Tesis

Grammatical Evolution untuk Ekstraksi Fitur dengan Pengukuran *Multi Fitness*

Go Frendi Gunawan
NRP : 5111201033

DOSEN PEMBIMBING
Prof. Dr. Ir. Joko Lianto Buliali, Msc.

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI CERDAS VISUAL
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2013

GRAMMATICAL EVOLUTION UNTUK EKSTRAKSI FITUR DENGAN PENGUKURAN MULTI FITNESS

Nama mahasiswa : Go Frendi Gunawan
NRP mahasiswa : 5111201033
Pembimbing : Prof. Dr. Ir. Joko Lianto Buliali, M.Sc

ABSTRAK

Ekstraksi fitur merupakan salah satu topik yang cukup berpengaruh untuk menyelesaikan masalah klasifikasi. Sampai saat ini, tidak ada cara yang baku untuk menentukan fitur-fitur terbaik dari suatu data. Dalam tesis ini, akan dicoba suatu pendekatan grammatical evolution dengan pengukuran multi fitness guna memperoleh fitur-fitur terbaik dari sebuah data.

Grammatical evolution merupakan turunan dari algoritma genetik yang menggunakan context-free grammar terdefinisi guna mencipatakan suatu fungsi matematika.

Beberapa metode pengukuran fitness telah dicoba dalam penelitian, antara lain pengukuran fitness global, pengukuran fitness per kelas, pengukuran fitness dengan metode tatami, dan pengukuran fitness dengan metode Gavrilis.

Hasil penelitian menunjukkan bahwa metode tatami menunjukkan hasil yang baik pada data-data sintesis, yang mana untuk memisahkan satu kelas dengan kelas-kelas lain dibutuhkan fitur-fitur yang berbeda.

Kata Kunci: ekstraksi fitur, *grammatical evolution*, klasifikasi, multi-fitness.

GRAMMATICAL EVOLUTION FOR FEATURE EXTRACTION WITH MULTI FITNESS EVALUATION

Student Name : Go Frendi Gunawan
Student Identity Number : 5111201033
Supervisor : Prof. Dr. Ir. Joko Lianto Buliali, M.Sc

ABSTRACT

Feature Extraction is a significant topic in classification problem solving. Until now, there is no such a standard way to determine the best features of a data. In this thesis, grammatical evolution with multiple fitness evaluation approach will be used in order to extract best features of the data.

Grammatical evolution is a derivative of genetics algorithm. It used predefined grammar to generate a mathematics function.

Some fitness measurement methods have been tested, including global fitness measurement, per-class fitness measurement, tatami fitness measurement, and Gavrilis fitness measurement.

Experiment shows that tatami fitness measurement give a vary good result for synthesis data which need different features to separate different classes.

Keywords: feature extraction, grammatical evolution, classification, multi-fitness.

DAFTAR ISI

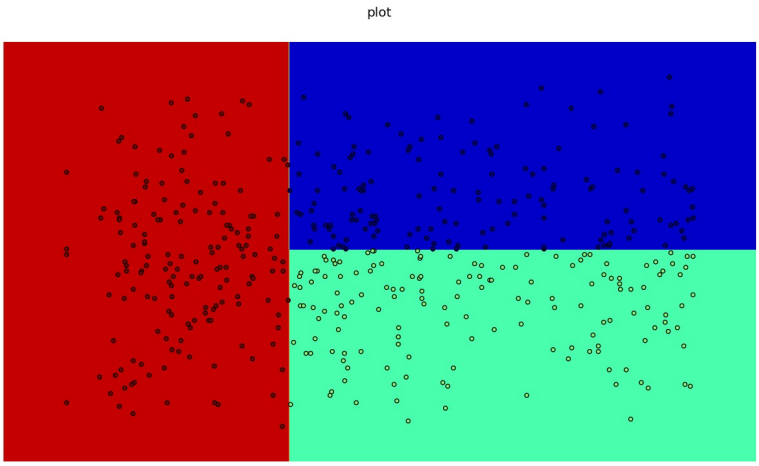
BAB 1
PENDAHULUAN

1.1 Latar Belakang

Ekstraksi fitur merupakan salah satu hal yang paling berpengaruh dalam pemecahan masalah klasifikasi. Pemilihan fitur yang tidak baik akan mengakibatkan kesulitan dalam memisahkan kelas-kelas data. Kegagalan pemisahan kelas-kelas data akan berdampak pada turunnya akurasi dalam proses klasifikasi.

Dalam penelitian sebelumnya (Gunawan, 2012), telah dicoba suatu pendekatan ekstraksi fitur dengan menggunakan *grammatical evolution*. Dalam penelitian tersebut, terdapat sebuah kelemahan dikarenakan hanya dilakukan 1 tolak ukur global untuk pengukuran *fitness value*. Hal ini mengakibatkan fitur-fitur yang sebenarnya cukup baik secara khusus, justru tersingkirkan karena nilai *fitness* globalnya rendah. Penelitian-penelitian lain seperti (Gavrilis, 2006; Gavrilis, 2008) juga menggunakan satu nilai *fitness* terhadap satu set fitur. (Guo, 2011; Li, 2011) juga melakukan hal yang hampir sama terhadap kasus yang berbeda.

Penelitian ini mengusulkan suatu cara baru dalam penilaian *fitness*. Penilaian *fitness* tersebut akan dilakukan dengan cara mengukur keterpisahan satu kelas terhadap kelas-kelas lain pada tiap dimensi. Metode tersebut, selanjutnya dinamakan *tatami* karena kemiripannya dengan bentuk lantai tradisional Jepang. Dalam metode ini, untuk memisahkan *n* buah kelas, maka dibutuhkan maksimal *n-1* buah fitur (dimensi).



Gambar 1.1 Pemisahan cluster dengan metode tatami

Pada gambar 1.1, terdapat tiga buah cluster yang masing-masing direpresentasikan dengan warna merah, biru dan cyan. Untuk memisahkan ketiga cluster tersebut dibutuhkan 2 buah fitur (dimensi). Dimensi horizontal bertugas untuk memisahkan cluster merah dan kedua cluster lain. Pada dimensi horizontal ini, cluster biru dan cyan tidak terpisahkan. Sedangkan pada dimensi vertikal, cluster cyan dan biru terpisahkan, walaupun kedua cluster tersebut tidak terpisah dari cluster merah. Dengan menggunakan kedua dimensi ini, maka akan terbentuk ruang fitur baru di mana cluster merah, biru dan cyan terpisah secara linear.

Pada akhir proses, diharapkan akan ditemukan sejumlah fitur terbaik, yang dapat membantu *classifier* (*decision tree*) untuk memisahkan kelas-kelas yang ada secara optimal.

1.2 Perumusan Masalah

Dalam penelitian ini, masalah-masalah yang akan diselesaikan dirumuskan sebagai berikut:

1. Bagaimana menentukan formula untuk mengukur nilai *fitness* dari sebuah fitur
2. Bagaimana menerapkan skenario pengukuran fitness untuk semua kelas

3. Bagaimana melakukan pengujian atas fitur-fitur yang sudah di *generate*

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. Data yang diproses adalah data numerik
2. Data yang diproses tidak memiliki *missing attribute*.
3. Grammar yang digunakan hanya meliputi operator dan fungsi-fungsi matematika umum (+, -, *, /, *exp*, *abs*, *sqr*, dan *sqrt*).

1.4 Tujuan Penelitian

Menghasilkan dan menguji suatu metode baru berbasis *grammatical evolution* untuk mengekstrasi fitur pada data numerik.

1.5 Manfaat Penelitian

Hasil penelitian dapat digunakan sebagai salah satu langkah *preprocessing* sebelum melakukan proses klasifikasi dengan menggunakan *decision tree*.

Dengan metode ekstraksi fitur dalam tahapan *preprocessing*, diharapkan proses klasifikasi data yang tidak memiliki korelasi langsung terhadap kelas dapat dilakukan dengan lebih baik.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan dibahas beberapa teori dasar yang menunjang dalam pembuatan Tesis.

2.1 Ekstraksi Fitur

Ekstraksi fitur adalah suatu proses untuk mencari transformasi atau pemetaan dari fitur-fitur original ke ruang fitur baru yang dapat memperbesar keterpisahan antar kelas (Guo, 2011).

Banyak peneliti menyetujui bahwa ekstraksi fitur adalah proses terpenting dan tersulit pada masalah pengenalan pola dan klasifikasi. Pemilihan fitur yang paling tepat, mungkin merupakan tugas tersulit dalam pengenalan pola (Micheli-Tzanakou, 2000). Ekstraksi fitur yang ideal akan menghasilkan sebuah representasi yang sangat memudahkan pekerjaan classifier (Duda, Hart, & Stork, 2001). Dalam banyak kasus, ekstraksi fitur dilakukan oleh manusia, berdasarkan pengetahuan atau pengalaman, bahkan intuisi para peneliti (Guo, 2011)

Adapun fitur-fitur hasil ekstraksi bisa dikatakan baik, jika berhasil memisahkan data berdasarkan kelas yang diharapkan dengan tingkat kesalahan sekecil mungkin.

Untuk menjelaskan tujuan dari ekstraksi fitur, pada tabel 2.1 ditampilkan contoh data numerik. Data tersebut terdiri dari 2 fitur original, yakni x dan y. Masing-masing baris dalam tabel digolongkan dalam 3 buah kelas, yakni A, B dan C.

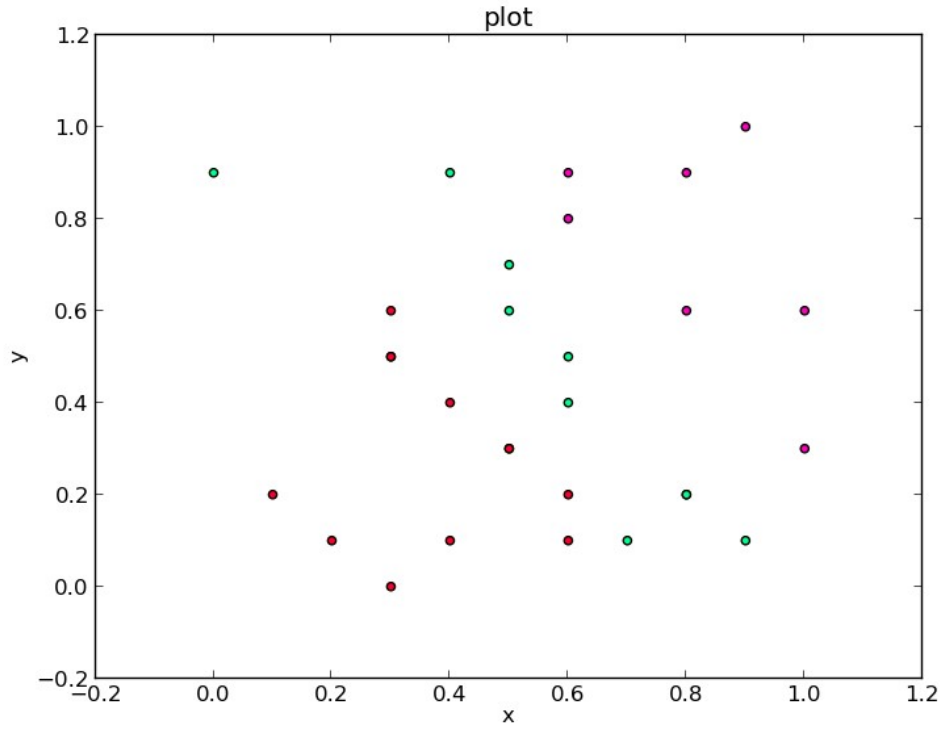
Tabel 2.1. Contoh Data numerik

Fitur original		kelas
x	**y**	
0.3	0.5	A
0.4	0.9	B
0.6	0.2	A

0.9	1.0	C
1.0	0.3	C
0.8	0.2	B
...	...	...

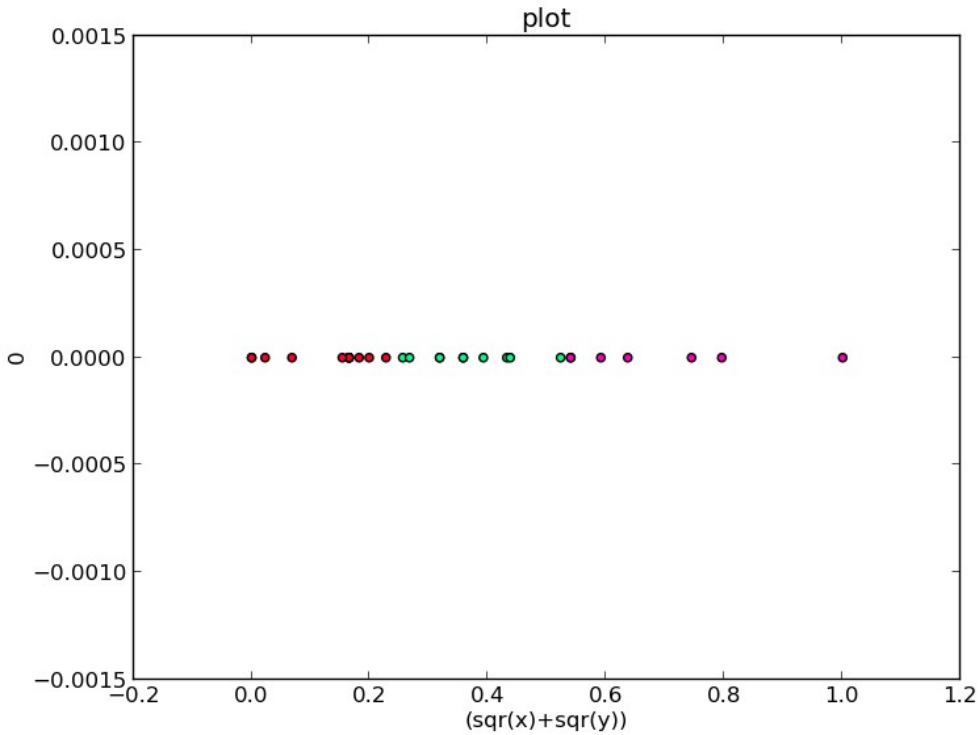
Jika data numerik pada tabel 2.1 dapat pula direpresentasikan dalam bentuk grafis seperti yang disajikan pada gambar 2.1 dengan fitur x sebagai dimensi horizontal, dan y sebagai dimensi vertikal, maka akan tampak bahwa penggunaan dimensi x dan y dapat menciptakan ruang fitur yang sanggup memisahkan kelas A, B dan C.

Adapun demikian, penggunaan dimensi x dan y secara terpisah akan mengakibatkan data-data pada kelas A,B dan C saling overlap (menempati posisi yang sama). Sebagai contoh, terdapat data dari kelas A, B dan C yang sama-sama memiliki nilai x = 0,6



Gambar 2.1 Representasi grafik dari Tabel 2.1

Proses ekstraksi fitur diharapkan mampu menciptakan sesedikit mungkin fitur yang dapat memisahkan kelas-kelas secara cukup baik. Fitur yang dihasilkan dari proses ekstraksi dapat merupakan suatu fungsi matematika yang menggunakan subset dari fitur-fitur original sebagai komponennya, semisal x^2+y^2. Gambar 2.2 menunjukkan bahwa penggunaan fitur x^2+y^2 ternyata telah cukup untuk memisahkan kelas A, B dan C.



Gambar 2.2 Representasi grafik data tabel 2.1 terhadap fitur x^2+y^2

2.2 Grammatical Evolution

Grammatical Evolution adalah pengembangan dari Genetics Programming (yang merupakan pengembangan dari algoritma genetika), yang merupakan suatu algoritma untuk mendapatkan satu set program dalam bahasa terntentu. Dengan memanfaatkan *context-free grammar* yang ditulis dalam *Backus Naur form*,

grammatical evolution mampu memisahkan representasi fenotip dan genotip dalam individu (Harper, Blair, 2006).

Pada *grammatical evolution*, sebuah individu memiliki dua buah representasi. Representasi yang pertama adalah representasi genotip, sedangkan representasi yang kedua adalah representasi fenotip.

Representasi genotip berupa sekumpulan angka sebagaimana layaknya pada algoritma genetika. Genotip dapat berupa angka biner maupun desimal. Representasi genotip pada grammatical evolution akan ditransformasikan menjadi representasi fenotip.

Representasi fenotip pada grammatical evolution dapat berupa fungsi matematika, kode program komputer, atau apapun, tergantung pada grammar yang digunakan.

Sama halnya seperti dalam algoritma genetika, pada grammatical evolution juga terdapat fitness function untuk mengukur kebaikan dari setiap individu. Untuk kasus ekstraksi fitur, umumnya tingkat akurasi classifier digunakan sebagai fitness function.

2.2.1. Grammar Pada Grammatical Evolution

Untuk mentransformasikan representasi genotip menjadi representasi fenotip dibutuhkan sebuah *grammar*. *Grammar* di sini sebenarnya mirip dengan *grammar* dalam bahasa natural. Hanya saja, direpresentasikan dalam bentuk *backus naur form* (BNF). Dalam sebuah *grammar* terdapat beberapa bagian penting, antara lain:

- T : Terminal set. Merupakan node-node yang sudah tidak mungkin dievolusikan
- N : Non-terminal set. Merupakan node-node yang masih mungkin dievolusikan
- P : Production rules. Merupakan keseluruhan *grammar*
- S :Start symbol. Merupakan salah satu anggota N yang digunakan sebagai node

Tabel 2.2. Contoh Grammar

Node Notation	Node	Aturan Produksi	Notasi Aturan
(A)	<expr>	<expr><op><expr>	(A1)
		<num>	(A2)
		<var>	(A3)
(B)	<op>	+	(B1)
		-	(B2)
		*	(B3)
		/	(B4)
(C)	<var>	x	(C1)
		y	(C2)
(D)	<num>	1	(D1)

Semisal, didefinisikan production rules (P) seperti pada tabel 2.2, maka +, -, *, /, x, y, 1 merupakan anggota dari himpuan Terminal Set (T). Node-node yang menjadi anggota T, merupakan node-node yang sudah tidak mungkin dapat dievolusikan. Sementara itu <expr>, <op>, <var>, <num> digolongkan sebagai Non-terminal Set (N). Node-node tersebut masih mungkin berevolusi menjadi node lain. Node <expr> berfungsi sebagai start symbol (S), artinya node <expr> merupakan node awal.

2.2.2. Transformasi Genotip ke Fenotip pada Grammatical Evolution

Transformasi genotip ke fenotip memanfaatkan grammar yang ada dan operasi modulo (sisa bagi) untuk memilih aturan transofrmasi.

Semisal terdapat representasi genotip 11.01.00.10.01, maka proses untuk mendapatkan fenotipnya dapat digambarkan secara lengkap pada tabel 2.3.

Tabel 2.3. Proses Transformasi Genotip ke Fenotip

Before	Gene	Rule	After Transformation
<expr>	11 -> 3	<expr><op><expr>	<expr><op><expr>
<expr>	01 -> 1	<num>	<num><op><expr>
<num>	-	1	1<op><expr>
<op>	00 -> 0	+	1+<expr>

<expr>	10 -> 2	<var>	1+<var>
<var>	01 -> 1	y	1+y

Proses transformasi diawali dengan start symbol (dalam hal ini <expr>). Selanjutnya diambil sebuah segmen dari genotiip (dalam hal ini 11). Segmen tersebut dapat pula dinyatakan dalam bilangan decimal (dalam hal ini 3). Node <expr> memiliki 3 kemungkinan perubahan (A0 : <expr><op><expr>, A1:<num>, dan A2:<var>). Untuk menentukan aturan mana yang akan digunakan, maka dilakukan operasi modulo (sisa bagi), di mana segmen genotip terpilih akan dibagi dengan jumlah kemungkinan evolusi. Karena 3 mod 3 = 0, maka dipilihlah aturan A0, yakni <expr><op><expr>. Proses ini dilanjutkan terus sampai seluruh node telah bertransformasi menjadi anggota terminal set (T).

Dalam contoh transformasi di tabel 2.3, diperoleh representasi fenotip dari 11.01.00.10.01 adalah 1+Y

2.3 Decision Tree

Decision Trees (DTs) merupakan salah satu metode yang umum digunakan untuk masalah regresi dan klasifikasi. Metode ini akan menghasilkan sebuah model yang dapat digunakan untuk menebak nilai variabel target. Model yang dihasilkan decision tree sebenarnya merupakan aturan inferensi sederhana yang didapat dari fitur-fitur yang ada (Pedregosa, dkk, 2011).

Decision tree memiliki kecenderungan overfit yang tinggi, selain itu korelasi tiap atribut terhadap keterpisahan kelas memegang peranan yang sangat penting. Karakteristik ini menyebabkan ekstraksi fitur menjadi salah satu hal yang berdampak pada akurasi decision tree.

BAB 3
METODE PENELITIAN

3.1 Langkah-Langkah Penelitiann

Pada penelitian ini, terdapat beberapa tahapan penyelesaian yang akan dilakukan, yang masing-masing tahapan menggunakan suatu metode tertentu. Adapun tahapan dan metode yang digunakan adalah sebagai berikut (gambar 3.1):

1. Studi literatur dan pencarian dataset

   Proses ini terdiri atas pencarian referensi-referensi pendukung yang sesuai, baik dari buku, jurnal, maupun artikel. Proses tersebut dilanjutkan dengan pencarian data-data numerik yang tersedia di internet sesuai dengan batasan permasalahan. Selain data-data umum, juga akan dibuat beberapa data sintesis yang dibuat dengan program *spreadsheet*.

2. Menyusun *grammar* dan rancang bangun sistem

   Proses ini terdiri atas perancangan formula *grammar* dan algoritma umum dalam proses ekstraksi fitur

3. Menyusun rancangan pengujian sistem

   Dalam proses ini ditentukan skenario pengujian. Pengujian yang dimaksud dapat berupa pembandingan hasil klasifikasi dengan ekstraksi fitur dalam penelitian ini, ekstraksi fitur dalam penelitian sebelumnya, dan tanpa ekstraksi fitur. Akurasi klasifikasi menggunakan classifier decision-tree akan dihitung sebagai perbandingan.

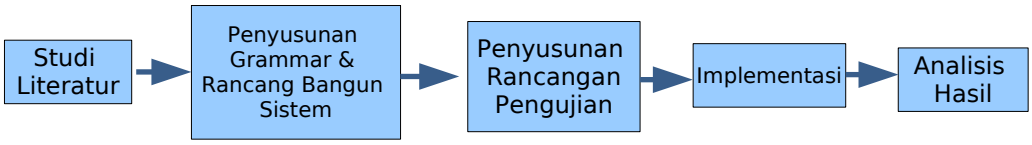
4. Mengimplementasikan sistem

   Sistem akan dibuat dalam bahasa pemrograman *Python* yang umum digunakan dalam kepentingan penelitian.

5. Menganalisis hasil yang diperoleh untuk menghasilkan kesimpulan

   Hasil ekstraksi fitur pada langkah nomor 4 akan diuji sesuai dengan rancangan pada langkah no 3. Selanjutnya akan disimpulkan apakah hasil

penelitian lebih baik dari penelitian sebelumnya. Jika tidak lebih baik, maka akan dianalisis penyebab kegagalannya.



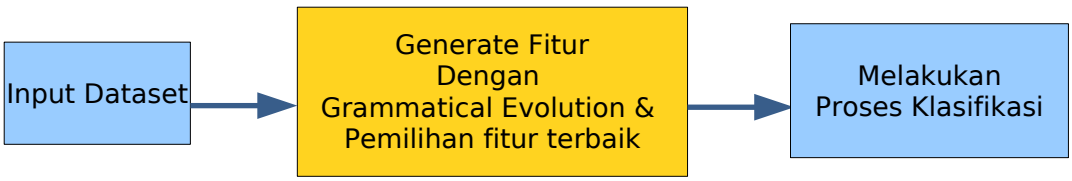
Gambar 3.1 Skema Metode Penelitian

3.2 Rancangan Sistem

Secara umum, algoritma yang akan digunakan adalah sebagai berikut:

Input dataset

1. Menggunakan *grammatical evolution* untuk mengekstraksi fitur
2. Generate genotip
    1. Transform genotip menjadi fenotip (fitur-fitur baru), sesuai dengan aturan *grammar* yang disediakan
    2. Hitung nilai *fitness* dari setiap fenotip (fitur-fitur baru) terhadap setiap kelas.
    3. Pilih fitur-fitur terbaik
3. Membangun *feature space* berdasarkan fitur-fitur terbaik, dan melakukan proses klasifikasi.



Gambar 3.2. Skema algoritma

3.2.1. Pembuatan Fitur

Proses pembuatan fitur dilakukan dengan menggunakan *grammatical evolution*. Proses ini ditujukan untuk membuat sebanyak mungkin calon fitur yang akan dinilai tingkat *fitness* nya.

Proses ini dimulai dengan pendefinisian *grammar*. *Grammar* yang telah didefinisikan, kemudian akan digunakan untuk mentransformasi sejumlah genotip yang dihasilkan secara random menjadi sejumlah fenotip. Setiap fenotip akan dihitung nilai *fitness* nya. Selanjutnya semua fenotip akan diurutkan berdasarkan nilai *fitness*. Detail tahapan yang diperlukan untuk pembuatan fitur adalah sebagai berikut:

3.2.1.1. Pendefinisian Grammar

Dalam metode grammatical evolution, pendefinisian grammar merupakan bagian yang cukup penting. Pendefinisian grammar akan menentukan berbagai kemungkinan terciptanya fenotip. Setiap fenotip yang tercipta akan menjadi fitur-fitur baru yang siap dievaluasi berdasarkan nilai fitness nya.

Grammar yang digunakan dalam penelitian ini adalah sebagai berikut (diimplementasikan dalam bahasa pemrograman Python):

```
self.variables = self.features
self.grammar = {
            '<expr>' : ['<var>', '<stmt>'],
            '<stmt>' : ['(<expr>) <op>
(<expr>)','<func>(<expr>)','sqrt(sqr(<expr>+<expr>)/2)'],
            '<var>'  : self.variables,
            '<op>'   : ['+','-','*','/'],
            '<func>' : ['exp','abs','sigmoid','sqr','sqrt','-']
        }
```

Gambar 3.3 Grammar yang digunakan

Variabel *self.variables* berisi fitur-fitur data original. Pada *self.grammar* didefinisikan bahwa <expr> dapat berevolusi menjadi <var>, (<expr>) <op> (<expr>), atau <func>(<expr>). Sedangkan <var> dapat berevolusi menjadi fitur-fitur original. Demikian pula dengan <op> yang dapat berevolusi menjadi operator-operator matematika dan <func> yang dapat berevolusi menjadi salah satu dari fungsi-fungsi matematika terdefinisi. Proses evolusi sendiri akan bermula dari node <expr>

3.2.1.2. Pembuatan Fitur

Proses pembuatan fitur tak lain adalah transformasi genotip (deretan angka acak yang telah digenerate) ke dalam bentuk fenotip menggunakan grammatical evolution dengan grammar terdefinisi. Proses ini telah dibahas dalam subbab 2.2.2. Dalam implementasinya, proses ini didefinisikan dalam sebuah fungsi yang mengembalikan fitur baru dalam tipe data string.

```
def _transform(self, gene):
    # this is a caching mechanism
    if gene in self.genotype_dictionary:
        return self.genotype_dictionary[gene]
    # maximum depth
    depth = 20
    gene_index = 0
    expr = self._start_node
    # for each level
    level = 0
    while level < depth:
        i=0
        new_expr = ''
        # parse every character in the expr
        while i<len(expr):
            found = False
            for key in self._grammar:
                # replace keyword with rule in production
                if (expr[i:i+len(key)] == key):
                    found = True
                    # count how many transformation possibility exists
                    possibility = len(self._grammar[key])
                    # binary digit needed to represent the possibilities
                    digit_needed = utils.bin_digit_needed(possibility)
                    # if end of gene, then start over from the beginning
                    if(gene_index+digit_needed)>len(gene):
                        gene_index = 0
                    # get part of gene that will be used
                    used_gene = gene[gene_index:gene_index+digit_needed]
                    gene_index = gene_index + digit_needed
                    rule_index = utils.bin_to_dec(used_gene)%possibility
                    new_expr += self._grammar[key][rule_index]
```

```
                    i+= len(key)-1
            if not found:
                new_expr += expr[i:i+1]
            i += 1
        expr = new_expr
        level = level+1
    # add to cache
    self.genotype_dictionary[gene] = expr
    return expr
```

Gambar 3.4 Fungsi Transformasi untuk Membuat Fitur

Fungsi _transform menerima parameter *gene* yang bertipe data string dan berisi deretan angka biner. Kemudian dengan menggunakan parameter *gene* dan grammar yang telah didefinisikan sebelumnya, digenerate sebuah fitur (fenotip) baru, Fenotip tersebut berupa string yang berisi potongan kode program dalam bahasa Python .

3.2.1.3. Normalisasi Proyeksi Data Berdasarkan Fitur Baru

Fitur yang telah di-generate pada subbab sebelumnya, selanjutnya digunakan untuk memproyeksikan data original. Proses ini didefinisikan dalam fungsi get_projection.

Untuk setiap record data yang ada, dilakukan proses evaluasi (didefinisikan pada utils.execute). Proses ini akan mengembalikan sebuah tupple yang berisi angka hasil evaluasi dan status error. Jika status error bernilai benar, maka ada kemungkinan bahwa hasil evaluasi tidak berupa angka (*Nan* atau *None*). Untuk meminimalisasi error hasil proyeksi, maka jika terjadi error, hasil evaluasi akan diasumsikan sebagai -1.

Selanjutnya, untuk semua data yang berhasil dievaluasi (tidak memunculkan error) akan dilakukan proses normalisasi. Proses normalisasi ini bertujuan untuk mengubah nilai minimum menjadi 0 dan nilai maksimum menjadi 1. Proses normalisasi ini bertujuan untuk memastikan bahwa setiap fitur memiliki *range* yang sama atau hampir sama.

Hasil proyeksi data direpresentasikan dalam bentuk *dictionary* dengan kelas sebagai *key*, dan list hasil proyeksi kelas tersebut sebagai *value*.

```
def get_projection(new_feature, old_features, all_data, used_data =
None, used_target = None):
    used_projection, all_result, all_error = [], [], []
    # get all result
    for data in all_data:
        result, error = utils.execute(new_feature, data, old_features)
        all_error.append(error)
        if error:
            all_result.append(None)
        else:
            all_result.append(result)
    all_is_none = True
    for i in all_result:
        if i is not None:
            all_is_none = False
            break
    if all_is_none:
        min_result = 0
        max_result = 1
    else:
        min_result = min(x for x in all_result if x is not None)
        max_result = max(x for x in all_result if x is not None)
    if max_result-min_result>0:
        result_range = max_result-min_result
    else:
        result_range = LIMIT_ZERO
    if used_data is None: # include all data
        for i in xrange(len(all_result)):
            if all_error[i]:
                all_result[i] = -1
            else:
                all_result[i] = (all_result[i]-min_result)/result_range
        used_projection = all_result
    else:
        used_result = []
        for i in xrange(len(used_data)):
```

```
            result, error = utils.execute(new_feature, used_data[i],
old_features)
            if error:
                used_result.append(-1)
            else:
                used_result.append((result-min_result)/result_range)
        used_projection = used_result

    # make is safer, only allow int and float to be the member of
used_projection
    for i in xrange(len(used_projection)):
        value = used_projection[i]
        if (not isinstance(value,float)) and (not
isinstance(value,int)):
            value = -1
        if math.isnan(value):
            value = -1
        used_projection[i] = round(value,2)

    if used_target is None:
        return used_projection

    group_projection = {}
    for i in xrange(len(used_projection)):
        group = used_target[i]
        if not group in group_projection:
            group_projection[group]=[]
        group_projection[group].append(used_projection[i])
    return group_projection
```

Gambar 3.5 Fungsi Proyeksi dan Normalisasi Data

3.2.2. Pemilihan Fitur Terbaik

Pemilihan fitur terbaik diperoleh dengan cara menghitung akurasi classifier berdasarkan fitur terpilih.

3.2.2.1. Metode GA Select

Dalam metode GA Select, akan dipilih subset dari fitur original yang paling mampu memisahkan kelas dalam data secara optimum. Penilaian fitness

dilakukan dengan memanfaatkan akurasi separator. Dalam implementasinya, untuk skenario ini digunakan algoritma genetika biasa.

Banyaknya fitur yang dapat digenerate dengan skenario ini berkisar antara nol sampai dengan jumlah fitur original.

3.2.2.2. Metode GE Global

Dalam metode GE global, akan dipilih sebuah fitur yang mampu memisahkan semua kelas secara cukup baik. Penilaian fitness dilakukan dengan cara mengukur keterpisahan data secara empiris. Metode GE Global merupakan implementasi dari penelitian sebelumnya (Gunawan, 2012)

Banyaknya fitur yang bisa digenerate dalam skenario ini adalah 1.

3.2.2.3. Metode GE Multi

Metode ini merupakan pengembangan dari GE Global. Dalam GE Multi , digunakan pengukuran multi fitness untuk memisahkan masing-masing kelas dengan keseluruhan kelas lain. Setiap individu dalam metode ini akan memiliki *n* buah nilai fitness, di mana *n* adalah jumlah kelas yang ada.

Banyaknya fitur yang bisa digenrate dalam skenario ini adalah sebanyak jumlah kelas yang ada.

3.2.2.4. Metode GE Tatami

Metode GE Tatami merupakan pengembangan dari GE Multi. Pengembangan tersebut berasal dari hipotesa bahwa jika sebuah kelas telah terpisah dari semua kelas lainnya, maka pada proses selanjutnya, kelas yang sudah terpisah tersebut bisa diabaikan. Dalam skenario ini, akan tercipta *n-1* buah fitur.

Jika terdapat *n* buah kelas, yang masing-masing disimbolkan dengan C1 sampai dengan Cn, maka akan dipilih satu kelas yang paling terpisah dari kelas-kelas lain. Kelas ini selanjutnya disimbolkan sebagai C*1.

Kemudian diekstrak fitur F1 yang bertugas untuk memisahkan C* dan C-C*1. Proses akan diulang dengan C-C*1 sebagai himpunan kelas yang baru. Di sini C*1 diabaikan, karena sudah terpisah dari kelas-kelas lain. Selanjutnya akan

dipilih C*2 yang baru, dari C-C*1. C*2 dan C-C*1 akan dipisahkan oleh fitur F2. Demikian seterusnya sampai C*n-1 dan fitur Fn-1.

Banyaknya fitur yang digenerate dalam skenario ini adalah sebanyak jumlah kelas -1.

3.2.2.5. Metode GE Gavrilis

Metode GE Gavrilis merupakan implementasi dari penelitian yang dilakukan oleh Gavrilis (Gavrilis, 2011). Dalam metode ini akan digenerate set-set fitur yang masing-masing diwakili oleh satu individu.

Pengukuran fitness dalam metode ini dilakukan dengan mengukur akurasi classifier secara empiris.

Jumlah fitur yang digenerate dalam GE Gavrilis akan berkisar antara nol sampai tak terhingga.

3.2.3. Pengukuran Performa Fitur

Sebagai classifier, digunakan *Decision Tree* yang merupakan salah satu algoritma umum dalam permasalahan klasifikasi. Semua metode yang telah dibahas pada subbab sebelumnya akan digunakan untuk mengenerate sekumpulan fitur baru. Fitur-fitur tersebut akan digunakan sebagai data baru bagi *Decision Tree*. Diharapkan metode GE Tatami akan memperoleh hasil yang lebih baik dibandingkan dengan metode-metode lain.

Dalam pengujian akan digunakan berbagai macam data. Selain data-data sintesis yang sengaja dibuat untuk menguji hipotesa, percobaan juga akan dilakukan pada dataset iris, e.coli, dan balanced-scale yang telah umum dipakai dalam penelitian-penelitian sejenis. Data-data non-sintesis yang digunakan didapatkan dari website UCI-Machine Learning (http://archive.ics.uci.edu/ml/)

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Percobaan diimplementasikan dengan menggunakan bahasa pemrograman Python 2.7 dan beberapa library eksternal. Adapun library eksternal yang digunakan adalah scipy, numpy, matplotlib dan scikit-learn.

Source code program dan hasil lengkap pengujian diletakkan di repository github. Repository tersebut berlisensi open-source dan bisa diakses secara publik di alamat https://github.com/goFrendiAsgard/feature-extractor dengan lisensi GNU, sehingga bebas dimodifikasi dan digunakan guna penelitian lebih lanjut.

Dalam percobaan yang dilakukan terdapat beberapa metode yang diujikan pada berbagai macam data. Setiap data diuji dengan menggunakan 5 fold cross validation.

4.1 Pengujian Terhadap Dataset Sintesis 01

Untuk kepentingan uji coba penelitian, maka dibuat beberapa buah dataset sintesis menggunakan aplikasi spreadsheet. Dalam penelitian ini digunakan libre-office.

Pada dataset sintesis 01, terdapat 3 buah kelas, yakni *defender*, *demon hunter* dan *wizard*. Ketiga kelas tersebut didapatkan dengan melakukan kalkulasi berdasarkan 4 fitur (*defense*, *attack*, *agility*, *stamina*). Keempat fitur yang ada bersifat random uniform dan memiliki range antara 0-10 dengan pembulatan satu angka di belakang koma.

Adapun Formula yang digunakan untuk menggolongkan kelas adalah sebagai berikut: *=IF(A2/B2 >= 1.4, "defender", IF(C2 >= D2, "demon hunter", "wizard"))*. Di mana A2, B2, C2 dan D2 masing-masing adalah atribut *defense*, *attack*, *agility*, dan *stamina*. Pemilihan angka-angka pada formula semata-mata

untuk membuat dataset *balanced* (memiliki jumlah data yang hampir sama untuk semua kelas)

Tabel 4.1 Pengujian Pada Dataset Sintesis 01

Experiment		**GA Select Feature**		**GE Global**		**GE Multi**		**GE Tatami Multi**		**GE Gavrilis**	
		Accuracy (%)	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**
Whole	**Train**	73.04	3	77.83	1	99.35	3	***100.0***	2	85.65	3
	Test	73.04		77.83		99.35		***100.0***		85.65	
	Total	73.04		77.83		99.35		***100.0***		85.65	
Fold 1	**Train**	75.95	3	78.92	1	**100.0**	3	***100.0***	2	84.59	61
	Test	67.78		35.56		76.67		*81.11*		**83.33**	
	Total	74.35		70.43		95.43		***96.3***		84.35	
Fold 2	**Train**	73.78	3	78.92	1	**100.0**	3	***100.0***	2	**100.0**	48
	Test	70.0		36.67		74.44		*65.56*		**80.0**	
	Total	73.04		70.65		95.0		*93.26*		**96.09**	
Fold 3	**Train**	71.62	3	77.57	1	**100.0**	3	***100.0***	2	85.14	3
	Test	75.56		25.56		**86.67**		***86.67***		**86.67**	
	Total	72.39		67.39		**97.39**		***97.39***		85.43	
Fold 4	**Train**	73.51	3	78.65	1	**100.0**	3	***100.0***	2	85.41	3
	Test	74.44		36.67		**77.78**		***77.78***		76.67	
	Total	73.7		70.43		**95.65**		***95.65***		83.7	
Fold 5	**Train**	75.68	3	81.08	1	**100.0**	3	***100.0***	2	87.03	2
	Testing	70.0		42.22		**94.44**		*82.22*		72.22	
	Total	74.57		73.48		**98.91**		*96.52*		84.13	

Hasil pengujian menunjukkan bahwa GE Multi dan GE Tatami memeberikan hasil yang cukup baik dengan jumlah fitur yang relatif sedikit. Pada fold 2, GE Gavrilis memperoleh akurasi tertinggi, namun memiliki jumlah fitur yang sangat banyak.

4.2 Pengujian Terhadap Dataset Sintesis 02

Data sintesis 02 memiliki struktur yang hampir sama dengan data sintesis 01. Pada dataset ini terdapat 4 fitur dan 4 kelas.

Formula yang digunakan pada dataset sintesis 02 adalah sebagai berikut: *=IF(A2/B2 >= 1.6, "defender", IF(C2/D2 >= 1.3, "demon hunter", IF(D2>B2, "monk", "wizard")))*. Di mana A2, B2, C2 dan D2 masing-masing adalah atribut *defense*, *attack*, *agility*, dan *stamina*.

Tabel 4.2 Pengujian Pada Dataset Sintesis 02

Experiment		**GA Select Feature**		**GE Global**		**GE Multi**		**GE Tatami Multi**		**GE Gavrilis**	
		Accuracy (%)	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**
Whole	**Train**	77.98	4	70.8	1	**100.0**	4	***100.0***	3	90.38	12
	Test	77.98		70.8		**100.0**		***100.0***		90.38	
	Total	77.98		70.8		**100.0**		***100.0***		90.38	
Fold 1	**Train**	78.66	4	73.17	1	99.39	4	***100.0***	3	89.84	12
	Test	**76.03**		33.06		46.28		*62.81*		72.73	
	Total	78.14		65.25		88.91		***92.66***		86.46	
Fold 2	**Train**	76.42	4	70.53	1	**100.0**	4	***100.0***	3	89.43	12
	Test	77.69		27.27		72.73		*74.38*		**85.95**	
	Total	76.67		61.99		94.62		***94.94***		88.74	
Fold 3	**Train**	79.67	4	71.75	1	99.39	3	***100.0***	3	90.04	12
	Test	68.6		34.71		64.46		***83.47***		68.6	
	Total	77.49		64.44		92.5		***96.74***		85.81	
Fold 4	**Train**	79.07	4	70.73	1	**100.0**	4	***100.0***	3	90.24	12
	Test	72.73		40.5		71.07		*61.16*		**80.17**	
	Total	77.81		64.76		**94.29**		*92.33*		88.25	
Fold 5	**Train**	78.05	4	71.75	1	100.0	4	***100.0***	3	86.99	12
	Test	73.55		32.23		83.47		***94.21***		70.25	
	Total	77.16		63.95		96.74		***98.86***		83.69	

Pada Data sintesis 02, GE Multi tidak lagi memberikan performa sebaik pada dataset sintesis 01. Hal tersebut disebabkan karena dengan semakin

banyaknya kelas, pemisahan dengan skenario one vs all akan menjadi semakin sulit.

4.3 Pengujian Terhadap Dataset Sintesis 03

Dataset sintesis 03 merupakan dataset ideal untuk GE Tatami. Dalam dataset ini terdapat 5 kelas, A, B, C, D dan E. Pembagian data ke dalam 5 kelas ditentukan berdasarkan 4 fitur utama m1, m2, m3, dan m4. Adapun keempat fitur utama tersebut tidak digunakan dalam dataset, melainkan disembunyikan menjadi 5 fitur (f1, f2, f3, f4 dan f5). Selain kelima fitur tersebut, terdapat pula 2 buah fitur noise (n1 dan n2).

Penentuan kelas menggunakan formula sebagai berikut: *=IF(A3<0.5, "A", IF(B3<0.5, "B", IF(C3<0.5, "C", IF(D3<0.5, "D","E"))))*. Di mana A3, B3, C3 dan D3 masing-masing adalah m1, m2, m3 dan m4.

Sementara itu, f1, f2, f3, f4 dan f5 ditentukan sebagai berikut:

- f1 diperoleh secara acak dengan formula =ROUND(RAND()*9.9+0.1,3)
- f2 diperoleh dengan menggunakan rumus f2 = f1/m1
- f3 diperoleh dengan menggunakan rumus f3 = f2/m2
- f4 diperoleh dengan menggunakan rumus f4 = m3/f1
- f5 diperoleh dengan menggunakan rumus f5 = f4/m4

Penggunaan rumus-rumus tersebut dimaksudkan untuk memberikan kondisi ideal di mana fitur pemisah asli dapat ditemukan menggunakan kombinasi matematis terhadap fitur-fitur yang tampak.

Hasil pengujian menunjukkan bahwa untuk dataset sintesis 03, GE Tatami menunjukkan hasil yang sangat baik.

Tabel 4.3 Pengujian Pada Dataset Sintesis 03

Experiment		GA Select Feature		GE Global		GE Multi		GE Tatami Multi		GE Gavrilis	
		Accuracy (%)	Features	Accuracy (%)	Features	Accuracy (%)	Features	Accuracy (%)	Features	Accuracy (%)	Features
Whole	**Train**	72.5	6	67.0	1	98.25	5	***100.0***	4	80.75	6
	Test	72.5		67.0		98.25		***100.0***		80.75	
	Total	72.5		67.0		98.25		***100.0***		80.75	
Fold 1	**Train**	74.45	6	67.91	1	99.07	5	***100.0***	4	82.55	6
	Test	26.58		27.85		45.57		***63.29***		32.91	
	Total	65.0		60.0		88.5		***92.75***		72.75	
Fold 2	**Train**	72.9	6	66.98	1	100.0	5	***100.0***	4	86.29	47
	Test	62.03		27.85		68.35		***94.94***		69.62	
	Total	70.75		59.25		93.75		***99.0***		83.0	
Fold 3	**Train**	71.34	6	69.47	1	99.69	5	***100.0***	4	83.18	6
	Test	29.11		24.05		55.7		***65.82***		46.84	
	Total	63.0		60.5		91.0		***93.25***		76.0	
Fold 4	**Train**	72.9	4	66.98	1	98.44	5	***100.0***	4	73.52	2
	Test	27.85		26.58		50.63		***78.48***		25.32	
	Total	64.0		59.0		89.0		***95.75***		64.0	
Fold 5	**Train**	72.27	6	68.85	1	99.38	5	***100.0***	4	85.67	47
	Test	25.32		48.1		59.49		***63.29***		49.37	
	Total	63.0		64.75		91.5		***92.75***		78.5	

4.4 Pengujian Terhadap Dataset Iris

Dataset iris merupakan dataset yang cukup banyak dipakai dalam penelitian. Data ini terdiri dari 3 kelas (Iris-Setosa, Iris Versicolor, dan Iris Virginica) serta 4 atribut fitur original (Sepal Length, Sepal Width, Petal Length, dan Petal Width). Dataset iris bersifat multi-variate, terdiri dari 150 data. Dataset iris dapat didownload dari website UCI Machine Learning dengan alamat (http://archive.ics.uci.edu/ml/datasets/Iris)

Hasil pengujian terhadap data iris menunjukkan hasil yang hampir seimbang untuk GE Global, GE Multi, GE Tatami dan GE Gavrilis. Namun

tampak bahwa GE Gavrilis memberikan hasil yang sedikit lebih unggul dibandingkan metode-metode lain.

Tabel 4.4 Pengujian Pada Dataset Iris

Experiment		GA Select Feature		GE Global		GE Multi		GE Tatami Multi		GE Gavrilis	
		Accuracy (%)	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**
Whole	**Train**	96.0	2	**98.67**	1	**98.67**	3	***98.67***	2	**98.67**	1
	Test	96.0		**98.67**		**98.67**		***98.67***		**98.67**	
	Total	96.0		**98.67**		**98.67**		***98.67***		**98.67**	
Fold 1	**Train**	96.67	2	**99.17**	1	**99.17**	3	*98.33*	2	**99.17**	1
	Test	86.67		**96.67**		**96.67**		***96.67***		**96.67**	
	Total	94.67		**98.67**		**98.67**		*98.0*		**98.67**	
Fold 2	**Train**	95.83	2	**100.0**	1	99.17	3	*99.17*	2	**100.0**	3
	Test	96.67		**96.67**		83.33		*66.67*		**96.67**	
	Total	96.0		**99.33**		96.0		*92.67*		**99.33**	
Fold 3	**Train**	96.67	2	98.33	1	98.33	3	***99.17***	2	98.33	1
	Test	93.33		76.67		**100.0**		***100.0***		**100.0**	
	Total	96.0		94.0		98.67		***99.33***		98.67	
Fold 4	**Train**	95.83	2	99.17	1	**99.17**	3	***99.17***	2	**99.17**	1
	Test	96.67		93.33		**96.67**		***96.67***		**96.67**	
	Total	96.0		98.0		**98.67**		***98.67***		**98.67**	
Fold 5	**Train**	96.67	2	98.33	1	98.33	3	***99.17***	2	**99.17**	3
	Testing	93.33		93.33		93.33		***96.67***		**96.67**	
	Total	96.0		97.33		97.33		***98.67***		**98.67**	

4.5 Pengujian Terhadap Dataset E-Coli

Data E-Coli merupakan dataset yang cukup banyak dipakai dalam penelitian. Data ini terdiri dari 7 atribut dan 6 kelas, yang terdiri dari 335 record. Dataset ini merupakan klasifikasi terhadap berbagai varian dari bakteri E-Coli.

Dataset E-Coli dapat didownload di website UCI Machine Learning dengan alamat (http://archive.ics.uci.edu/ml/datasets/Ecoli).

Berbeda dengan pengujian-pengujian sebelumnya, di sini justru GA Select menghasilkan akurasi yang paling tinggi dibandingkan keempat skenario lain. Adapun demikian, saat semua data digunakan untuk training sekaligus testing, GE Tatami tampak berhasil memberikan akurasi yang paling tinggi.

Tabel 4.5 Pengujian Pada Dataset E Coli

Experiment		**GA Select Feature**		**GE Global**		**GE Multi**		**GE Tatami Multi**		**GE Gavrilis**	
		Accuracy (%)	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**
Whole	**Train**	97.02	7	84.52	1	96.73	8	***97.62***	7	97.02	12
	Test	97.02		84.52		96.73		***97.62***		97.02	
	Total	97.02		84.52		96.73		***97.62***		97.02	
Fold 1	**Train**	97.42	6	87.45	1	**98.89**	8	*96.31*	7	97.79	12
	Test	**73.85**		58.46		53.85		*53.85*		69.23	
	Total	**92.86**		81.85		90.18		*88.1*		92.26	
Fold 2	**Train**	96.68	5	86.72	1	97.79	8	***98.52***	7	97.79	12
	Test	**78.46**		63.08		49.23		*1.54*		58.46	
	Total	**93.15**		82.14		88.39		*79.76*		90.18	
Fold 3	**Train**	97.79	7	89.3	1	**99.26**	8	*97.05*	7	98.52	26
	Test	70.77		53.85		**80.0**		*69.23*		53.85	
	Total	92.56		82.44		**95.54**		*91.67*		89.88	
Fold 4	**Train**	96.31	5	87.08	1	**98.52**	8	*97.42*	7	98.15	12
	Test	**73.85**		69.23		63.08		*43.08*		56.92	
	Total	**91.96**		83.63		91.67		*86.9*		90.18	
Fold 5	**Train**	97.05	4	86.72	1	98.15	8	*96.31*	7	**98.89**	18
	Test	**75.38**		61.54		38.46		*38.46*		67.69	
	Total	**92.86**		81.85		86.61		*85.12*		**92.86**	

4.6 Pengujian Terhadap Dataset Balanced-Scale

Dataset balanced-scale terdiri dari 625 data yang masing-masing terdiri dari 4 fitur dan 3 kelas. Dataset ini dibuat untuk tujuan pengujian psikologi dan dapat didownload pada website UCI Machine Learning dengan alamat (http://archive.ics.uci.edu/ml/datasets/Balance+Scale).

Hasil pengujian menunjukkan bahwa secara umum GE Multi lebih unggul dibandingkan semua metode lain.

Tabel 4.4 Pengujian Pada Dataset Balanced-Scale

Experiment		**GA Select Feature**		**GE Global**		**GE Multi**		**GE Tatami Multi**		**GE Gavrilis**	
		Accuracy (%)	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**	**Accuracy (%)**	**Features**
Whole	**Train**	70.88	3	84.8	1	***91.68***	1	***91.68***	2	82.56	9
	Test	70.88		84.8		***91.68***		***91.68***		82.56	
	Total	70.88		84.8		***91.68***		***91.68***		82.56	
Fold 1	**Train**	70.92	3	**100.0**	1	**100.0**	1	*92.03*	2	81.08	4
	Test	70.73		**91.87**		**91.87**		*85.37*		81.3	
	Total	70.88		**98.4**		**98.4**		*90.72*		81.12	
Fold 2	**Train**	71.91	3	85.46	1	92.23	1	***92.63***	2	83.86	9
	Test	66.67		69.92		**89.43**		*82.93*		78.86	
	Total	70.88		82.4		**91.68**		*90.72*		82.88	
Fold 3	**Train**	70.52	3	90.04	1	**99.0**	2	*92.03*	2	83.67	126
	Test	72.36		66.67		**85.37**		*71.54*		81.3	
	Total	70.88		85.44		**96.32**		*88.0*		83.2	
Fold 4	**Train**	72.51	3	86.65	1	**100.0**	1	*91.83*	2	82.27	2
	Test	68.29		78.05		73.98		***91.06***		77.24	
	Total	71.68		84.96		**94.88**		*91.68*		81.28	
Fold 5	**Train**	71.12	3	84.66	1	94.62	3	***100.0***	2	82.87	1
	Test	69.92		**82.93**		57.72		*66.67*		51.22	
	Total	70.88		84.32		87.36		***93.44***		76.64	

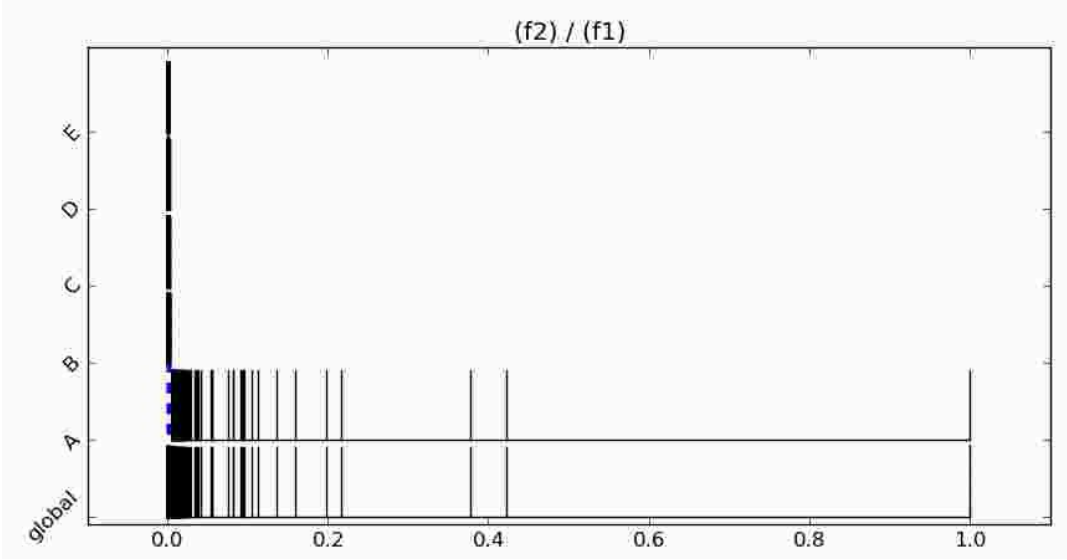
4.7 Pembahasan Karakteristik GE Tatami

Dari hasil percobaan, tampak bahwa GE Tatami menunjukkan hasil yang cukup baik pada data-data sintesis dan data iris. Dalam hal ini proses grammatical evolution berhasil menemukan fitur-fitur yang sanggup memisahkan data sesuai dengan hipotesis.

Pada dataset sintesis 03, tampak bahwa GE Tatami berhasil menemukan fitur-fitur yang sebanding dengan fitur asli (m1, m2, m3 dan m4) berdasarkan fitur-fitur tampak (f1, f2, f3, f4 dan f5). Hubungan antara fitur asli dan fitur tampak telah dijelaskan pada subbab 4.3. Adapun fitur-fitur yang berhasil di-generate oleh GE Tatami adalah sebagai berikut:

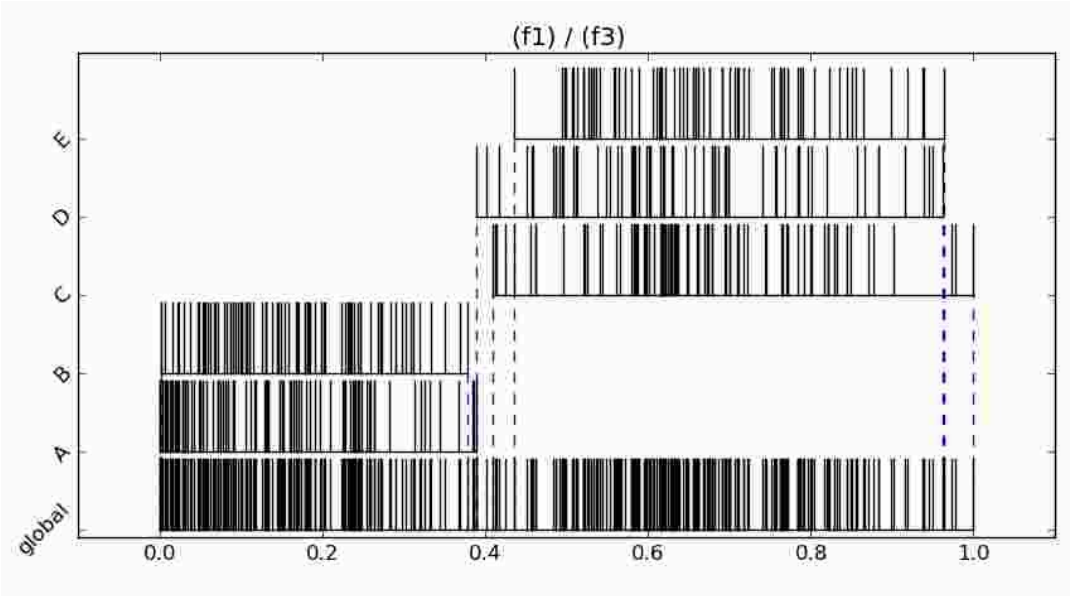
- (f2) / (f1)
- (f1) / (f3)
- sqrt(sqr(((f4) * (sqrt(sqr(f1+f1)/2))) - (n1)+n1)/2)
- (f5) / (f4)

Fitur (f2) / (f1) sanggup memisahkan kelas A dan keempat kelas lainnya. Hal ini tampak seperti pada gambar 4.1. Pada gambar 4.1, tampak bahwa kelas B, C, D dan E berimpit di sebelah kiri, terpisah secara linear dari kelas A. Panjang garis vertikal menunjukkan banyaknya data yang menempati nilai yang sama.



Gambar 4.1 Proyeksi Data terhadap fitur (f2) / (f1)

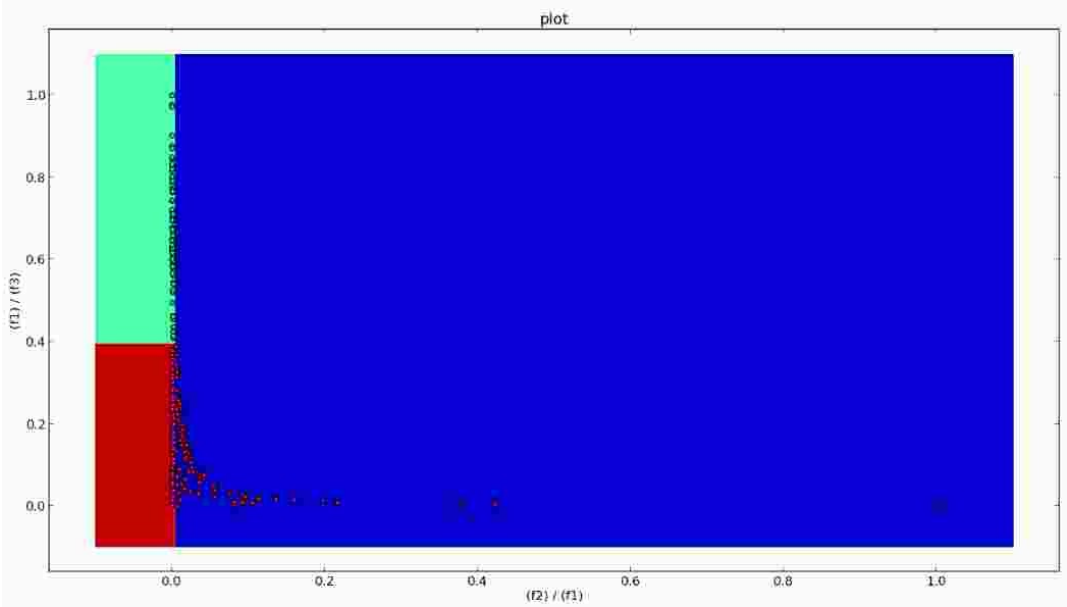
Setelah berhasil menemukan fitur (f2)/(f1) yang memisahkan kelas A dan keempat kelas lain (B, C, D, E), GE Tatami mencari fitur yang bisa memisahkan satu dari keempat kelas tersebut terhadap tiga kelas lainnya. Dalam proses ini ditemukan fitur (f1) / (f3). Proyeksi data ke dalam fitur (f1) / (f3) mengakibatkan kelas A tidak terpisah dengan kelas B. Namun hal ini tidak dipermasalahkan karena fitur sebelumnya, yakni (f2) / (f1) telah memisahkan kelas A dengan semua kelas lain.



Gambar 4.2 Proyeksi Data terhadap fitur (f1) / (f3)

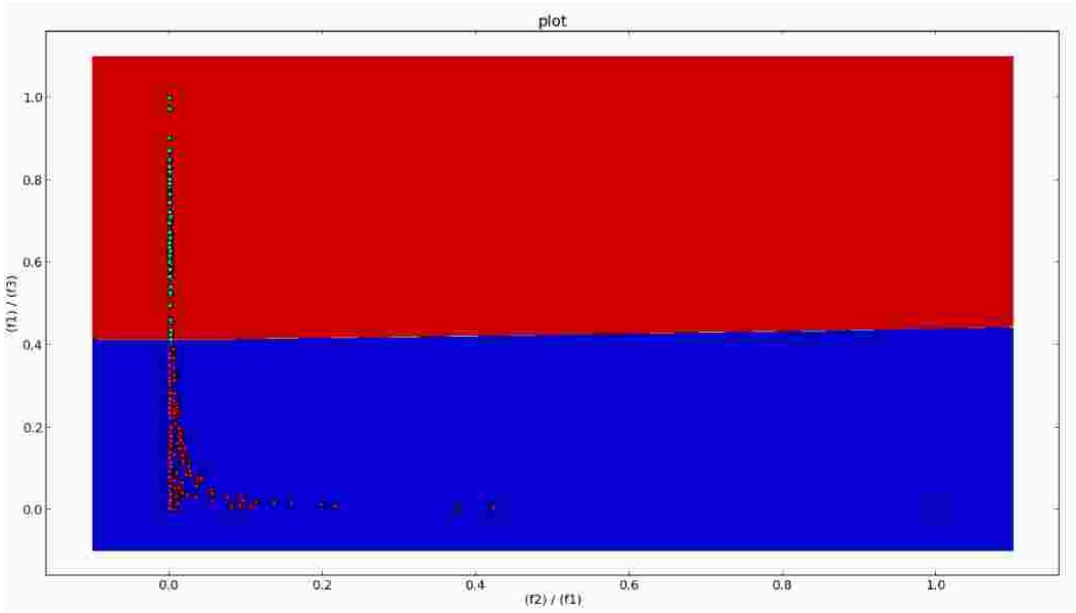
Proses yang sama juga berlaku untuk langkah selanjutnya, sehingga ditemukan fitur sqrt(sqr(((f4) * (sqrt(sqr(f1+f1)/2))) – (n1)+n1)/2) yang memisahkan C dengan D dan E, serta fitur (f5) / (f4) yang memisahkan D dan E.

Skenario GE Tatami tampak berhasil menciptakan feature space yang ideal bagi classifier Decision Tree, seperti ditujukkan pada gambar 4.3. Fitur (f2) / (f1) dan (f3) / (f1), memberikan kemudahan bagi decision tree untuk membagi data ke dalam



Gambar 4.3 Proyeksi Data terhadap Fitur (f2) / (f1) dan Fitur (f1) / (f3) dengan Classifier Decision Tree

Sekalipun GE Tatami menunjukkan hasil yang baik jika dikombinasikan dengan Decision Tree, namun tampaknya penggunaan classifier lain semisal SVM. Proyeksi data yang sama jika menggunakan classifier SVM ditunjukkan pada gambar 4.4



Gambar 4.3 Proyeksi Data terhadap Fitur (f2) / (f1) dan Fitur (f1) / (f3) dengan Classifier SVM

Fenomena ini menunjukkan bahwa Feature Space yang digenerate oleh GE Tatami hanya akan membantu untuk classifier yang memanfaatkan keterpisahan data per fitur tanpa mempedulikan pola dan bentuk sebaran data.

4.8 Pembahasan Kelemahan GE Tatami

Pada semua kasus yang ada, GE Tatami menunjukkan hasil terbaik saat semua data digunakan sebagai training dan testing sekaligus. Namun pada pengujian cross-validation didapati bahwa performa GE Tatami menurun. Terkadang (seperti dalam Fold-2 di dataset E-Coli), sekalipun akurasi yang diberikan dalam training sangat tinggi, akurasi testing nya sangat rendah. Ini menunjukkan bahwa GE Tatami cenderung membuat fitur-fitur yang overfit.

Kelemahan GE Tatami yang lain adalah jika terjadi kegagalan pada langkah pertama, maka langkah-langkah selanjutnya akan menjadi tidak berarti. Pada langkah pertama, GE Tatami harus berhasil menciptakan fitur yang memisahkan satu kelas tertentu dengan semua kelas lainnya secara cukup baik.

Jika hal ini gagal dilakukan, maka pada langkah selanjutnya kesalahan klasifikasi pada langkah awal akan sangat berpengaruh pada keseluruhan proses klasifikasi.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Hasil penelitian menunjukkan bahwa skenario GE Tatami berhasil membuat fitur-fitur yang membantu dalam proses klasifikasi untuk data sintesis dan iris. Namun metode tersebut gagal untuk menentukan fitur-fitur terbaik pada data ecoli. GE Tatami akan menghasilkan akurasi yang bagus jika ada proyeksi terhadap suatu fitur ter-generate yang menunjukkan keterpisahan menonjol antara satu kelas dengan semua kelas lainnya.

GE Tatami memberikan persyaratan yang lebih mudah dipenuhi daripada GE Global, GE Local, dan GE Multi. Pada GE Global, harus tergenerate sebuah fitur yang sanggup memisahkan setiap kelas. Pada GE Local, harus tergenerate n buah fitur yang sanggup memisahkan n kelas dengan kelas-kelas lain. Sementara pada GE Tatami, jika ada satu kelas yang sudah berhasil dipisahkan dari kelas-kelas lain, maka untuk pencarian fitur berikutnya, kelas tersebut dapat diabaikan.

Walaupun GE Tatami memberikan persyaratan yang lebih mudah dipenuhi, namun kompleksitas yang diberikan lebih tinggi dari metode-metode lain. Hal ini dikarenakan GE Tatami perlu melakukan perhitungan ulang sebanyak jumlah kelas-1 kali.

GE Tatami juga menunjukkan kegagalan saat tidak berhasil digenerate fitur yang memisahkan satu kelas dengan kelas-kelas lain secara cukup menonjol. Selain itu, GE Tatami juga memiliki kecenderungan untuk membuat fitur space yang overfit.

5.2 Saran

Hasil pengujian menunjukkan bahwa GE Tatami dan GE Multi memberikan hasil yang cukup baik. Dalam kasus-kasus ideal, GE Tatami tampak sanggup memberikan hasil yang sangat baik, namun dalam kasus-kasus lain, tampak bahwa GE Tatami menunjukkan hasil yang kurang baik. Sesuai dengan analisa yang dilakukan, kelemahan ini terjadi karena di langkah pertama, GE Tatami gagal memisahkan satu kelas dengan semua kelas lain. Hal ini selanjutnya berdampak pada proses selanjutnya. Oleh sebab itu, disarankan untuk menggabungkan fitur-fitur yang digenerate oleh GE Multi dan fitur-fitur yang digenerate oleh GE Tatami.

Untuk penggunaan GE Tatami dalam kasus nyata, sebaiknya digunakan sample yang cukup banyak dikarenakan kecenderungannya untuk membuat fitur-fitur yang overfit.

DAFTAR PUSTAKA

[1] Gunawan G. F., Gosaria S, Arifin A. Z. (2012). "*Grammatical Evolution For Feature Extraction In Local Thresholding Problem*", Jurnal Ilmu Komputer dan Informasi, Vol 5, No 2 (2012)

[2] Harper R., Blair A. (2006). "*Dynamically Define Functions in Grammatical Evolution*", IEEE Congress of Evolutionary Computation, July 16-21, 2006

[3] Gavrilis D., Tsoulous I. G., Georgoulas G., Glavas E. (2005). "*Classification of Fetal Heart Rate Using Grammatical Evolution*", IEEE Workshop on Signal Processing Systems Design and Implementation, 2005.

[4] Gavrilis D., Tsoulous I. G., Dermatas E. (2008). "*Selecting and Constructing Features Using Grammatical Evolution*", Journal Pattern Recognition Letters Volume 29 Issue 9, July, 2008 Pages 1358-1365 .

[5] Guo L., Rivero D., Dorado J., Munteanu C. R., Pazos A. (2011). "*Automatic feature extraction using genetic programming: An application to epileptic EEG classification* ", Expert Systems with Applications 38 Pages 10425-10436

[6] Li B., Zhang P.Y., Tian H., Mi S.S., Liu D.S., Ruo G.Q. (2011). "*A new feature extraction and selection scheme for hybrid fault diagnosis of gearbox*", Expert Systems with Applications 38 Pages 10000-10009

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). "*Scikit-learn: Machine Learning*

in Python", Journal of Machine Learning Research Vol. 12 Pages 2825-2830