek serebri atau di dalam jaringan otak (Heinemann, 1994). Terkait dengan letak elektroda di kepala, secara umum terdapat dua jenis pemeriksaan EEG yaitu pemeriksaan dengan elektrode yang ditempel di kepala (*skalp*) dan pemeriksaan yang elektrodenya dimasukkan dalam intra kranal (EEG kortikal dan intra kranial).

Aktivitas EEG ditunjukkan dengan dengan ukuran Hertz untuk satuan frekuensi, milisecond untuk durasi serta microvolt untuk amplitudo gelombang. Agar gelombang yang dihasilkan terbebas dari artefak dan hanya merekam gelombang listrik otak saja maka selama proses rekaman dibutuhkan filter yang bisa mengontrol gelombang berfrekuensi tinggi (*high filter*), frekuensi rendah (*low filter*) dan frekuensi aliran listrik rumah (*notch*). Rentang voltase untuk sinyal EEG berkisar antara 3-100 μV atau 100 kali lebih lemah dibandingkan sinyal *Electrocardiograph* (ECG).

Pada pemeriksaan EEG skalp pemasangan elektrode menggunakan kaidah 10–20 system yang di dasarkan pada rekomendasi komite federasi elektroensefalografi dan neurofisiologi klinik internasional (IFSECN) (Jesper, 1958). Teknik penempatan elektrode ini pada tahun 1991 telah dikembangkan dengan identifikasi dan lokasi spesifik untuk 75 posisi elektrode di skalp yang disebut 10–10 system (Nuwer, 1987) dan (Pernier, Perrin dan Bertrand , 1988). Penambahan titik yang lebih rapat ini bertujuan untuk mendapatkan lokasi yang lebih akurat dalam menentukan fokus epileptogenik. Bintoro (2012) mengilustrasikan pemasangan elektrode untuk pemeriksaan EEG seperti pada Gambar 2.6 berikut:



Gambar 2.6 Penempatan Elektrode Skalp 10-20 S*ystem*

(*Halaman ini sengaja dikosongkan*)

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

Data yang digunakan dalam penelitian kali ini diperoleh dari *laman* (http://ntsa.upf.edu/downloads/andrzejak-rg-et-al-2001-indications-nonlinear-deterministic-and-finite-dimensional). Data tersebut diperkenalkan pertama kali oleh Andrzejak *et al*. (2001). Kemudian data digunakan lagi oleh Guo et al. (2010) dengan makalah berjudul *Automatic Epileptic Seizure Detection in EEG based on Line Length Feature and Artificial Neural Network* yang diterbitkan tahun 2010 pada *Journal of Neuroscience Methods*. Data terdiri dari lima set data, yaitu Z, O, N , F dan S. masing-masing set data terdiri dari 100 *single-channel* EEG dengan durasi 23,6 detik dan laju sampling 173,6 Hz. Data yang digunakan telah dipilih dan dipotong dari tipe *continous multi-channel* EEG dan telah melalui proses penghilangan artefak seperti gerakan otot atau kedipan mata.

3.2 VariabelPenelitian

Variabel yang digunakan dalam penelitian ini adalah sinyal EEG yang telah melewati proses DWT. Jumlah variable predictor ditentukan oleh banyaknya level yang digunakan pada prosedur *subband coding.* Level dipilih sedemikian rupa hingga bagian-bagian dari sinyal yang berkorelasi dengan frekuensi yang diperlukan untuk klasifikasi bias terwakilkan dalam koefisien *wavelet*. Pada penelitian ini, level dekomposisi yang digunakan adalah empat level, enam level dan delapan level. Kemudian, masing-masing data yang diperoleh dari tiga kombinasi level tersebut masing-masing akan diklasifikasi menggunakan metode *Random Forest* dan SVM. Pada tahun 2007, Subasi dalam penelitiannya yang lain membuktikan bahwa Daubechies order empat atau (db4) mempunyai *smoothing feature* yang sesuai untuk mendeteksi perubahan pada EEG sehingga fungsi *wavelet* yang dipilih dalam penelitian ini adalah Daubechies order empat

(db4). Pada DWT dengan dekomposisi empat level, satu sinyal EEG terdekomposisi menjadi empat sub-sinyal (*detail*) yakni D1 sampai D4, ditambah satu sub-sinyal *final approximation* (A4). Pada DWT dengan enam level, satu sinyal EEG terdekomposisi menjadi enam sub-sinyal (*detail*) yakni D1 sampai D6, ditambah satu sub-sinyal *final approximation* (A6). Pada DWT dengan delapan level, satu sinyal EEG terdekomposisi menjadi enam sub-sinyal (*detail*) yakni D1 sampai D8, ditambah satu sub-sinyal *final approximation* (A8). Masing-masing sub-sinyal memuat informasi untuk pita frekuensi yang berbeda.

3.3 LangkahAnalisis

Langkah-langkah yang dilakukan dalam penelitian ini adalah:

Langkah 1.*Preprocessing*

Proses *preprocessingn* dilakukan dengan metode DWT dan LLF. DWT dilakukan untuk mendapatkan data yang telah terdekomposisi. Level yang digunakan dalam proses DWT adalah sebanyak empat level, enam level dan delapan level. Data hasil dekomposisi yang masih berbentuk sinyal diekstraksi dengan menggunakan LLF. Hasil LLF tersebut digunakan sebagai input untuk proses klasifikasi.

Langkah 2. Melakukan Pemisahan Data *Training* dan *Testing*

Setelah proses *preprocessing*, maka akan diperoleh tiga dataset. Masing-masing dataset yang diperoleh kemudian dibagi menjadi dua yakni data *training* dan data *testing*. Data *training* dan *testing* dipilih secara random dengan cara *resampling.* Kombinasi perbandingan data *training* dan *testing* yang digunakan dalam penelitian ini adalah sebesar 75%:25%, 80%:20%, 85%:15%, 90:10% dan 95%:5%.

Langkah 3. Melakukan analisis *Random Forest*

Frekuensi yang sudah terpisahkan melalui proses *subband coding* kemudian digunakan untuk klasifikasi dengan menggunakan *Random Forest*. Analisis klasifikasi *random forests* melalui tahapan berikut:

tahap a. Mengambil *n* sampel *bootstrap* dengan pengembalian dari data *training*

tahap b. Menentukan jumlah variabel prediktor yang akan dilakukan pengambilan secara acak dalam proses penentuan pemilah saat pembentukan pohon klasifikasi

$$\log_2(M+1) \text{ atau } \sqrt{p}$$

tahap c. Membentuk pohon klasifikasi dimana pemilihan *node* terbaik dilakukan berdasarkan variabel-variabel prediktor yang diambil secara acak

tahap d. Melakukan prediksi klasifikasi untuk data *training*

tahap e. Mengulangi tahap b sampai tahap d hingga *K* kali replikasi

tahap f. Melakukan voting mayoritas (*majority vote*) hasil prediksi klasifikasi dari *K* kali replikasi pembentukan pohon klasifikasi

tahap g. Menghitung ketepatan klasifikasi data *training*

tahap h. Menghitung ketepatan klasifikasi data *testing*

tahap i. Mengulangi tahap b sampai tahap h dengan mencobakan kombinasi jumlah pohon (*K*) yang berbeda, yaitu sebesar 100, 500 dan 1000.

tahap j. Memilih kombinasi jumlah pohon yang memiliki keakurasian tertinggi.

Langkah 4. Membuat model klasifikasi penyakit epilepsi menggunakan *Support Vector Machine* (SVM) dengan tahapan analisis sebagaiberikut:

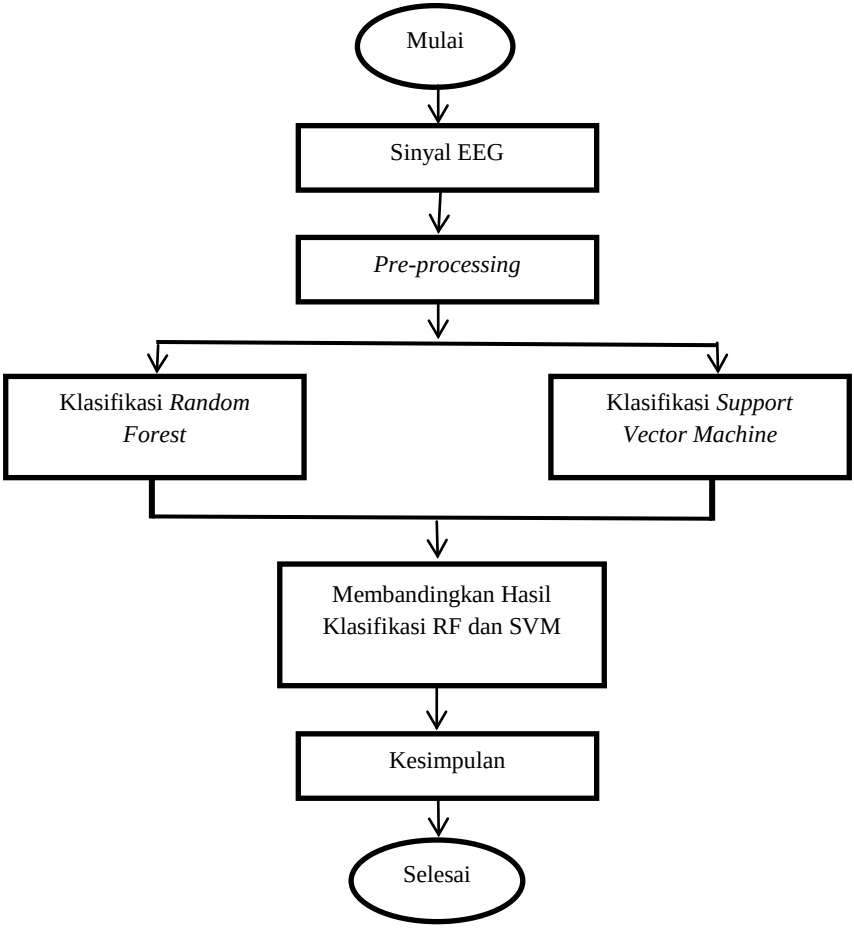
tahap a. Menggunakan *Gaussian Radial Basis Function* (RBF).

tahap b. Menentukan parameter *C* dan γ.

tahap c. Menghitung klasifikasi beserta ketepatan akurasinya.

Langkah 5. Membandingkan performansi ketepatan klasifikasi antara metode *Random Forest* dan *Support Vector Machine*.

Langkah-langkah analisis dalam penelitian ini diilustrasikan dengan diagram alir pada Gambar 3.1.



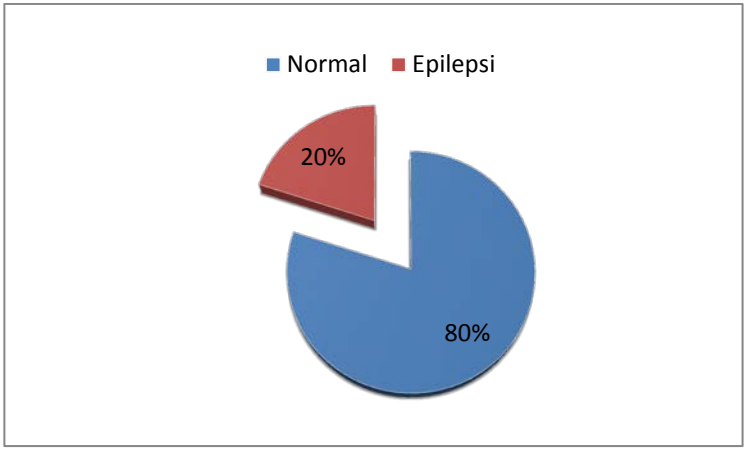
Gambar 3 1 Diagram Alir Peneltian

(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS DAN PEMBAHASAN

4.1 *Preprocessing*

Sebelum melakukan analisis klasifikasi, hal pertama yang dilakukan adalah melakukan prosedur prapemrosesan (*preprocessing*) data. *Preprocessing* yang dilakukan pada penelitian ini adalah *Discrete Wavelet Transform* (DWT) dan ekstraksi *Line Length Feature* (LLF). Data yang digunakan adalah data hasil rekaman EEG pada 500 orang yang terdiri dari pasien normal dan pasien yang terdiagnosis menderita epilepsi.



Gambar 4. 1 Presentase Kategori Data

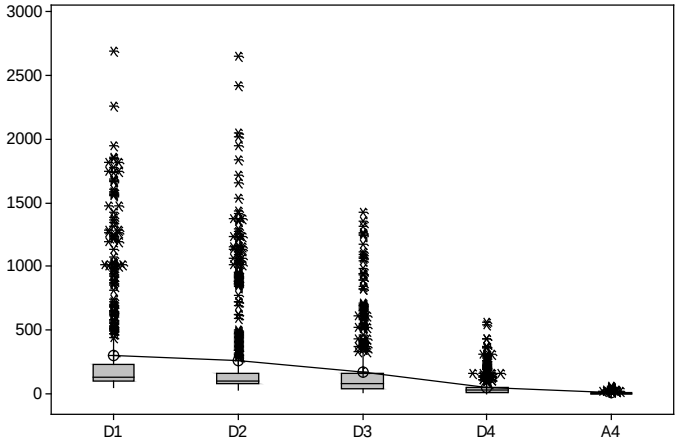
Gambar 4.1 memperlihatkan presentase dari data yang digunakan dalam penelitian ini. Sebanyak 80% atau 400 pengamatan adalah pasien normal atau tidak menderita epilepsi. Jumlah 400 pengamatan tersebut terdiri dari empat *dataset* yakni Z, N, O dan F dimana masing-masing *dataset* terdiri dari 100 pengamatan. Keempat *dataset* tersebut diamati dari pasien yang tidak menderita epilepsi tetapi dengan kondisi pengamatan yang berbeda. Sedangkan 20% atau 100 pengamatan tersisa adalah pasien yang menderita epilepsi yang terdiri dari *dataset* S yang merupakan penderita epilepsi..

4.1.1 *Discrete Wavelet Transform*

Data hasil rekaman EEG yang terdiri dari 500 sampel pengamatan diproses dengan menggunakan DWT untuk mendapatkan sub-sinyal yang telah terdekomposisi dan akan digunakan sebagai *input* pada tahap klasifikasi. Fungsi *wavelet* yang digunakan adalah Daubechies order empat (db4). Sedangkan level dekomposisi yang digunakan ada tiga level, yaitu empat level, enam level dan delapan level.

4.1.2 *Line Length Feature Extraction*

Setelah proses DWT, tahap *preprocessing* selanjutnya adalah ekstraksi sub-sinyal yang diperoleh dari proses DWT dengan menggunakan metode LLF. LLF menghitung masing-masing sub-sinyal kedalam bentuk vektor berukuran 1 x (k+1), dengan k adalah benyaknya level dekomposisi yang digunakan. Pada penelitian ini digunakan tiga macam level dekomposisi, yakni empat level, enam level dan delapan level. Secara teori, masing-masing sub-sinyal yang dihasilkan melalui proses DWT dan LLF mewakili informasi yang ada di tiap level sehingga karkteristiknya berbeda satu sama lainnya. Gambar 4.2, Gambar 4.3 dan Gambar 4.4 di bawah ini menunjukkan karakteristik dari sub-sinyal atau variabel yang dihasilkan pada ketiga macam level dekomposisi yang digunakan.

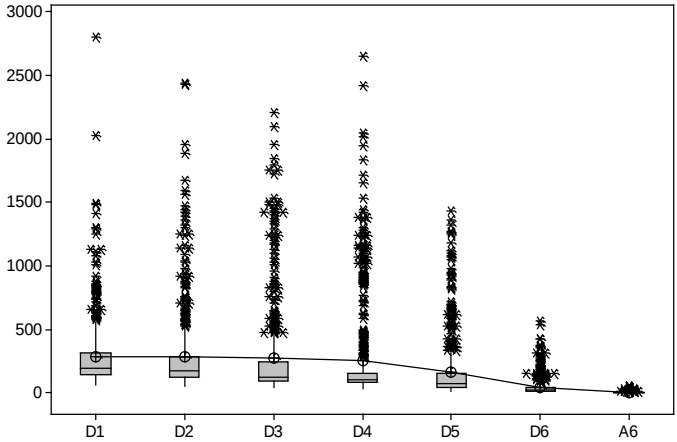


Gambar 4. 2 Bloxpot untuk *Output* Empat Level Dekomposisi

Gambar 4.1 memperlihatkan hasil dari *preprocessing* menggunakan DWT untuk dekomposisi dan LLF untuk ekstraksi. Variabel D1 adalah *output* di level satu, variabel D2 adalah *output* di level dua, variabel D3 adalah *output* di level tiga serta variabel D4 dan variabel A4 adalah *output* di level terakhir yakni level empat.

Nilai rata-rata, nilai minimum dan nilai maksimum dari kelima variabel tampak semakin kecil, nilai standar deviasi dari kelima variabel juga relatif menurun, serta jangkauan dari kelima variabel semakin besar levelnya semakin kecil.

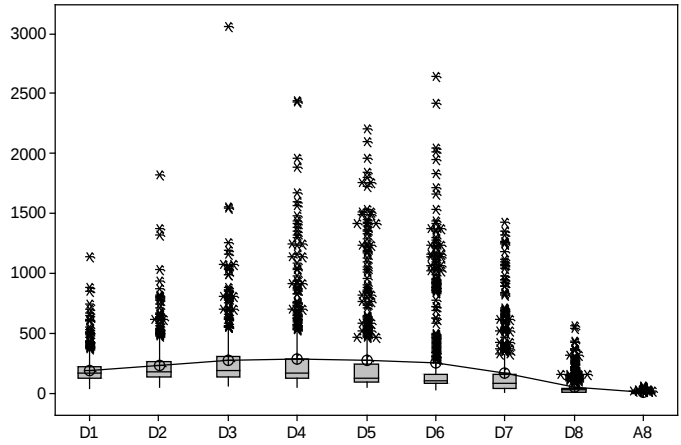
Uraian diatas menunjukkan bahwa kelima level variabel atau sub-sinyal yang dihasilkan jika dilihat dari karakteristik nilai rata-rata, nilai minimum, nilai maksimum, standar deviasi dan jangkauannya memiliki karakteristik yang berbeda satu sama lainnya. Selanjutnya adalah karakteristik dari sub-sinyal pada enam level dekomposisi.



Gambar 4. 3 Bloxpot untuk *Output* Enam Level Dekomposisi

Gambar 4.2 memperlihatkan hasil dari *preprocessing* menggunakan DWT untuk dekomposisi dan LLF untuk ekstraksi. Variabel D1 hingga D5 berturut-turut adalah *output* di level satu hingga level lima. Sedangkan variabel D6 dan A6 adalah *output* di level enam.

Nilai rata-rata dan nilai minimum dari ketujuh variabel tampak semakin kecil, tetapi tidak untuk nilai maksimumnya karena sempat naik di level empat kemudian turun lagi di level lima dan seterusnya. Nilai standar deviasi dan jangkauan data dari ketujuh variabel tampak naik turun. Uraian diatas menunjukkan bahwa ketujuh level variabel atau sub-sinyal yang dihasilkan jika dilihat dari karakteristik nilai rata-rata, nilai minimum, nilai maksimum, standar deviasi dan jangkauannya tidak memiliki karakteristik yang berbeda drastis satu sama lainnya. Selanjutnya adalah karakteristik dari sub-sinyal pada delapan level dekomposisi.



Gambar 4. 4 Bloxpot untuk *Output* Delapan Level Dekomposisi

Gambar 4.3 memperlihatkan hasil dari *preprocessing* menggunakan DWT untuk dekomposisi dan LLF untuk ekstraksi. Variabel D1 hingga D7 berturut-turut adalah *output* di level satu hingga level tujuh. Sedangkan variabel D8 dan A8 adalah *output* di level delapan.

Nilai rata-rata, nilai minimum dan nilai maksimum dari kesembilan variabel tampak naik dan kemudian menurun lagi. Nilai standar deviasi dan jangkauan data dari keenam variabel tampak naik turun. Uraian diatas menunjukkan bahwa kesembilan level variabel atau sub-sinyal yang dihasilkan jika dilihat dari karakteristik nilai rata-rata, nilai minimum, nilai maksimum, standar deviasi dan jangkauannya tidak memiliki karakteristik unik yang membedakan satu dengan lainnya.

4.2. Klasifikasi Menggunakan *Random Forest Classification*

Klasifikasi *random forest* dilakukan pada tiga dataset yang dihasilkan dari proses *preprocessing* yakni dataset empat level dekomposisi, dataset enam level dekomposisi dan dataset delapan level dekomposisi. Masing-masing dataset akan dibagi menjadi beberapa kombinasi data *training* dan data *testing,* yakni

75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:5%. Tiap-tiap dataset juga akan diklasifikasikan dengan *K* sebanyak 100, 500 dan 1000.

4.2.1 Klasifikasi dengan *Random Forest* pada Dataset Empat Level Dekomposisi

Empat level dekomposisi menghasilkan dataset dengan jumlah variabel prediktor sebanyak lima, yaitu D1, D2, D3, D4 dan A4. Dataset tersebut kemudian diklasifikasikan dengan metode *random forest* dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10% dengan jumlah pohon *K* sebanyak 100, 500 dan 1000.

Langkah pertama yang dilakukan dalam analisis *random forest* adalah menentukan jumlah variabel prediktor yang akan diambil secara acak dalam pemilihan pemilah saat pembentukan pohon klasifikasi. Jumlah variabel prediktor yang dipilih adalah $\sqrt{p}$ dengan *p* adalah banyaknya variabel prediktor. Jumlah *p* pada dataset empat level dekomposisi adalah 5 sehingga jumlah variabel prediktor yang dipilih sebagai kontrol adalah $\sqrt{5}$ = 2,24 dan dibulatkan menjadi 2 variabel. Jumlah pohon yang akan dibentuk adalah 100, 500 dan 1000.

Tabel 4. 1 Perbandingan Ketepatan Klasifikasi *Random Forests* pada Beberapa Kombinasi Data dan Jumlah Pohon untuk Dataset Empat Level Dekomposisi

Kombinasi data *Training* dan data *Testing*	**Ketepatan Klasifikasi (1-*APER*) (dalam %)**					
	***K*=100**		***K*=500**		***K*=1000**	
	Data *Training*	**Data *Testing***	**Data *Training***	**Data *Testing***	**Data *Training***	**Data *Testing***
75%:25%	100	100	100	100	100	100
80%:20%	100	95,83	100	95,83	100	95,83
85%:15%	100	100	100	100	100	100
90%:10%	100	97,96	100	95,83	100	95,83
95%:5%	100	100	100	100	100	100

Tabel 4.1 menunjukkan bahwa ketepatan klasifikasi yang diperoleh sangat baik. Kombinasi data *training* dan data *testing* 75%:25%, 85%:15% dan 95%:5% secara konsisten memiliki ketepatan klasifikasi 100% baik untuk data *training* maupun data *testing*. Hal tersebut terjadi pada semua kombinasi jumlah pohon (*K*) yang ada sehingga disimpulkan bahwa level jumlah pohon (*K*) tidak memberikan perbedaan yang berarti terhadap ketepatan klasifikasi. Kombinasi data *training* dan data *testing* sebesar 75%:25%, 85%:15% dan 95%:5% memberikan nilai ketepatan klasifikasi terbesar yakni sebesar 100% baik untuk data *training* maupun data *testing*. Kombinasi data *training* dan data *testing* 80%:20% memiliki ketepatan klasifikasi paling rendah yaitu sebesar 100% untuk data *training* dan 95,83% untuk data *tetsing*.

4.2.2 Klasifikasi dengan *Random Forest* pada Dataset Enam Level Dekomposisi

Proses selanjutnya adalah analisis pada dataset enam level dekomposisi. Dataset enam level dekomposisi menghasilkan memiliki variabel prediktor sebanyak tujuh, yaitu D1, D2, D3, D4, D5, D6 dan A6. Dataset tersebut kemudian diklasifikasikan dengan metode *random forest* dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10% dan jumlah pohon (*K*) sebanyak 100, 500 dan 1000.

Seperti langkah sebelumnya, dilakukan pemilihan variabel prediktor untuk diambil secara acak sejumlah $\sqrt{p}$ dengan *p* adalah banyaknya variabel prediktor. Jumlah *p* pada dataset empat level dekomposisi adalah 7 sehingga jumlah variabel prediktor yang dipilih sebagai kontrol adalah $\sqrt{7}$ = 2,64 dan dibulatkan menjadi 3 variabel. Jumlah pohon (*K*) yang akan dibentuk adalah 100, 500 dan 1000.

Tabel 4. 2 Perbandingan Ketepatan Klasifikasi *Random Forests* pada Beberapa Kombinasi Data dan Jumlah Pohon untuk Dataset Enam Level Dekomposisi

Kombinasi data *Training* dan data *Testing*	**Ketepatan Klasifikasi (1-*APER*) (dalam %)**					
	***K*=100**		***K*=500**		***K*=1000**	
	Data *Training*	**Data *Testing***	**Data *Training***	**Data *Testing***	**Data *Training***	**Data *Testing***
75%:25%	100	99,19	100	99,19	100	99,19
80%:20%	100	97,96	100	97,96	100	97,96
85%:15%	100	97,26	100	97,26	100	97,26
90%:10%	100	97,96	100	97,96	100	97,96
95%:5%	100	100	100	100	100	100

Tabel 4.2 memperlihatkan bahwa ketepatan klasifikasi untuk semua kombinasi data *training* dan data *testing* sama persis pada semua level jumlah pohon. Hal tersebut dapat disimpulkan bahwa level jumlah pohon (*K*) tidak berpengaruh sama sekali terhadap ketepatan klasifikasi pada dataset enam level dekomposisi.

Nilai ketepatan klasifikasi tertinggi untuk data *training* adalah sebesar 100%. Hal tersebut terjadi untuk semua kombinasi data *training* dan data *tetsing* di semua level jumlah pohon (*K*). Sedangkan, nilai tertinggi untuk data *testing* diperoleh pada data *training* 5% pada semua level jumlah pohon (*K*) dengan nilai ketepatan klasifikasi 100%. Nilai ketepatan klasifikasi terkecil dari data *testing* adalah sebesar 97,96% yang diperoleh dari data *testing* 10%, 15% dan 20% yang terjadi di semua level jumlah pohon (*K*).

Kombinasi ketepatan klasifikasi paling tinggi diperoleh dari kombinasi data *training* dan data *testing* 95%:5% dengan nilai ketepatan data *training* dan data *testing* masing-masing sebesar 100%. Sedangkan kombinasi ketepatan klasifikasi terkecil diperoleh dari kombinasi data *training* dan data *testing* 85%:15%

dengan nilai ketepatan data *training* dan data *testing* berturut-turut sebesar 100% dan 97,26%.

4.2.3 Klasifikasi dengan *Random Forest* pada Dataset Delapan Level Dekomposisi

Delapan level dekomposisi menghasilkan dataset dengan jumlah vriabel prediktor sebanyak sembilan, yaitu D1, D2, D3, D4, D5, D6, D7, D8 dan A8. Dataset tersebut kemudian diklasifikasikan dengan metode *random forest* dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10% dan jumlah pohon (*K*) sebanyak 100, 500 dan 1000.

Jumlah variabel prediktor yang dipilih secara acak adalah $\sqrt{9}$ sehingga jumlah variabel prediktor yang dipilih sebagai kontrol adalah $\sqrt{9}$ = 3 variabel. Jumlah pohon yang akan dibentuk adalah 100, 500 dan 1000.

Tabel 4. 3 Perbandingan Ketepatan Klasifikasi *Random Forests* pada Beberapa Kombinasi Data dan Jumlah Pohon untuk Dataset Delapan Level Dekomposisi

Kombinasi data *Training* dan data *Testing*	**Ketepatan Klasifikasi (1-*APER*) (dalam %)**					
	***K*=100**		***K*=500**		***K*=1000**	
	Data *Training*	**Data *Testing***	**Data *Training***	**Data *Testing***	**Data *Training***	**Data *Testing***
75%:25%	100	98,37	100	98,37	100	98,37
80%:20%	100	95,83	100	96,91	100	95,83
85%:15%	100	98,65	100	98,65	100	98,65
90%:10%	100	100	100	100	100	100
95%:5%	100	100	100	100	100	100

Tabel 4.3 memperlihatkan bahwa ketepatan klasifikasi untuk semua kombinasi data *training* dan data *testing* hampir sama pada semua level jumlah pohon, perbedaan hanya terjadi pada ketepatan klasifikasi data *testing* 20%. Hal tersebut dapat disimpulkan bahwa level jumlah pohon (*K*) tidak memberikan

perbedaan yang berarti pada hasil ketepatan klasifikasi untuk dataset delapan level dekomposisi.

Untuk data *training* ketepatan klasifikasi tertinggi adalah sebesar 100%. Hal tersebut terjadi untuk semua kombinasi data *training* dan di semua level jumlah pohon (*K*). Sedangkan, nilai tertinggi untuk data *testing* diperoleh pada data *training* 10% dan 5% pada semua level jumlah pohon (*K*) dengan nilai ketepatan klasifikasi 100%. Nilai ketepatan klasifikasi terkecil dari data *testing* adalah sebesar 95,83% yang diperoleh dari data *testing* 20% dengan level jumlah pohon 100 dan 1000.

Kombinasi ketepatan klasifikasi paling tinggi diperoleh dari kombinasi data *training* dan data *testing* 95%:5% dan 90%:10% dengan nilai ketepatan data *training* dan data *testing* masing-masing sebesar 100%. Sedangkan kombinasi ketepatan klasifikasi terkecil diperoleh dari kombinasi data *training* dan data *testing* 80%:20% dengan nilai ketepatan data *training* dan data *testing* berturut-turut sebesar 100% dan 95,83%.

4.2.4 Perbandingan Hasil Ketepatan Klasifikasi dengan Metode *Random Forest* untuk Ketiga Level Dekomposisi

Penelitian ini menggunakan tiga level dekomposisi yang berbeda, yaitu empat level, enam level dan delapan level. Setiap level diklasifikasikan dengan *random forest* dan melalui prosedur yang sama. Oleh karena itu, perlu diketahui apakah level dekomposisi yang digunakan saat proses DWT berpengaruh terhadap ketepatan klasifikasi yang dihasilkan.

Tabel 4. 4 Perbandingan Rata-rata Ketepatan Kalsifikasi untuk Ketiga Level Dekomposisi

	Rata-rata Ketepatan Klasifikasi (1-*APER*) (dalam %)		
Data	**4 Level**	**6 Level**	**8 Level**
Data Training	100	100	100
Data Testing	98,47	98,47	98,64

Tabel 4.4 menunjukkan bahwa untuk data *training* semua level dekomposisi memiliki nilai ketepatan klasifikasi yang sama yakni 100%. Sedangkan untuk data *testing* ketiga level memiliki nilai yang hampir sama. Oleh karena itu, disimpulkan bahwa ketiga level dekomposisi yang digunakan memberikan hasil ketepatan klasifikasi yang relatif sama satu dengan yang lainnya .

4.3 Klasifikasi Menggunakan *Support Vector Machine*

Pada kasus klasifikasi menggunakan metode SVM pada dataset yang dihasilkan dari tiga macam level dekomposisi, yakni empat level, enam level dan delapan level. Ada beberapa langkah yang harus dilakukan yakni pemilihan fungsi Kernel yang digunakan. Pada penelitian ini, fungsi Kernel yang digunakan adalah Gaussian. Langkah selanjutnya adalah menetukan parameter C dan γ dari fungsi Kernel untuk membentuk model. Langkah terakhir adalah klasifikasi dengan model yang telah terbentuk. Klasifikasi dilakukan pada data *training* dan data *testing* yang dibagi menjadi beberapa kombinasi yakni 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:5%. Hasil klasifikasi masing-masing kombinasi data *training* dan *testing* tersebut kemudian dibandingkan dan dipilih kombinasi yang menghasilkan akurasi terbaik.

4.3.1 Klasifikasi dengan *Support Vector Machine* pada Dataset Empat Level Dekomposisi

Empat level dekomposisi menghasilkan dataset dengan jumlah variabel prediktor sebanyak lima, yaitu D!, D2, D3, D4 dan A5. Dataset tersebut kemudian akan diklasifikasikan dengan metode SVM dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10%.

Langkah pertama yang dilakukan pada metode SVM adalah menetukan parameter C dan γ terbaik untuk fungsi Kernel. Pemilihan parameter C dan γ akan berpengaruh terhadap akurasi model yang terbentuk. Untuk itu, perlu dilakukan pemilihan

kombinasi parameter *C* dan γ yang memiliki nilai *error* terkecil sehingga model yang dihasilkan memiliki akurasi yang baik.

Ada beberapa cara yang bisa digunakan untuk menentukan kombinasi parameter *C* dan γ yang paling optimum seperti metode *uniform design* atau menggunakan prosedur *tuning* di *software* R. pada penelitian ini, pemilihan kombinasi parameter *C* dan γ dilakukan dengan prosedur *tune* dengan *software* R.

Saat menentukan kombinasi parameter *C* dan γ, terlebih dahulu harus dideskripsikan *range* dari kedua parameter untuk dikombinasikan. Parameter *C* yang paling tepat untuk SVM adalah antara 10^{-2} hingga 10^{4} (Huang et all., 2007). Pada penelitian ini juga digunakan *range* 10^{-2} hingga 10^{4} untuk parameter *C.* Kemudian untuk parameter γ, Huang dkk tahun 2007 merekomendasikan nilai antara $\frac{10^{-3}}{\rho}$ hingga $\frac{1{,}9}{\rho}$ dengan asumsi nilai ρ adalah 0,5 maka diperoleh nilai antara 0,02 hingga 3,8.

Hasil prosedur *tune* di *software* R menunjukkan bahwa parameter paling optimum untuk *C* dan γ berturut-turut adalah 91 dan 0,302 dengan nilai *error* sebesar 0,008. Nilai parameter *C* dan γ yang diperoleh kemudian digunakan untuk membuat model SVM.

Setelah diperoleh nilai parameter *C* dan γ yang paling optimum, maka selanjutnya nilai tersebut digunakan untuk membentuk model dan kemudian dihitung ketepatan akurasinya. Tabel 4.5 menunjukkan hasil rangkuman ketepatan klasifikasi dari model yang terbentuk.

Tabel 4. 5 Perbandingan Ketepatan Klasifikasi SVM pada Beberapa Kombinasi Data untuk Dataset Empat Level Dekomposisi

Kombinasi Data *Training* dan Data *Testing*	**Akurasi (1-*APER*) (dalam %)**	
	Data *Training*	**Data *Testing***
75%:25%	99,73	99,19
80%:20%	99,75	96,90
85%:25%	99,53	100
90%:10%	99,55	97,96
95%:5%	99,36	100

Tabel 4.5 memperlihatkan perbandingan ketepatan klasifikasi dari beberapa kombinasi data untuk dataset empat level dekomposisi. Untuk data *training* akurasi terbesar adalah pada data *training* 80% dengan nilai ketepatan klasifikasi sebesar 99,75%. Sedangkan nilai ketepatan klasifikasi paling rendah adalah dari data *training* 95% dengan nilai ketepatan klasifikasi sebesar 99,36%. Ketepatan klasifikasi tertinggi pada data *testing* adalah dari data *testing* 25% dan 5% dengan ketepatan klasifikasi sebesar 100%. Sedangkan ketepatan klasifikasi paling rendah untuk *testing* adalah pada data *testing* 20%. Kombinasi data *training* dan data *testing* yang memberikan nilai akurasi terbaik adalah data *training* 85% dan data *testing* 15% dengan nilai ketepatan klasifikasi berturut-turut adalah 99,53% dan 100%.

4.3.2 Klasifikasi dengan *Support Vector Machine* pada Dataset Enam Level Dekomposisi

Enam level dekomposisi menghasilkan dataset dengan jumlah variabel prediktor sebanyak tujuh, yaitu D1, D2, D3, D4, D5, D6 dan A6. Dataset tersebut kemudian akan diklasifikasikan dengan metode SVM dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10%.

Langkah pertama yang dilakukan pada metode SVM adalah proses tuning untuk menetukan parameter C dan γ terbaik untuk fungsi Kernel melalui prosedur *tuning* dengan *software* R.

parameter C dan γ optimum yang diperoleh untuk C adalah 20 dan γ sebesar 0,011.

Setelah diperoleh nilai parameter C dan γ yang paling optimum, maka selanjutnya nilai tersebut digunakan untuk membentuk model dan kemudian dihitung ketepatan akurasinya. Tabel 4.6 menunjukkan hasil rangkuman ketepatan klasifikasi dari model yang terbentuk.

Tabel 4. 6 Perbandingan Ketepatan Klasifikasi SVM pada Beberapa Kombinasi Data untuk Dataset Enam Level Dekomposisi

Kombinasi Data *Training* dan Data *Testing*	**Akurasi (1- *APER*) (dalam %)**	
	Data *Training*	**Data *Testing***
75%:25%	98,37	97,54
80%:20%	98,73	97,96
85%:15%	98,81	95,83
90%:10%	98,65	95,83
95%:5%	98,50	100

Tabel 6 memperlihatkan perbandingan ketepatan klasifikasi dari beberapa kombinasi data untuk dataset enam level dekomposisi. Untuk data *training* akurasi terbesar adalah pada data *training* 85% dengan nilai ketepatan klasifikasi masing-masing sebesar 98,81%. Sedangkan nilai ketepatan klasifikasi paling rendah adalah dari data *training* 75% dengan nilai ketepatan klasifikasi sebesar 98,37%. Ketepatan klasifikasi tertinggi pada data *testing* adalah dari data *testing* 5% dengan ketepatan klasifikasi sebesar 100%. Sedangkan ketepatan klasifikasi paling rendah untuk *testing* adalah pada data *testing* 10% dan 10% dengan nilai Ketepatan klasifikasi sebesar 95,83%. Kombinasi data *training* dan data *testing* yang memberikan nilai akurasi terbaik adalah data *training* 95% dan data *testing* 5% dengan nilai ketepatan klasifikasi berturut-turut adalah 98,50% dan 100%.

4.3.3 Klasifikasi dengan *Support Vector Machine* pada Dataset Delapan Level Dekomposisi

Delapan level dekomposisi menghasilkan dataset dengan jumlah variabel prediktor sebanyak sembilan, yaitu D1, D2, D3, D4, D5, D6, D7, D8 dan A8. Dataset tersebut kemudian akan diklasifikasikan dengan metode SVM dengan kombinasi data *training* dan *testing* berturut-turut 75%:25%, 80%:20%, 85%:15%, 90%:10% dan 95%:10%.

Langkah pertama yang dilakukan pada metode SVM adalah menetukan parameter C dan γ terbaik untuk fungsi Kernel melalui prosedur *tune* dengan *software* R. parameter C dan γ optimum yang diperoleh untuk C adalah 10000 dan γ sebesar 0,001.

Setelah diperoleh nilai parameter C dan γ yang paling optimum, maka selanjutnya nilai tersebut digunakan untuk membentuk model dan kemudian dihitung ketepatan akurasinya. Tabel 4.7 menunjukkan hasil rangkuman ketepatan klasifikasi dari model yang terbentuk.

Tabel 4. 7 Perbandingan Ketepatan Klasifikasi SVM pada Beberapa Kombinasi Data untuk Dataset Delapan Level Dekomposisi

Kombinasi Data *Training* dan Data *Testing*	**Akurasi (1- *APER*) (dalam %)**	
	Data *Training*	**Data *Testing***
75%:25%	99,73	99,19
80%:20%	99,75	98,99
85%:15%	99,76	100
90%:10%	99,77	100
95%:5%	99,79	100

Tabel 4.7 memperlihatkan perbandingan ketepatan klasifikasi dari beberapa kombinasi data untuk dataset delapan level dekomposisi. Untuk data *training* akurasi terbesar adalah pada data *training* 95% dengan nilai ketepatan klasifikasi sebesar 99,79%. Sedangkan nilai ketepatan klasifikasi paling rendah adalah dari data *training* 75% dengan nilai ketepatan klasifikasi

sebesar 99,73%. Ketepatan klasifikasi tertinggi pada data *testing* adalah dari data *testing* 5%, 10% dan 15% dengan ketepatan klasifikasi sebesar 100%. Sedangkan ketepatan klasifikasi paling rendah untuk data *testing* adalah pada data *testing* 20% dengan nilai Ketepatan klasifikasi sebesar 98,99%. Kombinasi data *training* dan data *testing* yang memberikan nilai akurasi terbaik adalah data *training* 95% dan data *tetsing* 5% dengan nilai ketepatan klsifikasi berturut-turut adalah 99,79% dan 100%.

4.3.4 Perbandingan Hasil Ketepatan Klasifikasi dengan Metode *Support Vector Machine* untuk Ketiga Level Dekomposisi

Penelitian ini menggunakan tiga level dekomposisi yang berbeda, yaitu empat level, enam level dan delapan level. Setiap level diklasifikasikan dengan SVM dan melalui prosedur yang sama. Oleh karena itu, perlu diketahui apakah level dekomposisi yang digunakan saat proses DWT berpengaruh terhadap ketepatan klasifikasi yang dihasilkan.

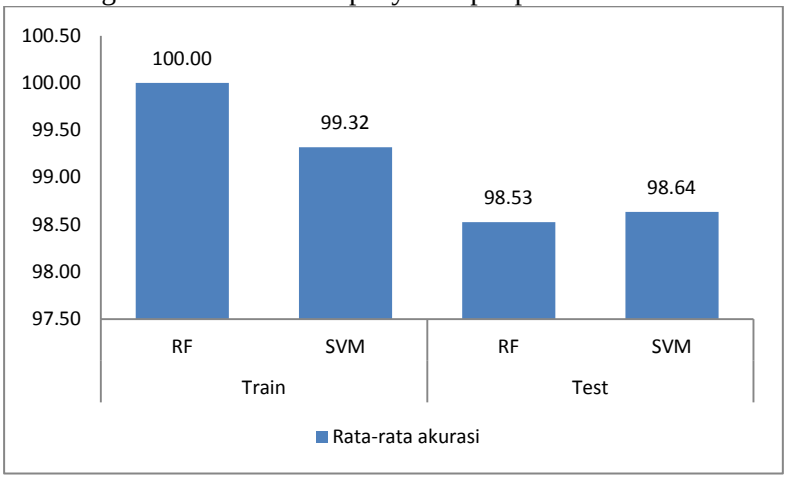
Tabel 4. 8 Perbandingan Rata-rata Ketepatan Kalsifikasi SVM untuk Ketiga Level Dekomposisi

	Rata-rata Ketepatan Klasifikasi (1-*APER*) (dalam %)		
Data	**4 Level**	**6 Level**	**8 Level**
Training	99,58	98,61	99,76
Testing	98,81	97,43	99,64

Tabel 4.9 menunjukkan bahwa untuk data empat level dekomposisi memiliki nilai rata-rata akurasi yang paling tinggi dengan nilai 99,09% untuk data *training* dan 99,38% untuk data *testing*. Sedangkan delapan level dekomposisi memiliki nilai rata-rata akurasi yang paling rendah. Pada metode SVM semakin banyak level yang digunakan nilai rata-rata akurasi cenderung semakin kecil.

4.4 Perbandingan Metode *Random Forest* dan *Support Vector Machine*

Setelah melakukan prediksi pada data pasien epilepsi dengan metode *random forest* dan SVM, langkah selanjutnya adalah membandingkan hasil klasifikasi dari dua metode yang digunakan untuk mengetahui metode mana yang paling baik untuk digunakan mendeteksi penyakit epilepsi.



Gambar 4. 5 Perbandingan Rata-rata Akurasi Metode *Random Forest* dan SVM

Gambar 4.5 menunjukkan grafik rata-rata akurasi dari metode *random forest* dan metode SVM. Pada data *training* metode *random forest* memiliki akurasi yang lebih baik dari pada metode SVM. Nilai rata-rata akurasi data *training* dengan metode *random forest* adalah sebesar 100%. Sedangkan nilai rata-rata akurasi untuk data *training* dengan metode SVM adalah 99,32%. Kemudian, untuk data *testing* metode SVM memiliki rata-rata akurasi yang lebih baik dengan nilai 98,64% dibandingkan dengan metode *random forest* yang nilai rata-rata data *testing*-nya sebesar 98,53%.

Secara individual, metode *random forest* memiliki performa yang lebih baik untuk mendeteksi penyakit epilepsi. Kombinasi data *training* data data *testing* sebesar 95% dan 5% pada metode

random forest masing-masing memiliki nilai akurasi sebesar 100% untuk semua level data yang digunakan. Oleh karena itu, dapat disimpulkan bahwa metode *random forest* lebih baik dari pada metode SVM untuk mendeteksi penyakit epilepsi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh dari proses analisis dan pembahasan yang telah dilakukan adalah sebagai berikut:

1. Tahap *preprocessing* dilakukan dengan metode *Discrete Wavelet Transform* (DWT) untuk mendekomposisi data menjadi empat level, enam level dan delapan level. Setelah, dekomposisi dengan DWT, data diekstraksi dengan metode *Line Length Feature* (LLF). Hasil ekstraksi LLF digunakan sebagai *input* pada tahap klasifikasi.
2. Pada metode *random forest* hasil klasifikasi terbaik adalah kombinasi data *training* dan data *testing* sebesar 95% dan 5% dengan akurasi sebesar 100% baik untuk data *training* maupun data *testing*. Pada metode SVM, hasil klasifikasi terbaik adalah kombinasi data *training* dan data *testing* sebesar 95% dan 5% dengan nilai akurasi sebesar 99,79% untuk data *training* dan 100% untuk data *testing*.
3. Hasil perbandingan kinerja kedua metode menunjukkan bahwa pada data *training* nilai rata-rata dari metode *random forest* lebih baik dari metode SVM. Sedangkan untuk data *testing* nilai rata-rata dari metode SVM lebih baik dari pada *random forest.*

5.2 Saran

Saran yang diberikan untuk penelitian yang akan datang adalah sebagai berikut:

1. Saat proses DWT, selisih level data yang digunakan lebih lebar agar efek perbedaan level dekomposisi terhadap hasil klasifikasi bisa diketahui dengan lebih jelas

2. Metode estimasi parameter fungsi Kernel untuk SVM bisa dilakukan dengan metode lain agar tidak membutuhkan waktu yang lama untuk proses *tuning*

DAFTAR PUSTAKA

Antonini, M.,dan Barlaud, M. (1992). *Image Coding Using Wavelet Transform, Jurnal IEEE on Image Processing,* vol. 1(2). hal. 205-220

Ardilla, Y. (2014). *Seteksi Penyakit Epilepsi dengan Menggunakan Entropi Permutasi, K-Means Clustering dan Multilayer Perceptron.* Jurnal Teknik POMITS. 3(1). 2337-3539

Bintoro, C. A. (2012). *Pemeriksaan EEG untuk Diagnosis dan Monitoring pada Kelainan Neurologi*. Jurnal Medica Hospitalia.1 (1) : 64-70

Breiman, L. (2001). *Random Forests*. Machine Learning. 45(1). 5-32.

Breiman, L., Friedman, J. H., Olshen, R. A.,dan Stone, C. J.. (1993). *Classification and Regression Trees*. New York : Chapman Hall.

Darmawan, A. (2014). Deteksi Dini Penyakit Kanker Leher Rahim (Serviks) di Kota Bogor Mengunakan *Support Vector Machine* (SVM). Surabaya : Institut teknologi Sepuluh Nopember.

Daubechies, I. (1990). *The Wavelet Transform Time-Frequency Localization and Signal Analysis,* Jurnal of IEEE. 36(5). 961-1005.

Esteller, R., Echauz, J., Tcheng, T. Litt, B., dan Pless, B.(2001). *Line Length: An Efficient Feature for Seizure Onset Detection.* 23rd Annual International Conference of the IEEE Engineering in Madicine and Biology Society. 2. 1707-1710.

Esteller, R., Echauz, J., Tcheng, T. (2004). *Comparison of Line Length Feature before and after brain electrical simulation in epileptic patient.* IEMBS '04. 26th Annual

International Conference of the IEEE Engineering in Medicine and Biology Society. 2. 4710-4713.

Geneur, R., Poggi, J. M.,dan Malot, C., T. (2009). *Variable Selection using Random Forests*. France : Laboratoire de Mathematiques, Universite Paris-Sud 11, Bat. 425, 91405.

Guo, L., Rivero, D., Dorado, J., Rabunal, J.R., Pajoz, A. (2010). *Automatic Epileptic Seizure Detectionin EEGs Based on Line Length Feature and Artificial Neural Network.* Journal of Neuroscience Method. 191. 101-109.

Gunn, S. (1998). *Support Vector Machine for Classification and Regression*. Taiwan : National Taiwan University.

Hsu, C.W., Chang, C.C., dan Lin, C.J. (2003). *A Practical Guide to Support Vector Classification*. England : University of Southampton

Hughes, J.R. (2003). *Practical Guide to Epilepsy*. Burlington Butterworth : Heinemenn.

Jesper, H.H. (1958). *The ten electrode system of the International Federation.* Electroencephalagr Clin Neurophysiol. 10. 371-373.

Li, C., Zheng, C., dan Tai, C. (1995). *Detection of EEG Characteristic Points Using Wavelet Transform.* Jurnal IEEE Trans. On Biomedical Eng. 42(1). 21-28.

Mallat. S., dan Hwang, W.L. (1992). *Singularity Detection and Processing with Wavelet,* IEEE Transaction Inform. Theory. 38. 617-643

Martinez, J.P., Almeida, R. Olmos, S., Rocha, A. P., dan Laguna, P. (2004). *A Wavelet-Based ECG Delineator : Evaluation on Standard Database.* Jurnal IEEE Trans. On Biomedical Eng.. 51(4). 570-801.

Nuwer, M.R. (1958). *Recording Ekctrodes Site Nomenclature.* J Clin europhysiol. 10. 371-373.

Pernier, J. P., dan Betrand, F. O. (1988). *Scalp Current Density Fields : Concepts and Properties.* Electroencephalagr Clin Neurophysiol. 69. 385-389.

Rahman, F. (2012). *Klasifikasi Tingkat Keganasan Breast Cancer Dengan Menggunakan Regresi Logistik Ordinal Dan Support Vector Machine*. Surabaya: Institut Teknologi Sepuluh Nopember.

Shorvon, G. S. (2000). *Handbook of Epilepsy Treatment*. London: Springer.

Subasi, A. (2007). *EEG Signal Classification Using Wavelet Feature Extraction and Mixture of Expert Model.* Expert Systems with Application. 32. 1084-1093.

Subasi, A., dan Gursoy, M. I. (2010). *EEG Signal Classification Using PCA, ICA, LDA and Support Vector Machine*. Journal of Expert System with Application. 37. 8659-8666.

Sungkono, J. (2013). *Resampling Bootstrap Pada R*. Magistra No. 84, Th. XXV. ISSN : 0215-9511.

Tawade, L., dan Warpe, H. (2011). *Detection of Epilepsy Disorder Using Discrete Wavelet Transform Using MATLABs*. International Jounal of Advance Science and Technology. 28. 17-24.

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

Halaman

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Lampiran 1 *Syntax* untuk proses DWT dan LLF

```
# -*- coding: utf-8 -*-
import os
import pywt
import math
import pandas as pd
import numpy as np
dataset_path = '/home/sindunuragarp/Documents/idrus/Epileptic Seizure'
level = 3
mode = pywt.MODES.zpd
def LLF(data):
        l = 0
        for i in range(len(data)-1):
                l = l + math.fabs(data[i+1]-data[i])
        l = l / (len(data)-1)
        return l
def code(label):
    if label == "S":
        out = 1
    else:
        out = 0
    return out
data = pd.DataFrame()
target = []
for child in os.listdir(dataset_path):
    child_path = dataset_path+'/'+child
    if os.path.isdir(child_path):
        for root, dirs, files in os.walk(child_path):
            for text_file in files:
                if os.path.isfile(root+'/'+text_file):
```

Lampiran 1 (lanjutan)
Lampiran 1 (lanjutan)

```
  signal = pd.read_csv(root+'/'+text_file)
                dec = pywt.wavedec(signal.values.ravel(), 'db4', mode,
level=level)
                inp = [LLF(x) for x in dec]
                data = data.append([inp])
                target.append(code(child))

data = data.as_matrix()
target = np.array(target)
dataname = "data-" + str(level) + ".csv"
targetname = "target-" + str(level) + ".csv"
np.savetxt(dataname, data, delimiter=",")
np.savetxt(targetname, target, delimiter=",")
```

Lampiran 2 *Syntax* untuk prosedur *tune*

```
#untuk 4 level dekomposisi
data= read.table("d:/Level 4.csv",sep=",", header=FALSE)
tuned <- tune.svm(V1~., data = data, gamma=0.011, cost = 90)
summary(tuned)

#untuk 6 level dekomposisi
data= read.table("d:/Level 6.csv",sep=",", header=FALSE)
tuned <- tune.svm(V1~., data = data, gamma=0.011, cost = 90)
summary(tuned)

#untuk 8 level dekomposisi
data= read.table("d:/Level 8.csv",sep=",", header=FALSE)
tuned <- tune.svm(V1~., data = data, gamma=0.011, cost = 90)
summary(tuned)
```

Lampiran 3 *output tune* SVM

a. *Output* untuk dataset 4 Level Dekomposisi

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma cost
 0.302   91
- best performance: 0.1775
```

b. *Output* untuk dataset 6 Level Dekomposisi

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma cost
 0.011   20
- best performance: 0.1816
```

c. *Output* untuk dataset 6 Level Dekomposisi

```
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma cost
 0.001   10000
- best performance: 0.1948
```

Lampiran 4 Syntax untuk Klasifikasi Random Forest pada Dataset Empat Level Dekomposisi

```
train75= read.table("d://L4_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L4_t25.csv",sep=",", header=FALSE)
```

```
RF75<-randomForest(V1~.,data=train75,mtry=2,ntree=1000)
```

Lampiran 4 (lanjutan)

```
training=predict(RF75, train75[,-1])
tabtrain75=table(pred=training,true=train75[,1])
testing=predict(RF75, tes25[,-1])
tabtes25=table(pred=testing,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

train80= read.table("d://L4_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L4_t20.csv",sep=",", header=FALSE)
RF80<-randomForest(V1~.,data=train80,mtry=2,ntree=1000)
training=predict(RF80, train80[,-1])
tabtrain80=table(pred=training,true=train80[,1])
testing=predict(RF80, tes20[,-1])
tabtes20=table(pred=testing,true=tes20[,1])
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100

train85= read.table("d://L4_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L4_t15.csv",sep=",", header=FALSE)
RF85<-randomForest(V1~.,data=train85,mtry=2,ntree=1000)
training=predict(RF85, train85[,-1])
tabtrain85=table(pred=training,true=train85[,1])
testing=predict(RF85, tes15[,-1])
tabtes15=table(pred=testing,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100

train90= read.table("d://L4_c90.csv",sep=",", header=FALSE)
```

```
tes10= read.table("d://L4_t10.csv",sep=",", header=FALSE)
RF90<-randomForest(V1~.,data=train90,mtry=2,ntree=1000)
```

Lampiran 4 (lanjutan)

```
training=predict(RF90, train90[,-1])
tabtrain90=table(pred=training,true=train90[,1])
testing=predict(RF90, tes10[,-1])
tabtes10=table(pred=testing,true=tes10[,1])
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100

train95= read.table("d://L4_c95.csv",sep=",", header=FALSE)
tes5= read.table("d://L4_t5.csv",sep=",", header=FALSE)
RF95<-randomForest(V1~.,data=train95,mtry=2,ntree=1000)
training=predict(RF95, train95[,-1])
tabtrain95=table(pred=training,true=train95[,1])
testing=predict(RF95, tes5[,-1])
tabtes5=table(pred=testing,true=tes5[,1])
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100

print(akurasitrain75)
print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
print(akurasitrain90)
print(akurasites10)
print(akurasitrain95)
print(akurasites5)
```

Lampiran 5 Syntax untuk Klasifikasi Random Forest pada Dataset Enam Level Dekomposisi

```
train75= read.table("d://L6_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L6_t25.csv",sep=",", header=FALSE)
RF75<-randomForest(V1~.,data=train75,mtry=3,ntree=100)
training=predict(RF75, train75[,-1])
tabtrain75=table(pred=training,true=train75[,1])
testing=predict(RF75, tes25[,-1])
tabtes25=table(pred=testing,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

train80= read.table("d://L6_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L6_t20.csv",sep=",", header=FALSE)
RF80<-randomForest(V1~.,data=train80,mtry=3,ntree=100)
training=predict(RF80, train80[,-1])
tabtrain80=table(pred=training,true=train80[,1])
testing=predict(RF80, tes20[,-1])
tabtes20=table(pred=testing,true=tes20[,1])
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100

train85= read.table("d://L6_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L6_t15.csv",sep=",", header=FALSE)
RF85<-randomForest(V1~.,data=train85,mtry=3,ntree=100)
```

```
training=predict(RF85, train85[,-1])
tabtrain85=table(pred=training,true=train85[,1])
```

Lampiran 5 (lanjutan)

```
testing=predict(RF85, tes15[,-1])
tabtes15=table(pred=testing,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100

train90= read.table("d://L6_c90.csv",sep=",", header=FALSE)
tes10= read.table("d://L6_t10.csv",sep=",", header=FALSE)
RF90<-randomForest(V1~.,data=train90,mtry=3,ntree=100)

training=predict(RF90, train90[,-1])
tabtrain90=table(pred=training,true=train90[,1])
testing=predict(RF90, tes10[,-1])
tabtes10=table(pred=testing,true=tes10[,1])
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100

train95= read.table("d://L6_c95.csv",sep=",", header=FALSE)
tes5= read.table("d://L6_t5.csv",sep=",", header=FALSE)
RF95<-randomForest(V1~.,data=train95,mtry=3,ntree=100)
training=predict(RF95, train95[,-1])
tabtrain95=table(pred=training,true=train95[,1])
testing=predict(RF95, tes5[,-1])
tabtes5=table(pred=testing,true=tes5[,1])
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
```

```
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100
```

Lampiran 5 (lanjutan)

```
print(akurasitrain75)
print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
print(akurasitrain90)
print(akurasites10)
print(akurasitrain95)
print(akurasites5)
```

Lampiran 6 Syntax untuk Klasifikasi Random Forest pada Dataset Delapan Level Dekomposisi

```
train75= read.table("d://L8_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L8_t25.csv",sep=",", header=FALSE)
RF75<-randomForest(V1~.,data=train75,mtry=3,ntree=100)
training=predict(RF75, train75[,-1])
tabtrain75=table(pred=training,true=train75[,1])
testing=predict(RF75, tes25[,-1])
tabtes25=table(pred=testing,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

train80= read.table("d://L8_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L8_t20.csv",sep=",", header=FALSE)
RF80<-randomForest(V1~.,data=train80,mtry=3,ntree=100)
training=predict(RF80, train80[,-1])
tabtrain80=table(pred=training,true=train80[,1])
testing=predict(RF80, tes20[,-1])
```

```
tabtes20=table(pred=testing,true=tes20[,1])
```

Lampiran 6 (lanjutan)

```
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100

train85= read.table("d://L8_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L8_t15.csv",sep=",", header=FALSE)
RF85<-randomForest(V1~.,data=train85,mtry=3,ntree=100)
training=predict(RF85, train85[,-1])
tabtrain85=table(pred=training,true=train85[,1])
testing=predict(RF85, tes15[,-1])
tabtes15=table(pred=testing,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100

train90= read.table("d://L8_c90.csv",sep=",", header=FALSE)
tes10= read.table("d://L8_t10.csv",sep=",", header=FALSE)
RF90<-randomForest(V1~.,data=train90,mtry=3,ntree=100)
training=predict(RF90, train90[,-1])
tabtrain90=table(pred=training,true=train90[,1])
testing=predict(RF90, tes10[,-1])
tabtes10=table(pred=testing,true=tes10[,1])
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100

train95= read.table("d://L8_c95.csv",sep=",", header=FALSE)
tes5= read.table("d://L8_t5.csv",sep=",", header=FALSE)
RF95<-randomForest(V1~.,data=train95,mtry=3,ntree=100)
training=predict(RF95, train95[,-1])
tabtrain95=table(pred=training,true=train95[,1])
```

```
testing=predict(RF95, tes5[,-1])
tabtes5=table(pred=testing,true=tes5[,1])
```

Lampiran 6 (lanjutan)

```
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100

print(akurasitrain75)
print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
print(akurasitrain90)
print(akurasites10)
print(akurasitrain95)
print(akurasites5)
#memanggil data
```

Lampiran 7 *Syntax* untuk Klasifikasi SVM Dataset Empat Level Dekomposisi

```
Train75= read.table("d://L4_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L4_t25.csv",sep=",", header=FALSE)

#membuat model
model= svm(V1~., data=train75, method="C-classification",
kernel ="radial", gamma=0.302, cost=91)
summary(model)

#Prediksi dan akurasi
testing=predict(model, tes25[,-1])
tabtes=table(pred=testing,true=tes25[,1])
tabtes
training=predict(model, train75[,-1])
```

```
tabtrain=table(pred=training,true=train75[,1])
tabtrain
```

Lampiran 7 (lanjutan)

```
train95= read.table("d://L4_c95.csv",sep=",", header=FALSE)

tes5= read.table("d://L4_t5.csv",sep=",", header=FALSE)
model= svm(V1~., data=train95, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training95=predict(model, train95[,-1])
tabtrain95=table(pred=training95,true=train95[,1])
testing5=predict(model, tes5[,-1])
tabtes5=table(pred=testing5,true=tes5[,1])
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100

train90= read.table("d://L4_c90.csv",sep=",", header=FALSE)
tes10= read.table("d://L4_t10.csv",sep=",", header=FALSE)
model= svm(V1~., data=train90, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training90=predict(model, train90[,-1])
tabtrain90=table(pred=training90,true=train90[,1])
testing10=predict(model, tes10[,-1])

tabtes10=table(pred=testing10,true=tes10[,1])
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100
```

```
train85= read.table("d://L4_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L4_t15.csv",sep=",", header=FALSE)
```

Lampiran 7 (lanjutan)

```
model= svm(V1~., data=train85, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training85=predict(model, train85[,-1])
tabtrain85=table(pred=training85,true=train85[,1])
testing15=predict(model, tes15[,-1])
tabtes15=table(pred=testing15,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100

train80= read.table("d://L4_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L4_t20.csv",sep=",", header=FALSE)
model= svm(V1~., data=train80, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training80=predict(model, train80[,-1])
tabtrain80=table(pred=training80,true=train80[,1])
testing20=predict(model, tes20[,-1])
tabtes20=table(pred=testing20,true=tes20[,1])
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100

train75= read.table("d://L4_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L4_t25.csv",sep=",", header=FALSE)
model= svm(V1~., data=train75, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
```

```
training75=predict(model, train75[,-1])
tabtrain75=table(pred=training75,true=train75[,1])
```

Lampiran 7 (lanjutan)

```
testing25=predict(model, tes25[,-1])
tabtes25=table(pred=testing25,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

print(akurasitrain75)

print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
print(akurasitrain90)
print(akurasites10)
print(akurasitrain95)
print(akurasites5)
```

Lampiran 8 *Syntax* untuk Klasifikasi SVM Dataset Enam Level Dekomposisi

```
train95= read.table("d://L6_c95.csv",sep=",", header=FALSE)
tes5= read.table("d://L6_t5.csv",sep=",", header=FALSE)
model= svm(V1~., data=train95, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training95=predict(model, train95[,-1])
tabtrain95=table(pred=training95,true=train95[,1])
testing5=predict(model, tes5[,-1])
tabtes5=table(pred=testing5,true=tes5[,1])
```

```
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
```

Lampiran 8 (lanjutan)

```
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100

train90= read.table("d://L6_c90.csv",sep=",", header=FALSE)
tes10= read.table("d://L6_t10.csv",sep=",", header=FALSE)
model= svm(V1~., data=train90, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training90=predict(model, train90[,-1])
tabtrain90=table(pred=training90,true=train90[,1])
testing10=predict(model, tes10[,-1])
tabtes10=table(pred=testing10,true=tes10[,1])
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100

train85= read.table("d://L6_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L6_t15.csv",sep=",", header=FALSE)
model= svm(V1~., data=train85, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training85=predict(model, train85[,-1])
tabtrain85=table(pred=training85,true=train85[,1])
testing15=predict(model, tes15[,-1])
tabtes15=table(pred=testing15,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100
```

```
train80= read.table("d://L6_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L6_t20.csv",sep=",", header=FALSE)
model= svm(V1~., data=train80, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training80=predict(model, train80[,-1])
tabtrain80=table(pred=training80,true=train80[,1])
testing20=predict(model, tes20[,-1])
tabtes20=table(pred=testing20,true=tes20[,1])
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100

train75= read.table("d://L6_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L6_t25.csv",sep=",", header=FALSE)
model= svm(V1~., data=train75, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training75=predict(model, train75[,-1])
tabtrain75=table(pred=training75,true=train75[,1])
testing25=predict(model, tes25[,-1])
tabtes25=table(pred=testing25,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

print(akurasitrain75)
print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
```

```
print(akurasitrain90)
print(akurasites10)
```

Lampiran 8 (lanjutan)

```
print(akurasitrain95)
print(akurasites5)
```

Lampiran 9 *Syntax* untuk Klasifikasi SVM Dataset Delapan Level Dekomposisi

```
train95= read.table("d://L8_c95.csv",sep=",", header=FALSE)
tes5= read.table("d://L8_t5.csv",sep=",", header=FALSE)
model= svm(V1~., data=train95, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training95=predict(model, train95[,-1])
tabtrain95=table(pred=training95,true=train95[,1])
testing5=predict(model, tes5[,-1])
tabtes5=table(pred=testing5,true=tes5[,1])
akurasitrain95=(1-((length(train95[,1])-
sum(diag(tabtrain95)))/sum(diag(tabtrain95))))*100
akurasites5=(1-((length(tes5[,1])-
sum(diag(tabtes5)))/sum(diag(tabtes5))))*100

train90= read.table("d://L8_c90.csv",sep=",", header=FALSE)
tes10= read.table("d://L8_t10.csv",sep=",", header=FALSE)
model= svm(V1~., data=train90, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training90=predict(model, train90[,-1])
tabtrain90=table(pred=training90,true=train90[,1])
testing10=predict(model, tes10[,-1])
tabtes10=table(pred=testing10,true=tes10[,1])
```

```
akurasitrain90=(1-((length(train90[,1])-
sum(diag(tabtrain90)))/sum(diag(tabtrain90))))*100
```

Lampiran 9 (lanjutan)

```
akurasites10=(1-((length(tes10[,1])-
sum(diag(tabtes10)))/sum(diag(tabtes10))))*100

train85= read.table("d://L8_c85.csv",sep=",", header=FALSE)
tes15= read.table("d://L8_t15.csv",sep=",", header=FALSE)
model= svm(V1~., data=train85, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training85=predict(model, train85[,-1])
tabtrain85=table(pred=training85,true=train85[,1])
testing15=predict(model, tes15[,-1])
tabtes15=table(pred=testing15,true=tes15[,1])
akurasitrain85=(1-((length(train85[,1])-
sum(diag(tabtrain85)))/sum(diag(tabtrain85))))*100
akurasites15=(1-((length(tes15[,1])-
sum(diag(tabtes15)))/sum(diag(tabtes15))))*100

train80= read.table("d://L8_c80.csv",sep=",", header=FALSE)
tes20= read.table("d://L8_t20.csv",sep=",", header=FALSE)
model= svm(V1~., data=train80, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training80=predict(model, train80[,-1])
tabtrain80=table(pred=training80,true=train80[,1])
testing20=predict(model, tes20[,-1])
tabtes20=table(pred=testing20,true=tes20[,1])
akurasitrain80=(1-((length(train80[,1])-
sum(diag(tabtrain80)))/sum(diag(tabtrain80))))*100
akurasites20=(1-((length(tes20[,1])-
sum(diag(tabtes20)))/sum(diag(tabtes20))))*100
```

```
train75= read.table("d://L8_c75.csv",sep=",", header=FALSE)
tes25= read.table("d://L8_t25.csv",sep=",", header=FALSE)
model= svm(V1~., data=train75, method="C-classification", kernel
="radial", gamma=0.001, cost=10000)
summary(model)
training75=predict(model, train75[,-1])
tabtrain75=table(pred=training75,true=train75[,1])
testing25=predict(model, tes25[,-1])
tabtes25=table(pred=testing25,true=tes25[,1])
akurasitrain75=(1-((length(train75[,1])-
sum(diag(tabtrain75)))/sum(diag(tabtrain75))))*100
akurasites25=(1-((length(tes25[,1])-
sum(diag(tabtes25)))/sum(diag(tabtes25))))*100

print(akurasitrain75)
print(akurasites25)
print(akurasitrain80)
print(akurasites20)
print(akurasitrain85)
print(akurasites15)
print(akurasitrain90)
print(akurasites10)
print(akurasitrain95)
print(akurasites5)
```

Lampiran 10 Hasil *preprocessing* untuk 4 Level Dekomposisi

No	Y	D1	D2	D3	D4	A4
1	0	106	78.6	87.1	29.3	5.1
2	0	112	102	92.1	29.8	5.41
3	0	64.6	50.5	45.8	17.2	3.83
4	0	69.5	59.9	49.7	19.6	3.85
5	0	115	99	98.5	33.2	6
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
491	1	538	465	173	45	6.85
492	1	298	488	403	158	20
493	1	1670	2050	1180	255	26.8
494	1	2260	1420	544	172	19.2
495	1	402	230	167	64.7	8.41
496	1	1950	1380	1080	432	59.7
497	1	387	367	130	35.6	6.79
498	1	864	1140	524	94.7	14.1
499	1	439	477	163	53.5	6.29
500	1	1010	1380	1100	310	39.5

Lampiran 11 Hasil *preprocessing* untuk 6 Level Dekomposisi

No	y	D1	D2	D3	D4	D5	D6	A6
1	Z	221	170	113	78.6	87.1	29.3	5.1
2	Z	218	173	102	102	92.1	29.8	5.41
3	Z	99.7	83.2	58.2	50.5	45.8	17.2	3.83
4	Z	155	92	61.9	59.9	49.7	19.6	3.85
5	Z	195	153	107	99	98.5	33.2	6
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
490	S	857	1440	629	381	285	95.1	11.6
491	S	347	399	454	465	173	45	6.85
492	S	252	287	256	488	403	158	20
493	S	211	230	644	2050	1180	255	26.8
494	S	493	708	2100	1420	544	172	19.2
495	S	737	764	378	230	167	64.7	8.41
496	S	1130	1680	1760	1380	1080	432	59.7
497	S	322	247	315	367	130	35.6	6.79
498	S	229	235	601	1140	524	94.7	14.1
499	S	236	311	294	477	163	53.5	6.29
500	S	716	873	909	1380	1100	310	39.5

Lampiran 12 Hasil *preprocessing* untuk 8 Level Dekomposisi

Y	D1	D2	D3	D4	D5	D6	D7	D8	A8
Z	349	360	204	170	113	78.6	87.1	29.3	5.1
Z	186	211	203	173	102	102	92.1	29.8	5.41
Z	138	144	93.3	83.2	58.2	50.5	45.8	17.2	3.83
Z	187	133	135	92	61.9	59.9	49.7	19.6	3.85
Z	148	159	181	153	107	99	98.5	33.2	6
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
S	319	587	812	1440	629	381	285	95.1	11.6
S	159	362	394	399	454	465	173	45	6.85
S	91.5	117	290	287	256	488	403	158	20
S	227	134	190	230	644	2050	1180	255	26.8
S	364	297	508	708	2100	1420	544	172	19.2
S	177	269	699	764	378	230	167	64.7	8.41
S	510	561	1040	1680	1760	1380	1080	432	59.7
S	138	284	269	247	315	367	130	35.6	6.79
S	147	208	216	235	601	1140	524	94.7	14.1
S	77.9	86.4	230	311	294	477	163	53.5	6.29

Lampiran 13 Tabel Koefisien orthogonal Daubechies untuk *Low Pass Filtering*

Untuk *high pass filtering* koefisien menjadi $b_k = (-1)^k a_{N-1-k}$, dengan *b* adalah koefisien wavelet, *k* adalah indeks panjang wavelet

D2(Haar)	D4	D6	D8	D10	D12	D14	D16	D18	D20
1	0.683013	0.470467	0.325803	0.226419	0.157742	0.110099	0.076956	0.05385	0.037717
1	1.183013	1.141117	1.010946	0.853944	0.699504	0.560791	0.442467	0.34483	0.266122
	0.316987	0.650365	0.892201	1.024327	1.062264	1.031148	0.955486	0.85535	0.745575
	-0.18301	-0.19093	-0.03958	0.195767	0.445831	0.664372	0.827817	0.92955	0.973628
		-0.12083	-0.26451	-0.34266	-0.31999	-0.20351	-0.02239	0.18837	0.397638
		0.049818	0.043616	-0.0456	-0.18352	-0.31684	-0.40166	-0.41475	-0.35334
			0.046504	0.109703	0.137888	0.100847	6.68E-04	-0.13695	-0.27711
			-0.01499	-0.00883	0.038923	0.114003	0.182076	0.21007	0.180127
				-0.01779	-0.04466	-0.05378	-0.02456	0.04345	0.131603
				4.72E-03	7.83E-04	-0.02344	-0.06235	-0.09565	-0.10097
					6.76E-03	0.01775	0.019772	3.55E-04	-0.04166
					-1.52E-03	6.08E-04	0.012369	0.03162	0.04697
						-2.55E-03	-6.89E-03	-6.68E-03	5.10E-03
						5.00E-04	-5.54E-04	-6.05E-	-0.01518

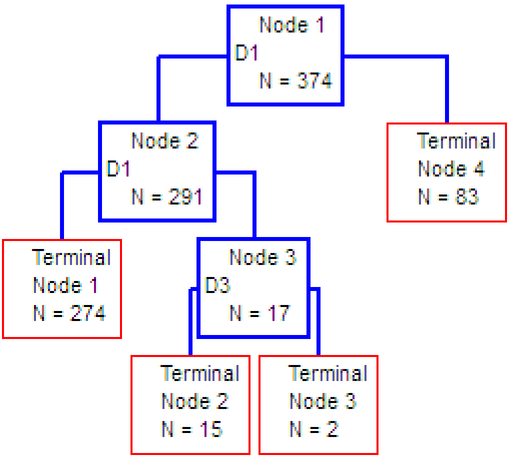
								03	
							9.55E-04	2.61E-03	1.97E-03
							-1.66E-04	3.26E-04	2.82E-03
								-3.56E-04	-9.70E-04
								5.56E-05	-1.65E-04
									1.32E-04
									-1.88E-05

Lampiran 14. Ilustrasi Random Forest

a. Hasil salah satu pohon klasifikasi yang terbentu untuk data empat level dekomposisi

Variabel	Skor
D1	100
D2	95.88
D4	73.02
D3	72.85
A4	43.5

Tabel diatas adalah tabel kontribusi tiap variabel prediktor untuk pembentukan pohon k lasifikasi. Terlihat bahwa variabel D1 memiliki kontribusi terbesar dengan skor 100.



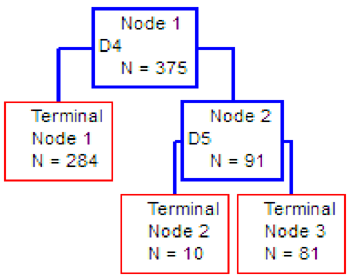
Karakteristik kelas 0 (bukan penderita epilepsi) dipilih dari terminal node 4. Hal tersebut dikarenakan dari 3 terminal node yang memprediksi kelas 0, terminal node 1 memiliki presentase

terbesar. Terminal node 1 memiliki karakteristik variabel D1>323. Sedangkan untuk kelas 1 (penderita epilepsi) terminal node yang dipilih adalah terminal node 4 dengan karakteristik variabel D1≤244,5.

b. Hasil salah satu pohon klasifikasi yang terbentu untuk data enam level dekomposisi

Variabel	Skor Variabel
D3	100
D4	97.32
D5	86.69
D6	80.48
A6	53.41
D2	44.51
D1	0

Tabel diatas adalah tabel kontribusi tiap variabel prediktor untuk pembentukan pohon klasifikasi. Terlihat bahwa variabel D3 memiliki kontribusi terbesar dengan skor 100 dan variabel D4 dengan skor 97,32.



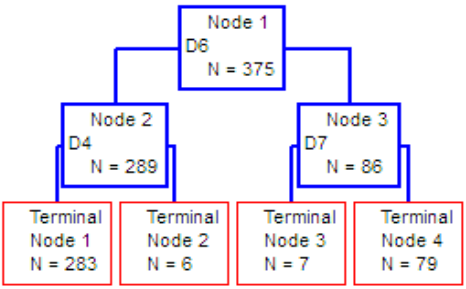
Karakteristik kelas 0 (bukan penderita epilepsi) dipilih dari terminal node 3. Hal tersebut dikarenakan dari 2 terminal node

yang memprediksi kelas 0, terminal node 3 memiliki presentase terbesar. Terminal node 1 memiliki karakteristik variabel D4>166,5 dan variabel D5>94,05. Sedangkan untuk kelas 1 (penderita epilepsi) terminal node yang dipilih adalah terminal node 1 dengan karakteristik variabel D4≤166,5.

c. Hasil salah satu pohon klasifikasi yang terbentu untuk data delapan level dekomposisi

Variabel	Skor Variabel
D6	100
D5	95.48
D7	83.87
D8	78.82
D4	49.33
A8	48.29
D3	2.98
D2	2.41

Tabel diatas adalah tabel kontribusi tiap variabel prediktor untuk pembentukan pohon k lasifikasi. Terlihat bahwa variabel D6 memiliki kontribusi terbesar dengan skor 100 dan variabel D5 dengan skor 95,48.



Karakteristik kelas 0 (bukan penderita epilepsi) dipilih dari terminal node 4. Hal tersebut dikarenakan dari 3 terminal node yang memprediksi kelas 0, terminal node 3 memiliki presentase terbesar. Terminal node 4 memiliki karakteristik variabel D6>193,5 dan variabel D7>121,5. Sedangkan untuk kelas 1 (penderita epilepsi) terminal node yang dipilih adalah terminal node 1 dengan karakteristik variabel D6≤166,5 dan variabel D4≤555.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Penulis bernama lengkap Muhammad Idrus Fachruddin.Lahir di Bojonegoro 17 Oktober 1993, merupakan anak pertama dari empat bersaudara.Saat ini tinggal di Desa Karangkembang, Kecamatan Babat, Kabupaten Lamongan. Penulis menempuh pendidikan formal dari SD hingga MTs di Yayasan Al Falah, Baureno. Kemudian melanjutkan SMA di SMAN 1 Babat.Penulis diterima di jurusan Statistika ITS angkatan 2011 melalui jalur SNMPTN Tulis. Selain sibuk dengan perkuliahan di kelas, penulis juga menghabiskan perkuliahan 4 tahun di ITS dengan aktif di beberapa kepanitian dan organisasi diantaranya GERIGI 2012, RDK 33 dan 34, MUNAS IHMSI, PRS 2013 serta PJ regional Pekanbaru STATION 2013. Organisasi yang pernah diikuti oleh penulis diantaranya, staf BPU JMMI 12/13, Kadep Tablighul Islam FORSIS 12/13, staf ITS Bangun Desa, Ketua Umum FORSIS ITS 13/14 dan Kadep Syiar KKH JMMI 14/15. Penulis juga menjabat sebagai Komting Statistika angkatan 2011 hingga saat ini. Penulis dapat dihubungi melalui *e-mail* idrus.stat11@gmail.com atau nomor telefon 085706752356.