ISSN 2085-4552

Rancang Bangun Aplikasi Pendeteksi Penyakit Ginjal Kronis dengan Menggunakan Algoritma C4.5

Robi Rianto[1], Ni Made Satvika Iswari[2]
Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
robyryanto@yahoo.com[1], satvika@umn.ac.id2

Diterima 12 April 2017
Disetujui 22 Mei 2017

Abstract*— Kidneys are two bean-shaped organs, each about the size of a fist. They are located just below the rib cage, one on each side of the spine. Kidneys are vital organs contained in the human body and serve to filter the blood of metabolic waste product and throw it in the form of urine. Given the changing conditions in the body can affect the kidneys, causing a decrease in the function of these organ and lead to chronic kidney disease. In an article on the website of the National Institute of Diabetes and Digestive and Kidney Diseases (2014) argued that chronic kidney disease is a silent disease, in which patients appear normal and show no symptomp but the test results stating the patient's kidney function had decreased. Based on the above information, the research on how to detect chronic kidney disease will be conducted. Applications built on the basis of desktop and using Decision Tree algorithm C4.5. The trial was conducted to determine how much the level of accuracy that can be generated by the application. The testing process is done by using cross-validation and based on the results already calculated this application has an accuracy of 91.50% at the time of the decision tree is made without using preprocess menu.

***Index Terms*— C4.5, *chronic kidney disease, decision tree, detection system*.**

I. PENDAHULUAN

Ginjal merupakan salah satu organ penting yang terdapat dalam tubuh manusia dan berfungsi untuk menyaring sampah hasil metabolisme pada darah lalu membuangnya dalam bentuk urin[1]. Namun adanya perubahan kondisi tertentu dalam tubuh dapat mempengaruhi kondisi ginjal sehingga menyebabkan penurunan fungsi organ tersebut dan memicu penyakit ginjal kronis.

Dalam artikel yang terdapat pada *website* milik *National Institute of Diabetes and Digestive and Kidney Diseases* pada tahun 2014 [2] dikatakan bahwa penyakit ginjal kronis merupakan *silent disease*, dimana pasien terlihat normal dan tidak menunjukkan gejala tetapi hasil tes menyatakan ginjal pasien tersebut sudah mengalami penurunan fungsi.

Peran ginjal yang penting bagi tubuh manusia dan sulitnya melakukan deteksi terhadap penyakit ginjal kronis menjadi latar belakang untuk membangun sebuah aplikasi yang dapat melakukan prediksi terhadap penyakit tersebut. Aplikasi dibangun agar penyakit ginjal kronis dapat dideteksi sejak dini dan mencegah penyakit tersebut berkembang sehingga menyebabkan gagal ginjal pada penderita.

Penelitian sebelumnya dilakukan oleh Abdul Rohman[3], yang bertujuan untuk membantu dalam proses pendeteksian penyakit jantung dan menggunakan algoritma C4.5. Algoritma berhasil melakukan prediksi terhadap penyakit jantung, dan menghasilkan akurasi 86.59%.

Penelitian lainnya telah dilakukan Nita Apriliani Puteri, Warih Maharani, dan Mahmud Dwi Suliiyo [4], yang juga dilakukan untuk mendeteksi penyakit jantung, tetapi algoritma yang digunakan adalah *classification and regression tree*. Algoritma tersebut berhasil melakukan prediksi dan menghasilkan nilai akurasi sebesar 77%.

Pada prinsipnya, *decision tree* digunakan untuk memprediksi keanggotaan objek untuk kategori yang berbeda (*class*), dengan mempertimbangkan nilai-nilai yang sesuai dengan atributnya[5]. Algoritma C4.5 adalah algoritma yang berfungsi untuk membuat *Decision Tree* berdasarkan *dataset* yang telah disediakan [6].

Berdasarkan informasi di atas maka penelitian mengenai cara mendeteksi penyakit ginjal kronis akan dilakukan. Aplikasi akan dibangun dengan basis *desktop* dan menggunakan metode *Decision Tree* dengan algoritma C4.5. Pemilihan algoritma tersebut dikarenakan C4.5 dapat menyimpulkan suatu keputusan berdasarkan sebuah *dataset* yang ada dan perbandingan antara penelitian yang sudah ada sebelumnya, hasil yang didapatkan oleh penelitian menggunakan algoritma C4.5 lebih besar dibandingkan menggunakan algoritma *classification and regression tree*. Perbedaan dengan penelitian sebelumnya adalah penelitian ini memprediksi penyakit ginjal kronis menggunakan data yang didapat dari *dataset* dengan metode *Decision Tree* dan algoritma C4.5.

II. DASAR TEORI

A. Ginjal Kronis

Ginjal kronis merupakan akibat akhir dari kehilangan fungsi ginjal lanjut secara bertahap. Kegagalan ginjal kronis terjadi bila ginjal sudah tidak mampu mempertahankan lingkungan internal yang konsisten dengan kehidupan dan pemulihan fungsi tidak dimulai [7].

Seiring bertambahnya usia juga akan diikuti oleh penurunan fungsi ginjal. Hal tersebut terjadi terutama karena pada usia lebih dari 40 tahun akan terjadi proses hilangnya beberapa nefron. Dengan semakin meningkatnya usia dan ditambah dengan penyakit kronis seperti tekanan darah tinggi atau diabetes, ginjal cenderung akan menjadi rusak dan tidak dapat dipulihkan kembali. Selain dari faktor usia, penurunan fungsi ginjal juga dapat disebabkan oleh gaya hidup, gaya hidup tidak banyak bergerak ditambah dengan pola makan buruk yang tinggi lemak dan karbohidrat serta tidak diimbangi serat menyebabkan menumpuknya lemak dengan gejala kelebihan berat badan. Dalam jangka panjang, hal tersebut akan menyebabkan penumpukan lemak dalam lapisan pembuluh darah. Ginjal bergantung pada sirkulasi darah dalam menjalankan fungsinya sebagai pembersih darah dari sampah metabolisme tubuh [8].

B. Data mining

Data mining adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan dan machine learning untuk mengambil dan mengenal informasi yang terdapat pada berbagai *database*. Data mining sering disebut juga sebagai *Knowledge Discovery in Database* (KDD)[9].

C. Decision Tree

Pada prinsipnya, *decision tree* digunakan untuk memprediksi keanggotaan objek untuk kategori yang berbeda (*class*), dengan mempertimbangkan nilai-nilai yang sesuai dengan atributnya. Meskipun *decision tree* tidak begitu luas di bidang *pattern recognition* dari sudut pandang statistik probabilistik pandang, algoritma ini banyak digunakan di domain lainnya seperti, misalnya, obat-obatan (diagnosis), ilmu komputer (struktur data), botani (klasifikasi), psikologi (teori keputusan perilaku), dan sebagainya[5].

Berikut merupakan tahap-tahap dalam pembuatan *Decision Tree* menurut Dr. Saed Sayad[13]:

1. Hitung *entropy* dari target yang ingin dicari dengan rumus ini jika menghitung *entropy* untuk satu *attribute*.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \qquad (1)$$

E(S) : Entropy target
Pi : Peluang terjadinya kejadian i

Jika menghitung *entropy* untuk dua *attribute* maka gunakan rumus berikut.

$$E(T,X) = \sum_{c \in X} P(c)E(c) \qquad (2)$$

E(T,X) : Entropy target
P(c) : Peluang terjadinya kejadian c
E(c) : Nilai Entropy c

2. Hitung *Information Gain* dengan rumus berikut :

$$Gain(T.X) = Entropy(T) - Entropy(T,X) \qquad (3)$$

3. Lalu pilih *attribute* dengan nilai *information gain* terbesar untuk dijadikan sebagai *decision node.*
4. Cabang yang mimiliki nilai *entropy* 0 merupakan *leaf node*.
5. Cabang yang memiliki nilai *entropy* lebih dari 0 berarti harus dipecah kembali.

D. Algoritma C4.5

Algoritma C4.5 adalah algoritma yang berfungsi untuk membuat Decision Tree berdasarkan *dataset* yang telah disiapkan. Secara umum algoritma ini digunakan untuk membuat Decision Tree dengan cara sebagai berikut [6]:

1. Pilih *attribute* sebagai akar.
2. Buat cabang untuk masing-masing nilai.
3. Bagi kasus dalam cabang.
4. Ulangi proses untuk masing - masing nilai.
5. Ulangi proses untuk masing - masing cabang sampai semua kasus selesai.

E. Data Preprocessing

Data Preprocessing digunakan untuk meningkatkan kualitas data yang akan dipakai dalam data mining, beberapa faktor yang menentukan tingkat kualitas data yaitu *accuracy*, *completeness*, *consistency*, *timeliness*, *believability*, dan *interpretability*. Beberapa tugas utama yang dilakukan untuk melakukan *data preprocessing* adalah sebagai berikut [10]:

1. **Data Cleaning**
   Dalam proses *data cleaning* pekerjaan yang dilakukan adalah mencoba melakukan pengisian *missing value*, melancarkan *noise*, mengidentifikasi *outlier* dan membenarkan *inconsistencies* dalam *dataset*.
2. **Data Integration**
   Data integration dilakukan dengan tujuan untuk menghindari dan mengurangi *redundancies* dan *inconsistencies* yang terdapat pada *dataset*.
3. **Data Reduction**
   Teknik *data reduction* dapat digunakan untuk mengurangi representasi data yang digunakan, tetapi tetap mempertahankan integrity dari data asli.
4. **Data Transformation dan Data Discretization**

Data transformation dilakukan dengan tujuan untuk membuat hasil dari *data mining* menjadi lebih *efficient* dan pola yang dihasilkan menjadi lebih mudah dimengerti.

F. Cross-Validation

Dalam *k-fold cross-validation* data awal secara acak dibagi menjadi k. Pelatihan dan pengujian dilakukan berulang sebanyak k. Pada iterasi i partisi digunakan sebagai *test set*, dan partisi lainnya secara kolektif digunakan untuk melatih model. Berbeda dengan metode *holdout and random subsampling*, *cross-validation* menggunakan setiap sampel dengan jumlah pengulangan yang sama untuk pelatihan dan sekali untuk *testing*, estimasi akurasi dihasilkan dari jumlah *testing* yang berhasil dibagi dengan total data awal. Secara umum *10-fold cross-validation* dianjurkan untuk memperkirakan akurasi karena bias dan variansi yang relatif rendah [10].

G. Weka Library

Weka library merupakan sebuah koleksi dari algoritma *machine learning* yang digunakan untuk pekerjaan *data mining*. Algoritma yang terdapat dalam *weka library* dapat digunakan langsung dengan menggunakan *software* yang dibuat oleh *weka* atau dapat digunakan dalam pembuatan aplikasi yang berbasis *java*. Dalam *library* tersebut terdapat perlengkapan tentang *data preprocessing*, *classification*, *regression*, *clustering*, *association rules*, dan *visualization* yang dapat membantu dalam pembuatan aplikasi. [11].

H. Dataset

Dalam penelitian ini, digunakan *dataset* yang didapatkan dari *website The UCI Machine Learning Repository*. *Website* tersebut menyediakan koleksi *database*, *domain theories*, dan *data generators* yang banyak digunakan oleh *machine learning community* untuk melakukan sebuah analisa terhadap algoritma *machine learning*[14]. Tabel 1 menyajikan *dataset* yang digunakan dalam penelitian ini beserta singkatan yang digunakan dalam artikel ini untuk kemudahan penulisan.

Tabel 1 *Dataset* sebagai Data Input Algoritma

No.	Singkatan	Data Input
1.	age	Age
2.	bp	Blood Pressure
3.	sg	Specific Gravity
4.	al	Albumin
5.	su	Sugar
6.	rbc	Red Blood Cells
7	pc	Pus Cell
8.	pcc	Pus Cell Clumps
9.	ba	Bacteria
10.	bgr	Blood Glucose Random
11.	bu	Blood Urea
12.	sc	Serum Creatinine
13.	sod	Sodium
14.	pot	Potassium
15.	hemo	Hemoglobin
16.	pcv	Packed Cell Volume
17.	wc	White Blood Cell Count
18.	rc	Red Blood Cell Count
19.	htn	Hypertension
20.	dm	Diabetes Mellitus
21.	cad	Coronary Artery Disease
22.	appet	Appetite
23.	pe	Pedal Edema
24.	ane	Anemia

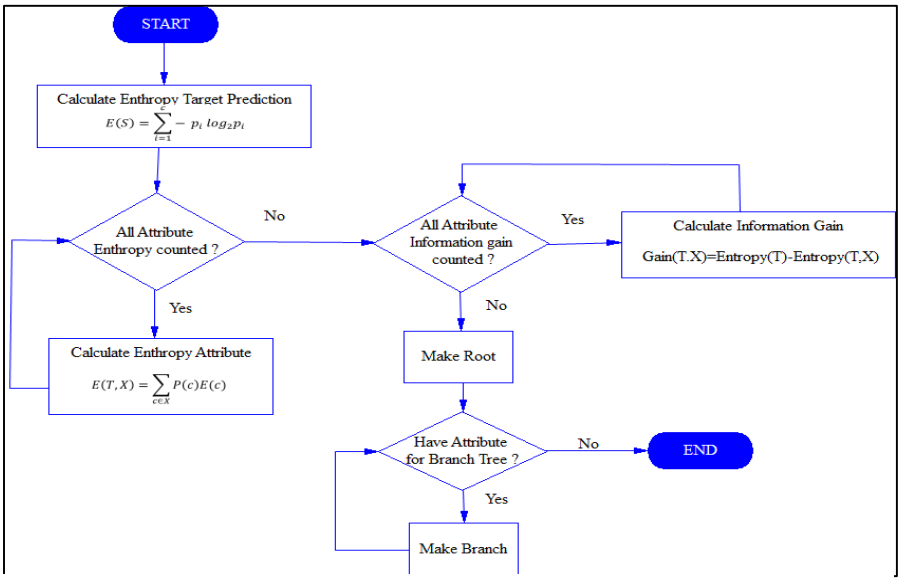
III. METODE PENELITIAN DAN PERANCANGAN APLIKASI

Pada perancangan aplikasi, digunakan *flowchart* untuk merancang fungsi-fungsi dan fitur-fitur pada aplikasi. Rancangan *flowchart* digunakan berdasarkan urutan proses yang dilakukan oleh aplikasi. Secara umum, yang dapat dilakukan pengguna pada aplikasi ini adalah melihat *dataset* yang digunakan untuk pembuatan *decision tree*, melihat *report* dari pembentukan *decision tree* dan melakukan prediksi terhadap penyakit ginjal kronis. Untuk merancang data apa saja yang digunakan pada aplikasi saat dijalankan, digunakan *Data Flow Diagram* pada saat perancangan. *Data Flow Diagram* terdiri dari tiga *level*, yaitu diagram *Context* DFD *level* 1, dan 2.

A. Flowchart

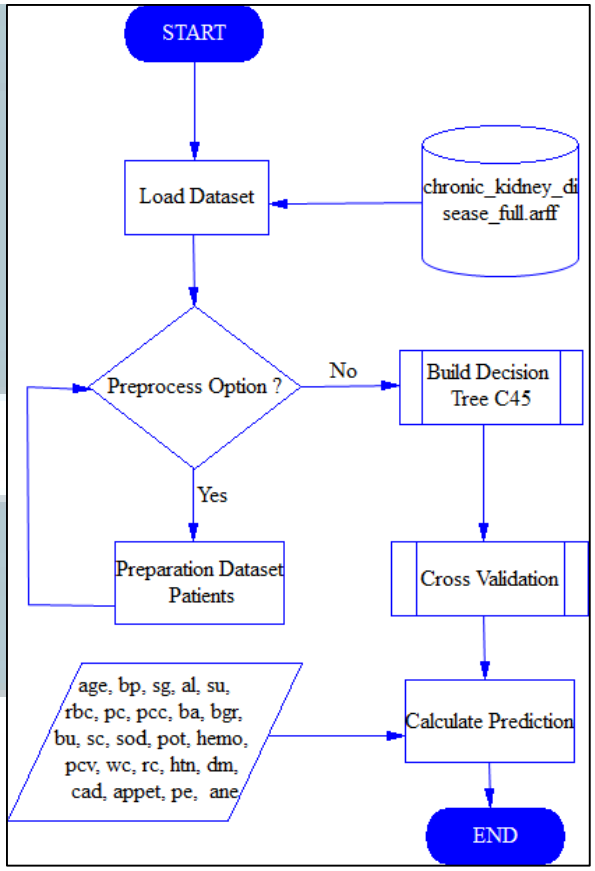
Pada Gambar 1 di bawah aplikasi ini akan digunakan oleh *user* dan pada saat *user* membuka aplikasi maka aplikasi langsung melakukan proses *load data*. Setelah proses tersebut selesai *user* dapat memilih preproses apa saja yang akan digunakan dalam pembuatan *decision tree*¸ saat *dataset* berhasil dibuat oleh aplikasi dengan menggunakan proses pengecekan terhadap *preprocess option*, jika *user* memilih beberapa *preprocess option* maka aplikasi akan menjalakan proses *preparation dataset patients* berdasarkan dari *preprocess option* yang dipilih oleh *user*. Setelah itu proses pengecekan kembali dilakukan, apakah masih ada *preprocess option* yang belum dilakukan, jika masih ada maka proses akan kembali melakukan proses *preparation dataset patients*, jika tidak maka aplikasi siap untuk melakukan proses selanjutnya. Setelah proses *preparation dataset patients* dilakukan maka *user* dapat melihat data-data yang akan digunakan dalam pembuatan *decision tree*.

Proses yang dilakukan berikutnya adalah *build decision tree C4.5* dan *Cross-Validation*. Disaat kedua proses selesai dilakukan, *user* dapat melihat *report decision tree* yang sudah dibuat, *report* tersebut berupa bentuk *decision tree*, *size of tree*, *number of leaf* dan juga akurasi yang dihasilkan oleh proses pengujian *cross-validation*. Proses penghitungan akurasi dilakukan dengan membagi *dataset* menjadi *training set* dan juga *test set*.



Gambar 1 Diagram Alur Proses *Build Decision Tree*

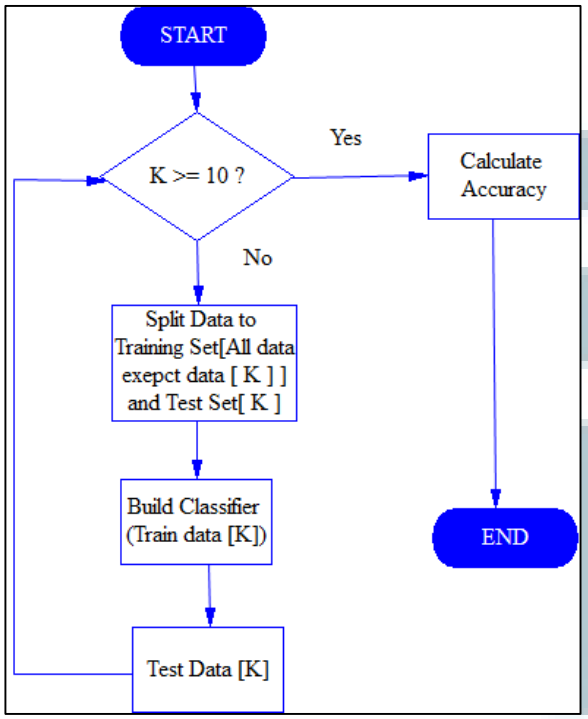
Setelah proses diuji maka aplikasi dapat melakukan penghitungan terhadap *input* yang dimasukan oleh *user*. Hasil yang diberikan oleh proses ini adalah berupa pernyataan apakah data yang dimasukkan tergolong terkena penyakit *chronic kidney disease* atau tidak terkena penyakit tersebut. Pada saat proses ini selesai dilakukan maka *user* dapat tetap melihat *dataset* yang digunakan, *report* dari hasil pembuatan *decision tree*.



Gambar 2 Diagram Alur Aplikasi *Chronic Kidney Disease Detection System*

Pada Gambar 2 menunjukkan proses *build decision tree C4.5* yang dibantu dengan *weka* library. Proses dimulai dari perhitungan *entrophy target prediction* yang merupakan *attribute class* pada *dataset* setelah itu proses dilanjutkan dengan penghitungan *entrophy attribute* yang lain. Setelah sistem mendapatkan seluruh

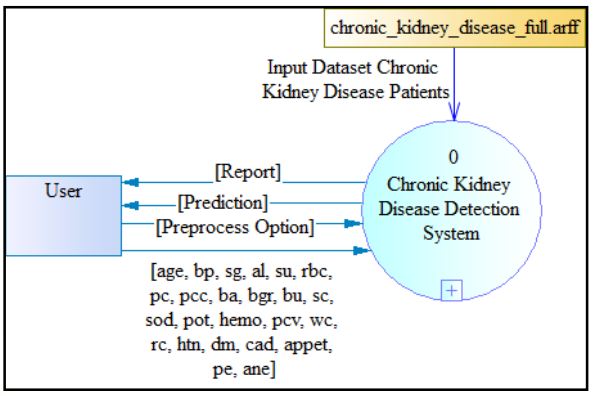
entrophy dari masing-masing *attribute* maka akan dilakukan penghitungan terhadap *information gain* tiap *attribute*. Setelah perhitungan dilakukan maka proses selanjutnya adalah proses pembuatan *root tree* yang diambil berdasarkan *attribute* yang memiliki nilai *information gain* terbesar. Proses selanjutnya adalah pengecekan apakah masih ada *attribute* yang belum menjadi *branch tree*. Jika masih maka proses pembuatan *branch tree* akan dilakukan tapi jika sudah habis maka proses telah selesai membuat *decision tree* dan *rules* dengan algoritma C4.5.



Gambar 3 Diagram Alur Proses *Cross-Validation*

Pada Gambar 3 menunjukkan bahwa proses *cross-validation* dimulai dengan melakukan proses *split data to training set and test set*, dalam proses ini dilakukan pembagian *dataset* menjadi *training set* dan juga *test set*, pembagian ini dilakukan berdasarkan *variable* k. *Dataset* dibagi menjadi 10 bagian lalu *training set* terdiri dari semua bagian *dataset* kecuali *index* k, dan *test set* terdiri dari bagian *dataset* yang memiliki *index* k. Proses selanjutnya adalah membuat *decision tree* berdasarkan dari *training set* yang sudah dibuat pada saat proses sebelumnya. Pembuatan *decision tree* ini dibantu dengan menggunakan *weka library*. Setelah pembuatan itu berhasil selanjutnya diuji dengan menggunakan *test data* yang sudah dibagi sebelumnya. Proses pengujian ini menghasilkan *accuracy* pada setiap putaran K, *accuracy* akhir akan didapatkan saat putaran K mencapai 10 dan hasilnya akan dirata-rata untuk mendapatkan *accuracy* terakhir. Proses perhitungan *accuracy* tersebut dilakukan di proses terakhir yaitu *calculate accuracy*.

B. Data Flow Diagram



Gambar 4 Context Diagram Sistem Pendeteksi Penyakit Ginjal Kronis

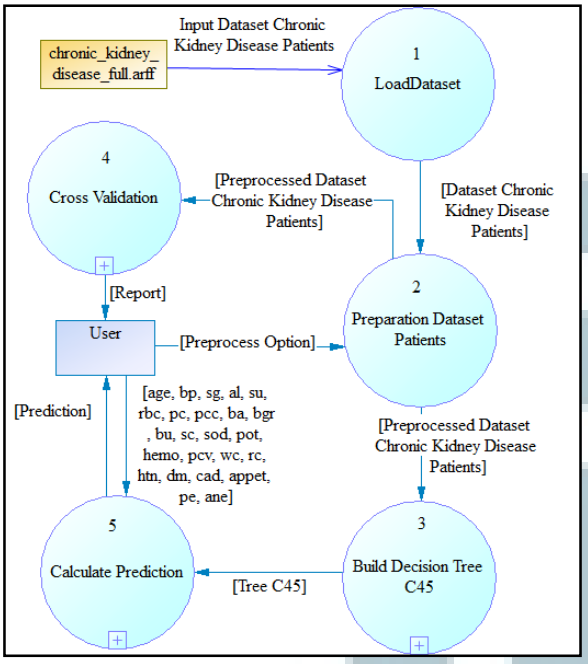
Gambar 4 di atas menunjukkan *Context* Diagram dari Sistem Pendeteksi Penyakit Ginjal Kronis. *Dataset* yang diambil dari arff *file* merupakan *dataset* yang akan digunakan untuk melakukan pembentukan *decision tree* yang dibuat dengan menggunakan *weka library*. *User* memasukkan *input* berupa *preprocess option* yang berfungsi untuk memilih cara-cara preproses yang ingin dilakukan terhadap *dataset*, setelah itu *input* selanjutnya adalah data *user* yang ingin melakukan penghitungan prediksi. Data yang diperlukan age, bp, sg, al, su, rbc, pc, pcc, ba, bgr, bu, sc, sod, pot, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane. *Output* yang akan diterima oleh user berupa preproses *report* yang berupa laporan tentang preproses terhadap *dataset* dan juga hasil dari prediksi yang dilakukan oleh *system* dengan *decision tree* yang sudah dibuat dan diuji sebelumnya.

Gambar 5 di bawah menunjukkan *Data Flow Diagram level* 1, pada diagram ini dijelaskan mengenai pemecahan *system* menjadi beberapa proses. Proses awal yang dilakukan oleh *system* adalah membaca data yang terdapat pada arff *file* selanjutnya *system* akan mempesiapkan *dataset* tersebut dengan cara melakukan beberapa preproses yang sudah dipilih oleh *user*. *Input user* tersebut juga terdapat pada gambar di atas yang berupa preproses *option*, preproses yang dapat dipilih oleh *user* ini terbagi menjadi 3 jenis yaitu *remove low information gain*, *replace missing value(mean , mode)*, dan *discretize*.

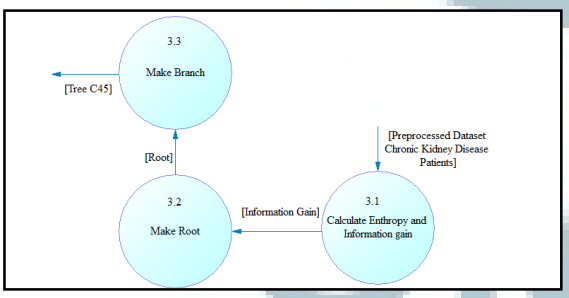
Setelah data siap untuk diproses maka *output* berupa data olahan tersebut dikirimkan ke proses *Cross-validation* agar dapat diuji berapakah akurasi yang dimiliki oleh aturan yang dibuat oleh *decision tree*. *Output* berupa data olahan tadi juga dikirimkan ke proses pembuatan *decision tree* C4.5 untuk dibuat *decision tree* dengan menggunakan bantuan *weka library* dalam perhitungan *information gain* dan juga pembuatan *rules* untuk *decision tree*.

Selanjutnya *tree C4.5* dikirimkan ke proses perhitungan prediksi, disini proses penghitungan

prediksi dilakukan setelah menerima *input* dari user yang berupa data lengkap pasien yaitu age, bp, sg, al, su, rbc, pc, pcc, ba, bgr, bu, sc, sod, pot, hemo, pcv, wc, rc, htn, dm, cad, appet, pe, ane setelah itu *system* akan melakukan penghitungan berdasarkan *rules* yang sudah dibuat sebelumnya dan akhirnya proses tersebut akan memberikan *output* kepada *user* yang berupa hasil dari prediksi penyakit berdasarkan data pasien dan juga *rules* yang dibuat.

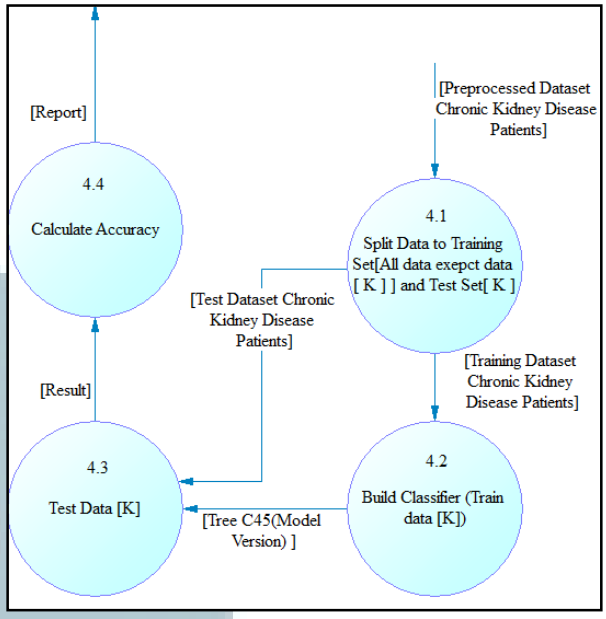


Gambar 5 *Data Flow Diagram Level 1*



Gambar 6 *Data Flow Diagram Build Decision Tree*

Gambar 6 di atas menunjukkan *Data Flow Diagram Build Decision Tree* secara detail, proses pembuatan *decision tree* ini dibuat dengan bantuan *weka library*. Proses ini dimulai dengan penghitungan *enthropy* dan juga *information gain* dari *attribute dataset*, *attribute* ini diambil dari *input* yang berupa *preprocessed dataset chronic kidney disease*. Setelah proses perhitungan berhasil maka nilai *information gain* dikirimkan ke proses selanjutnya, yaitu proses *make root* yang berfungsi untuk memilih *attribute* untuk dijadikan *root* dari *decision tree* yang akan dibuat. Selanjutnya *root* yang telah dibuat dikirimkan ke proses *make branch*, proses ini berfungsi untuk membuat *branch* pada *decision tree* yang dibuat. Proses pembuatan *branch* tersebut terus berulang sampai *attribute dataset* yang terakhir dan tidak memiliki *branch* selanjutnya lagi. Setelah proses ini selesai maka *decision tree* yang terbentuk dikirimkan ke proses selanjutnya.

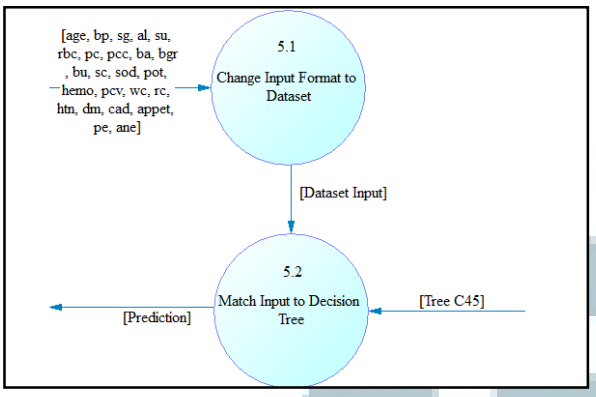


Gambar 7 *Data Flow Diagram Cross-validation*

Gambar 7 di atas menunjukkan *Data Flow Diagram Cross-validation* yang merupakan proses pengujian yang dilakukan terhadap *preprocessed dataset chronic kidney disease*. Proses dimulai dengan membagi *dataset* tersebut menjadi *training data* yang terdiri dari seluruh *dataset* kecuali bagian *dataset* k dan *test data* yang terdiri dari *dataset* bagian k. *Training data* dikirimkan ke proses *build classifier* untuk dijadikan *classifier model* berdasarkan algoritma C4.5. Selanjutnya *test data* akan dikirimkan ke proses *test data [k]* yang berfungsi untuk menguji *decision tree* yang dibuat berdasarkan *train data*, pada proses *test data [k]* ini terdapat *input* data yang dihasilkan oleh proses *build classifier* yang berupa *decision tree*. Setelah proses pengujian telah selesai maka data yang berupa *result* akan dikirimkan ke proses *calculate accuracy* agar proses ini dapat menghitung seberapa besar *accuracy* yang didapatkan dari hasil pengujian tersebut. Setelah itu hasil yang dihasilkan berupa data yang bernama *report* dan dikirimkan ke proses selanjutnya.

Gambar 8 di bawah menunjukkan *Data Flow Diagram Calculate Prediction* yang berawal dengan masuknya *input user* ke proses *change input format to dataset*, proses ini mengubah *input user* yang berupa *string* menjadi *format dataset* yang digunakan oleh *weka library*. Setelah proses itu selesai maka akan dihasilkan *dataset input* yang dikirimkan ke proses *match input to decision tree* yang juga menerima *input*

berupa *decision tree* dari proses sebelumnya. Dalam proses ini penghitungan prediksi dilakukan dengan mencocokkan *input user* kedalam *rules-rules* yang dibuat oleh *decision tree*, setelah proses pencocokan tersebut selesai maka akan dihasilkan sebuah data yang berupa *predicition* dan akan dikirimkan ke *user* dan ditampilkan di layar aplikasi.



Gambar 8 *Data Flow Diagram Calculate Prediction*

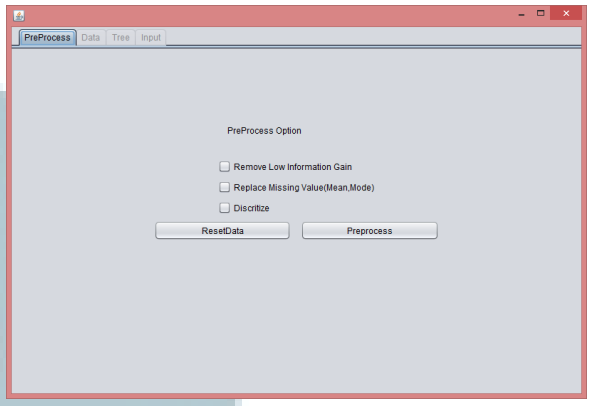
IV. IMPLEMENTASI APLIKASI DAN UJI DETEKSI

A. *Implementasi Algortima dan Metode Preprocess*

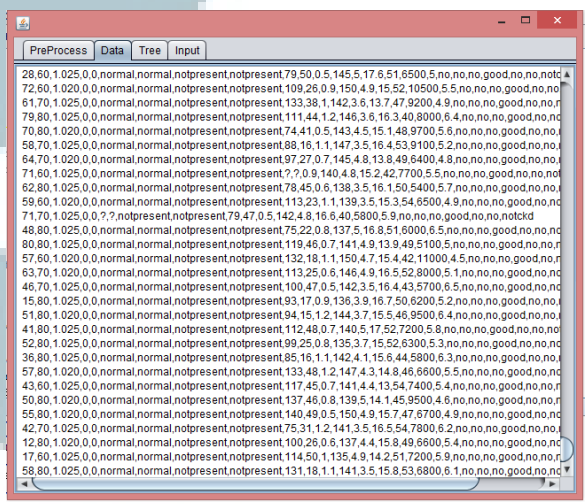
Algoritma C4.5 diimplementasi dengan bantuan dari *weka library*. Yaitu dengan menggunakan *class j48* yang berguna untuk membuat *pruned or unpruned* algoritma *C4.5* [12]. Sebelum proses implementasi dilakukan aplikasi terlebih dahulu melakukan pembacaan terhadap *dataset* yang ingin dijadikan sebagai *training data*.

Untuk dapat meningkatkan kualitas dari *dataset* tersebut maka diperlukan *data preprocessing*. Selain mengisi *missing value* teknik tersebut juga teknik yang akan digunakan dalam penelitian ini adalah *data reduction*, dalam proses ini *data attribute* yang memiliki nilai *information gain* dibawah 0.14 akan dibuang. *Information gain* digunakan sebagai tolak ukur dalam *data reduction* ini adalah dikarenakan setelah melakukan beberapa kali percobaan terhadap aplikasi, penghapusan *data attribute* yang memiliki nilai kecil tidak mempengaruhi *accuracy* yang dihasilkan oleh *decision tree*. Teknik dari *data transformation* yang digunakan pada penelitian ini adalah dengan melakukan *discretize* pada *attribute data* yang memiliki tipe *numeric*. Proses *discretize* dilakukan dengan cara mengubah data yang berbentuk *countinous* menjadi kategori dengan cara mencari *range value* dari data tersebut [10].

Gambar 9 menunjukkan tampilan aplikasi saat *user* membuka aplikasi. Tombol ResetData berguna untuk mengulang proses *data preprocessing* dengan cara membaca ulang *dataset*. Jika pilihan opsi *Remove Low Information Gain* dipilih maka aplikasi akan menseleksi data dengan *information gain* yang memiliki nilai dibawah dari 0.14 akan dibuang dan tidak masuk dalam perhitungan saat pembuatan *decision tree* dimulai. Opsi *replace missing value (mean, mode)* akan berfungsi untuk mengganti seluruh *missing value* yang ada di dalam *dataset* dengan menggunakan *mean* atau *modus* dari *attribute* tersebut. Jika *attribute* memiliki tipe *numeric* maka pengisian *missing value* akan diisi dengan nilai *mean* dari attribute tersebut, jika *attribute* memiliki tipe *nominal* maka pengisian *missing value* akan diisi dengan nilai *modus* dari attribute tersebut. Opsi *discretize* berfungsi untuk mengganti tipe data dari *attribute* yang memiliki tipe *numeric* menjadi *nominal* dengan cara membuat *range* dari nilai *attribute* tersebut.



Gambar 9 Tampilan Awal Aplikasi

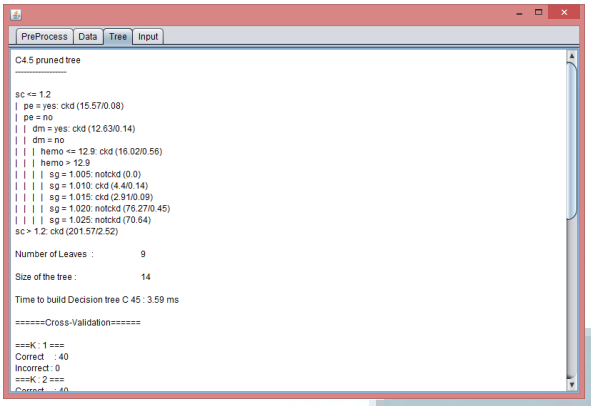


Gambar 10 Tampilan Aplikasi Pada Halaman Data

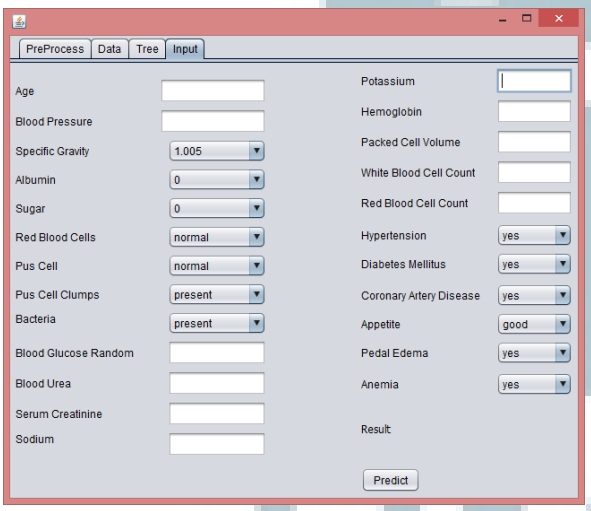
Gambar 10 menunjukkan tampillan aplikasi saat menu *tab* dipilih, pada saat ini aplikasi akan menampilkan seluruh data yang terdapat dalam *dataset* yang sudah dibaca sebelumnya.

Pada Gambar 11, ditampilkan beberapa laporan dari aplikasi terhadap *data preprocessing* yang dilakukan sebelumnya di halaman *preprocessing*, jika *user* tidak memilih opsi apapun saat halaman tersebut maka dalam halaman ini hanya akan ditampilkan struktur *decision tree* yang telah dibuat oleh aplikasi saja. Dalam halaman ini laporan mengenai pengujian *cross-*

validation yang dihitung berdasarkan data prediksi menggunakan *rules* yang dibuat dengan hasil yang sebenarnya yang terdapat di *dataset* ditunjukkan oleh aplikasi.



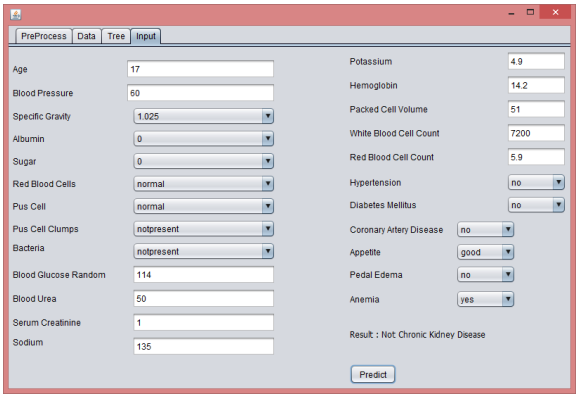
Gambar 11 *Screenshot* Halaman *Tree*



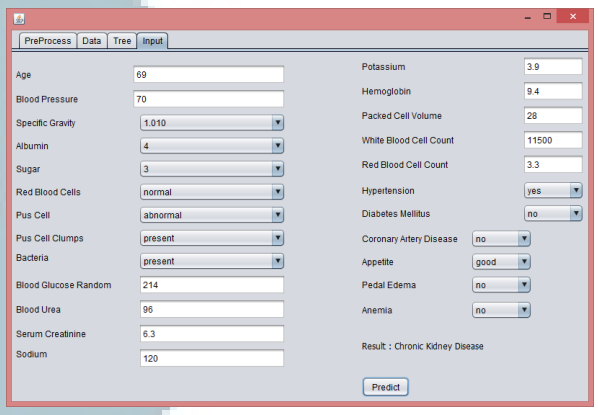
Gambar 12 *Screenshot* Halaman *Input*

Pada Gambar 12 menunjukkan tampilan aplikasi yang digunakan untuk melakukan percobaan deteksi penyakit ginjal kronis dengan menggunakan *decision tree* yang sudah dibuat sebelumnya. Beberapa dari *input* berbentuk *combo box* karena untuk mengisi *attribute* yang bertipe *nominal*, hal ini dikarenakan diperlukan penjagaan agar *user* tidak memasukkan nilai *value* yang tidak seharusnya dimasukkan dalam kotak *input* tersebut.

Pada Gambar 13 menunjukkan tampilan *input* yang sudah menampilkan hasil prediksi. Hasil prediksi pada gambar tersebut menunjukkan bahwa hasil prediksi adalah *Not Chronic Kidney Disease*. Yang berarti data yang dimasukan diprediksikan tidak mengidap penyakit ginjal kronis.



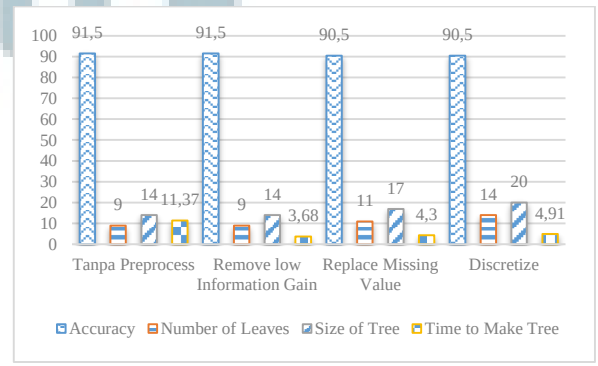
Gambar 13 *Screenshot* Halaman *Input* dengan Hasil Prediksi *Not Chronic Kidney Disease*



Gambar 14 *Screenshot* Halaman *Input* dengan Hasil Prediksi *Chronic Kidney Disease*

Pada Gambar 14 menunjukkan tampilan *input* yang sudah menampilkan hasil prediksi. Hasil prediksi pada gambar tersebut menunjukkan bahwa hasil prediksi adalah *Chronic Kidney Disease*. Yang berarti data yang dimasukan diprediksikan mengidap penyakit ginjal kronis.

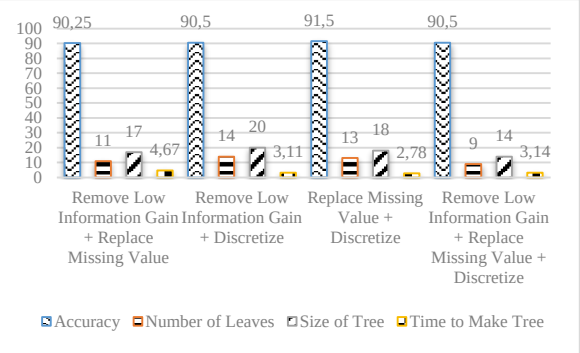
B. Uji Coba Program dan Analisa Preprocess



Gambar 15 *Bar Chart* Pertama Hasil Uji Coba

Gambar 15 menunjukkan tampilan *bar chart* dari hasil uji coba yang telah dilakukan sebelumnya. Pada

tampilan tersebut yang memiliki akurasi terbesar adalah percobaan dengan menggunakan *remove low information gain* dan tanpa menggunakan *preprocess*. Percobaan yang memiliki *number of leaves* dan *size of tree* terkecil adalah tanpa *preprocess* dan *remove low information gain*. Percobaan dengan waktu pembuatan *tree* tercepat adalah *remove low information gain*.



Gambar 16 *Bar Chart* Kedua Hasil Uji Coba

Gambar 16 menunjukkan tampilan *bar chart* dari hasil uji coba menggunakan dua kombinasi metode preproses yang telah dilakukan sebelumnya. Pada tampilan tersebut yang memiliki akurasi terbesar adalah percobaan dengan menggunakan kombinasi *replace missing value* dan *discretize*. Percobaan yang memiliki *number of leaves* dan *size of tree* terkecil adalah kombinasi *remove low information gain*, *replace missing value*, dan *discretize*. Percobaan dengan waktu pembuatan *tree* tercepat adalah kombinasi *replace missing value* dan *discretize*.

V. SIMPULAN

Rancang bangun aplikasi pendeteksi penyakit ginjal kronis dengan menggunakan algoritma C4.5 telah berhasil dibuat, dengan bantuan dari *weka library* dalam pembuatan algoritma *decision tree* yang dihasilkan dapat melakukan prediksi terhadap penyakit ginjal kronis. Uji coba dilakukan untuk mengetahui seberapa besar tingkat akurasi yang dapat dihasilkan oleh aplikasi. Proses pengujian dilakukan dengan menggunakan *cross-validation* dan berdasarkan hasil yang sudah dihitung aplikasi ini memiliki akurasi 91.50% pada saat *decision tree* dibuat tanpa menggunakan *preprocess menu*. Perkembangan yang didapatkan dalam melakukan preproses adalah pengurangan waktu yang diperlukan untuk membuat *decision tree*. Akurasi yang didapatkan setelah preproses tidak ada yang memiliki peningkatan. Ukuran *decision tree* bertambah besar jika menggunakan preproses *discretize* dan *replace missing value*. Metode yang meningkatkan cepatnya pembuatan *decision tree* adalah *remove low information gain*, *replace missing value*, dan *discretize*. Metode yang paling baik digunakan berdasarkan percobaan yang dilakukan adalah *remove low information gain*, yang menghasilkan akurasi 91.50 % dengan ukuran *tree* dan waktu pembuatan *tree* yang minimum.

DAFTAR PUSTAKA

[1] Aswano. 2014. *MAKALAH GAGAL GINJAL KRONIK*. Tersedia dalam: http://www.slideshare.net/septianraha/makalah-gagal-ginjal-kronik-37197034. Diakses 2 Maret 2016.

[2] Kidney Disease Statistics for the United States. 2016. National Institute of Diabetes and Digestive and Kidney Diseases. [Online]. Avalaible: https://www.niddk.nih.gov/health-information/health-statistics/kidney-disease

[3] Rohman, A. 2013. *Penerapan Algoritma C4.5 Berbasis Adaboost untuk Prediksi Penyakit Jantung*. Majalah Ilmiah Universitas Pandanaran Vol 11. No. 26.

[4] Puteri, N. A., Maharani, W., Suliiyo, M. D. 2013. *Prediksi Penyakit Jantung dengan Algoritma Classification and Regression Tree*. Telkom University

[5] Gorunescu, F. 2011. Data Mining: Concepts, Models and Techniques.

[6] Kusrini. 2007. Konsep dan Aplikasi Sistem Pendukung Keputusan. Yogyakarta: Penerbit Andi.

[7] Sangadah, N. dan Tri, S. P. R. 2012. *MAKALAH GAGAL GINJAL AKUT DAN KRONIS*. Tersedia dalam: http://www.scribd.com/doc/93094328/MAKALAH-GAGAL-GINJAL-KRONIS#scribd. Diakses 2 Maret 2016.

[8] Aisyah, J. 2011. *Karakteristik Penderita Gagal Ginjal Rawat Inap Di RS Haji Medan Tahun 2009* Tersedia dalam: http://repository.usu.ac.id/handle/123456789/24681. Diakses 2 Maret 2016.

[9] Ridwan, M., Suryono, H., Sarosa, M. 2013. Penerapan Data Mining Untuk Evaluasi Kinerja Akademik Mahasiswa Menggunakan Algoritma Naïve Bayes Classifier.

[10] Han J., Kamber M., Pei J. 2012. Data Mining: Concepts and Techniques 3rd Edition.

[11] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[12] Dr. Bhargava, N., Dr. Bhargava, R., Mathuria, M. 2013. *International Journal of Advanced Research in Computer Science and Software Engineering*..

[13] Sayad, S. 2010. *Decision Trees: Classification & Regression*. University of Toronto.

[14] Lichman, M. 2013. *{UCI} Machine Learning Repository*. [Online]. Available : http://archive.ics.uci.edu/ml