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 grammar terdefinisi guna mencipatakan suatu fungsi matematika. Beberapa skenario pengukuran fitness telah dicoba dalam penelitian, dan skenario yang diusulkan memberikan hasil yang sangat baik pada data dengan aturan tertentu.

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 scenario has been tested, and the proposed method shows a very good result for data with certain rules.

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

DAFTAR ISI

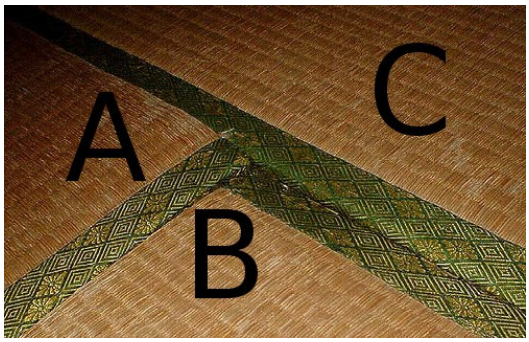
BAB 1
PENDAHULUAN

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 et al, 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 Gravilis et al, 2006 dan Gravilis et al, 2008 juga menggunakan satu nilai *fitness*. Guo et al, 2011 dan Li et al, 2011 menggunakan *classifier* sebagai bagian dalam pengukuran *fitneess*.

Dalam penelitian ini, akan diusulkan suatu standar baru dalam penilaian *fitness*. Penilaian *fitness* tersebut akan dilakukan dengan cara mengukur keterpisahan satu kelas terhadap kelas-kelas lain. Skenario tersebut, selanjutnya dinamakan *tatami-scenario*. Ide dasarnya berasal dari bentuk lantai tradisional jepang yang memisahkan satu bagian dengan bagian lain (diilustrasikan pada gambar 1.1).



Gambar 1.1 Tatami

Seandainya terdapat n buah kelas, maka dibutuhkan n-1 buah fitur untuk memisahkan n buah kelas tersebut. Dalam gambar 1.1, kelas A, B, dan C dapat dipisahkan hanya dengan menggunakan 2 garis.

Pada akhir proses, diharapkan akan ditemukan sejumlah fitur terbaik, yang dapat 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 harus lengkap dan bisa dipisahkan per kelas.
3. Grammar yang digunakan hnaya meliputi fungsi-fungsi matematika umum.

1.4 Tujuan Penelitian

Menciptakan dan menguji suatu metode baru dengan menggunakan prinsip *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 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 merupakan sebuah teknik untuk memilih fitur-fitur terbaik yang bisa digunakan untuk mengklasifikasikan data secara paling sederhana.

Di dalam proses ektraksi fitur, terkadang dimunculkan fitur-fitur baru berdasarkan fitur-fitur original. Proses tersebut sering pula dikenal dengan sebutan feature construction. Namun ada kalanya, hanya digunakan sebagian dari fitur-fitur original, atau biasa disebut sebagai feature selection. Dalam berbagai kasus, seringkali keduanya dilakukan bersamaan, sehingga muncul fitur-fitur baru yang berdampingan dengan beberapa fitur original.

Adapun fitur-fitur hasil ekstraksi bisa dikatakan baik, jika berhasil memisahkan data berdasarkan kelas yang diharapkan dengan tingkat kesalahan sekecil mungkin. Walau demikian, tingkat kompleksitas masing-masing fitur dan tingkat kompleksitas operasi untuk melakukan pemisahan data juga perlu dipertimbangkan. Kerumitan pada saat pembuatan fitur baru maupun pada saat operasi klasifikasi akan berpengaruh buruk pada performa proses klasifikasi itu sendiri.

Untuk menggambarkan tujuan ekstraksi fitur secara lebih jelas, maka disediakan contoh data numerik pada tabel 2.1. Jika digambarkan dalam bentuk cartesian dengan fitur x dan fitur y sebagai aksis dan ordinat, maka kelas A, B dan C tidak bisa dipisahkan secara linear. Pengklasifikasian data seperti ini dengan menggunakan neural network misalnya, akan membutuhkan banyak hidden neuron yang berujung pada peningkatan kompleksitas proses klasifikasi.

Tabel 2.1. Contoh Data numerik

Fitur original		kelas
x	y	
0	0	A
1	1	A
-1	-1	A
1	-1	A
-1	1	A
2	2	B
-2	-2	B
-2	2	B
2	-2	B
3	3	C
-3	-3	C
3	-3	C
-3	3	C

Proses ektraksi fitur diharapkan dapat menyederhanakan proses klasifikasi. Proses ektraksi fitur dapat saja menghasilkan sebuah fitur tunggal x^2 + y^2 yang sanggup memisahkan data secara linear. Fitur tersebut cukup baik, namun bukan yang terbaik, karena memiliki kompleksitas yang cukup tinggi jika dibandingkan dengan fitur-fitur original.

Salah satu alternatif yang lebih baik adalah dengan menggunakan abs(x) +abs(y). Fitur ini memiliki kompleksitas yang lebih rendah daripada fitur sebelumnya, namun tetap mampu memisahkan data secara linear.

2.2 Grammatical Evolution

Dikarenakan setiap data akan memiliki karakteristik yang berbeda-beda. Oleh sebab itu, tidak ada rumus baku untuk menghasilkan fitur baru secara umum. Cara yang cukup masuk akal untuk menghasilkan fitur-fitur semacam itu adalah dengan *trial and error*.

Algoritma genetika dan turunan-turunannya terbukti cukup baik dalam men-simulasikan trial and error yang dilakukan oleh manusia. *Grammatical evolution* merupakan salah satu turunan algoritma genetika yang memiliki context

free *grammar*, sehingga sangat cocok digunakan untuk mengekstraksi fitur dari data-data numerik.

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

Representasi genotip di sini berupa sekumpulan angka sebagaimana layaknya pada algoritma genetika. Di sini representasi genotip akan digunakan untuk mengevolusikan sebuah start symbol pada sebuah *grammar* untuk menjadi sebuah kalimat. Kalimat tersebutlah yang menjadi representasi fenotip individu.

Representasi genotip pada *grammatical evolution* tidaklah berbeda dengan representasi individu pada algoritma genetika, yakni berupa sekumpulan angka. Angka yang dimaksud bisa berupa angka biner maupun desimal. Contoh representasi genotip pada *grammatical evolution* dalam bentuk biner adalah 11001001.

Representasi genotip perlu diubah ke dalam bentuk fenotip dengan menggunakan *grammar*. Representasi fenotip pada *grammatical evolution* dapat berupa fungsi matematika ataupun sebuah program komputer, tergantung pada *grammar* yang didefinisikan. Contoh representasi fenotip yang berupa fungsi matematika adalah X+1, (X+3)*2 dan seterusnya.

Untuk proses penentuan *fitness* value, yang digunakan adalah representasi fenotip, sedangkan untuk operasi genetika, yang digunakan adalah representasi genotip.

2.2.1. Grammar Pada Grammatical Evolution

Seperti yang telah disebutkan, bahwa 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:

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)

- 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

Semisal, didefinisikan production rules (P) seperti pada tabel 2.2, maka T={+, -, *, /, x, y, 1}. Node-node yang menjadi anggota T, sudah tidak mungkin dapat dievolusikan. Sedangkan yang dimaksud N adalah {<expr>, <op>, <var>, <num>}. Node-node tersebut masih mungkin berevolusi. Semisal node <op>, dapat berevolusi menjadi +, -, *, ataupun /. Sementara itu, yang menjadi start symbol (S) adalah <expr>. Jadi jika ada sebuah genotip yang akan dicari representasi fenotipnya, maka akan digunakan node <expr> sebagai node awal.

2.2.2. Transformasi Genotip ke Fenotip pada Grammatical Evolution

Seandainya kita memiliki sederetan angka representasi genotip dalam bentuk biner 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

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.

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

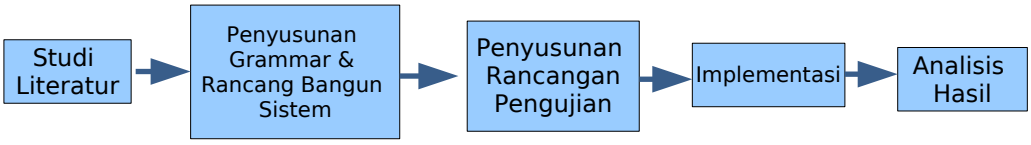
Dalam proses ini ditentukan skenario pengujian. Pengujian yang dimaksud dapat berupa pembandingan hasil klasifikasi dengan ekstraksi fitur dalam penelitian ini, ekstraksi fitur dalam penelitian sebelumnya, dan tanpa ekstraksi fitur. Performa klasifikasi (seperti kompleksitas) akan turut dihitung. Untuk proses klasifikasi sendiri akan digunakan soft SVM

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.



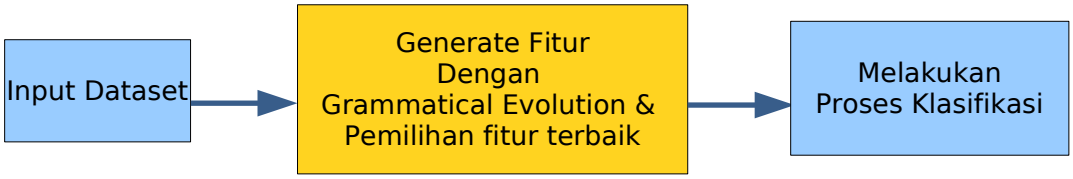
Gambar 3.1 Skema Metode Penelitian

3.2 Rancangan Sistem

Secara umum, algoritma yang akan digunakan adalah sebagai berikut:

Input dataset

1. Melakukan *grammatical evolution* untuk menciptakan fitur-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>','(<expr>) <op> (<expr>)','<func>(<expr>)'],
    '<var>'  : self.variables,
    '<op>'   : ['+','-','*','/'],
    '<func>' : ['exp','sigmoid','abs','sin','cos','sqr','sqrt']
}
self.start_node = '<expr>'
```

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

Secara umum, sebuah fitur akan dikatakan baik apabila:

- Proyeksi data terhadap fitur tidak tumpang tindih antara kelas-kelas yang berbeda
- Dalam proyeksi data terhadap fitur tidak ada kelas lain di antara kelas yang diharapkan

- Jarak intra-class dari kelas yang diharapkan cukup kecil, sedangkan jarak inter-class terhadap kelas-kelas yang tidak diharapkan cukup besar.
- Fitur relatif tidak kompleks

Ada dua cara yang akan digunakan untuk menilai *goodness* dari sebuah fitur. Cara pertama adalah dengan menggunakan akurasi classifier. Cara kedua adalah dengan mengukur keterpisahan data secara empiris. Dalam penelitian ini, kedua cara tersebut digunakan. Adapun pengukuruan keterpisahan data secara empiris dilakukan dengan menggunakan rumus sebagai berikut:

```
fitness = (SI + 2*(1-IP) + 3*(1-CP))/6
```

Di mana:

SI = Separability Index

IP = Intrusion Proportion

CP = Collisiion Proportion

Collision Proportion merupakan perbandingan antara jumlah data dalam satu kelas yang tidak terpisahkan dengan data dari kelas lain dengan keseluruhan data

Intrusion Proportion merupakan perbandingan antara jumlah data dalam satu kelas yang berada di antara nilai minimum dan maksimum dari kelas lain dengan keseluruhan data.

3.2.2.1. Skenario Feature Selection (GA_Select)

Dalam skenario ini, 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 0 sampai dengan jumlah fitur original. Nol menyatakan sama sekali tidak ada fitur yang terpilih.

3.2.2.2. Skenario Global Separator (GE_Global)

Dalam skenario global, akan dipilih sebuah fitur yang mampu memisahkan semua kelas secara cukup baik. Penilaian fitness dilakukan dengan cara mengukur keterpisahan data secara empiris.

Banyaknya fitur yang bisa digenerate dalam skenario ini adalah 1.

3.2.2.3. Skenario Local Separator (GE_Local)

Dalam skenario lokal, akan dipilih *n* fitur yang mampu memisahkan *n* kelas secara cukup baik. Semisal C adalah himpunan kelas yang ada, maka kelas pertama disimbolkan sebagai C1, kelas ke dua disimbolkan sebagai C2, dan kelas terakhir disimbolkan sebagai Cn. Dalam kasus ini, akan dibuat n buah fitur yang akan memisahkan setiap kelas dengan komplemen nya.

Maka untuk n buah kelas, akan dibuat n buah fitur. F1 akan bertugas memisahkan C1 dengan kelas-kelas lain. F2 akan bertugas memisahkan C2 dengan kelas-kelas lain, demikian seterusnya. Penilaian fitness dalam skenario ini dilakukan dengan cara mengukur keterpisahan data secara empiris.

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

3.2.2.4. Skenario Multi Accuration (GE_Multi)

Skenario ini merupakan variasi dari skenario local separator. Yang berbeda dari skenario ini adalah cara menilai fitness. Pada skenario, digunakan akurasi classifier sebagai penentu fitness.

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

3.2.2.5. Skenario Tatami Local Separator (GE_Tatami)

Skenario ini merupakan pengembangan dari skenario local separator. 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.

Secara konsep, 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.

Inspirasi skenario ini berasal dari bentuk lantai tradisional Jepang yang dikenal dengan sebutan tatami. Adapun penentuan fitness dalam skenario ini menggunakan pengukuran keterpisahan data secara empiris.

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

3.2.3. Pengukuran Performa Fitur

Untuk klasifikasi, akan digunakan classifier *Naive-Bayes* yang sudah umum digunakan. Dalam proses ini, akan dilakukan berbagai skenario pembandingan sesuai dengan yang telah dibahas pada subbab sebelumnya. Diharapkan skenario tatami akan memperoleh hasil yang lebih baik dibandingkan dengan skenario-skenario lain.

Pengujian dilakukan terhadap berbagai macam data. Selain data sintesis yang sengaja dibuat untuk menguji hipotesa, percobaan juga akan dilakukan pada data iris dan data e.coli yang telah umum dipakai dalam penelitian-penelitian sejenis.

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Percobaan skenario 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 penelitian 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 skenario yang diujikan pada berbagai macam data.

4.1 Pengujian Terhadap Data Sintesis 01

Untuk kepentingan uji coba penelitian, maka dibuat sebuah data sintesis. Data ini dapat dibuat dengan menggunakan aplikasi-aplikasi spreadsheet yang umum digunakan. Dalam penelitian ini digunakan libre-office. Pada data yang digenerate, diberlakukan penentuan kelas sesuai dengan formula percabangan yang dibuat secara khusus.

Dalam data sintesis ini ada 4 atribut antara lain attack, defense, agility dan stamina. Setiap atribut memiliki pola sebaran random uniform. Berdasarkan keempat atribut tersebut, data dibagi menjadi 3 buah kelas dengan aturan sebagai berikut:

- Kelas defender jika attack/defense >= 1.2
- Kelas demon_hunter jika attack/defense<1.2 dan agility>=stamina
- Kelas wizard jika attack/defense<1.2 dan agility<stamina

Pengujian terhadap data ini dilakukan dengan menggunakan 5 skenario pengekstrasian fitur yang telah dibahas pada subbab 3.3.2.

Percobaan dilakukan dua kali. Percobaan pertama dilakukan dengan melibatkan keseluruhan dataset sebagai training sekaligus testing. Percobaan kedua melibatkan skenario folding. Dalam skenario folding digunakan 5 fold dengan masing-masing melibatkan 80% dataset sebagai training set dan 20% dataset sebagai test set.

Pada skenario pertama, Grammatical Evolution dengan skenario tatami memperoleh akurasi tertinggi baik pada training maupun testing. Hal ini sesuai dengan hipotesis yang diajukan.

Pada skenario kedua, grammatical evolution dengan skenario tatami dan grammatical evolution dengan skenario local separability masing-masing menunjukkan hasil yang cukup baik.

Percobaaan		**GA Select**	**GE Global**	**GE Local**	**GE Multi**	**GE Tatami**
Whole	**Training**	90.00%	69.78%	90.65%	92.17%	**95.87%**
	Testing	90.00%	69.78%	90.65%	92.17%	**95.87%**
	Total	90.00%	69.78%	90.65%	92.17%	**95.87%**
Fold 1 dari 5 fold	**Training**	88.89%	71.82%	91.06%	93.49%	**95.39%**
	Testing	86.81%	67.03%	92.30%	**95.60%**	**95.60%**
	Total	88.47%	70.87%	91.30%	93.91%	**95.43%**
Fold 2 dari 5 fold	**Training**	88.89%	73.44%	**97.28%**	91.05%	95.93%
	Testing	92.30%	73.62%	**97.80%**	91.20%	**97.80%**
	Total	89.56%	73.48%	**97.39%**	91.08%	96.30%
Fold 3 dari 5 fold	**Training**	87.80%	73.98%	**94.85%**	93.49%	**94.85%**
	Testing	90.10%	71.42%	**95.60%**	93.40%	**95.60%**
	Total	88.26%	73.47%	**95.00%**	93.48%	**95.00%**
Fold 4 dari 5 fold	**Training**	89.43%	73.44%	94.03%	92.95%	**95.39%**
	Testing	85.71%	73.62%	94.50%	89.01%	**96.70%**
	Total	88.69%	73.48%	94.13%	92.17%	**95.65%**
Fold 5 dari 5 fold	**Training**	92.68%	74.25%	**98.64%**	96.20%	97.56%
	Testing	86.81%	71.43%	**97.80%**	86.81%	92.30%

	Total	91.52%	73.70%	**98.47%**	94.34%	96.52%

4.2 Data Sintesis 02

Data sintesis 02 memiliki struktur yang hampir sama dengan data sintesis 01. Namun di sini, diberikan lebih banyak kelas.

Dalam percobaan dilakukan dua jenis skenario ujicoba. Untuk ujicoba pertama, di mana data training sama dengan data testing, GE Tatami menunjukkan hasil yang sangat baik:

Sementara pada skenario kedua, dimana data dibagi menjadi 5 fold, GE tatami kembali menunjukkan hasil yang baik.

Percobaaan		**GA Select**	**GE Global**	**GE Local**	**GE Multi**	**GE Tatami**
Whole	**Training**	85.80%	54.65%	83.19%	87.60%	**91.02%**
	Testing	85.80%	54.65%	83.19%	87.60%	**91.02%**
	Total	85.80%	54.65%	83.19%	87.60%	**91.02%**
Fold 1 dari 5 fold	**Training**	85.77%	55.08%	88.01%	87.19%	**93.29%**
	Testing	86.78%	57.02%	92.56%	85.95%	**92.56%**
	Total	85.97%	55.46%	88.91%	86.95%	**92.66%**
Fold 2 dari 5 fold	**Training**	85.97%	55.89%	88.21%	84.76%	**92.68%**
	Testing	81.82%	54.55%	88.43%	80.17%	**92.56%**
	Total	85.15%	55.62%	88.25%	83.84%	**92.66%**
Fold 3 dari 5 fold	**Training**	86.38%	55.28%	89.63%	83.13%	**93.29%**
	Testing	77.68%	50.41%	86.77%	82.64%	**95.86%**
	Total	84.67%	54.32%	89.07%	83.03%	**93.86%**
Fold 4 dari 5 fold	**Training**	84.96%	51.82%	87.80%	85.36%	**93.90%**
	Testing	85.95%	57.85%	88.43%	90.98%	**91.73%**
	Total	85.15%	53.61%	87.92%	86.46%	**93.47%**
Fold 5 dari 5 fold	**Training**	84.35%	54.67%	85.57%	86.18%	**93.29%**
	Testing	87.60%	56.19%	85.12%	87.60%	**94.21%**
	Total	84.99%	54.97%	85.48%	86.46%	**93.47%**

4.3 Data Iris

Data iris merupakan dataset yang cukup banyak dipakai dalam penelitian. Data ini terdiri dari 3 kelas dan 4 atribut fitur original. Data iris bersifat multi-variate, terdiri dari 150 baris.

Pada percobaan pertama, dimana semua data digunakan untuk training sekaligus testing, GE dengan pengukuran fitness berdasarkan akurasi klasifier menunjukkan hasil terbaik (98%), disusul GE Tatami (97.3%).

Pada percobaan kedua, GE Tatami dan GE Global Fitness menunjukkan hasil terbaik.

Percobaaan		**GA Select**	**GE Global**	**GE Local**	**GE Multi**	**GE Tatami**
Whole	**Training**	96.67%	**98.00%**	97.33%	97.33%	**98.00%**
	Testing	96.67%	**98.00%**	97.33%	97.33%	**98.00%**
	Total	96.67%	**98.00%**	97.33%	97.33%	**98.00%**
Fold 1 dari 5 fold	**Training**	96.67%	97.50%	**98.33%**	**98.33%**	96.67%
	Testing	90.00%	**100%**	96.67%	93.33%	**100%**
	Total	95.53%	**98.00%**	**98.00%**	97.33%	97.33%
Fold 2 dari 5 fold	**Training**	96.67%	98.33%	98.33%	**99.16%**	**99.16%**
	Testing	93.33%	96.67%	90.00%	**98.33%**	96.67%
	Total	96.00%	98.00%	96.67%	98.00%	**98.67%**
Fold 3 dari 5 fold	**Training**	96.67%	**98.33%**	96.67%	**98.33%**	97.50%
	Testing	96.67%	96.67%	**100%**	93.33%	93.33%
	Total	96.67%	**98.00%**	97.33%	97.33%	96.67%
Fold 4 dari 5 fold	**Training**	96.67%	98.33%	97.50%	97.50%	**98.33%**
	Testing	96.67%	96.67%	96.67%	96.67%	**100%**
	Total	96.67%	98.00%	97.33%	97.33%	**98.67%**
Fold 5 dari 5 fold	**Training**	96.67%	**98.33%**	96.67%	97.50%	97.50%
	Testing	93.33%	96.67%	96.67%	96.67%	**100%**
	Total	96.00%	**98.00%**	96.67%	97.33%	**98.00%**

4.4 Data 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.

Hasil percobaan menunjukkan bahwa skenario GA Select menghasilkan akurasi yang paling tinggi dibandingkan keempat skenario lain.

Percobaaan		**GA Select**	**GE Global**	**GE Local**	**GE Multi**	**GE Tatami**
Whole	**Training**	**88.09%**	62.20%	87.79%	83.92%	83.92%
	Testing	**88.09%**	62.20%	87.79%	83.92%	83.92%
	Total	**88.09%**	62.20%	87.79%	83.92%	83.92%
Fold 1 dari 5 fold	**Training**	87.82%	43.91%	85.61%	**88.56%**	87.08%
	Testing	78.46%	44.62%	70.77%	78.46%	**81.53%**
	Total	86.01%	44.05%	82.73%	**86.60%**	86.01%
Fold 2 dari 5 fold	**Training**	88.19%	69.00%	86.72%	86.72%	**88.56%**
	Testing	80.00%	70.77%	80.00%	**83.07%**	80.00%
	Total	86.60%	69.35%	85.42%	86.01%	**86.90%**
Fold 3 dari 5 fold	**Training**	87.08%	54.24%	79.70%	84.87%	**87.45%**
	Testing	**92.30%**	55.38%	83.07%	89.23%	89.23%
	Total	**88.09%**	54.46%	80.36%	85.71%	87.80%
Fold 4 dari 5 fold	**Training**	87.08%	62.73%	**87.45%**	86.72%	84.50%
	Testing	**93.85%**	64.61%	87.69%	83.08%	86.15%
	Total	**88.39%**	63.10%	87.50%	86.01%	84.82%
Fold 5 dari 5 fold	**Training**	**89.30%**	62.73%	88.19%	85.98%	85.98%
	Testing	**84.62%**	67.69%	80.00%	81.53%	83.07%
	Total	**88.39%**	63.69%	86.60%	85.12%	85.41%

4.5 Pembahasan

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.

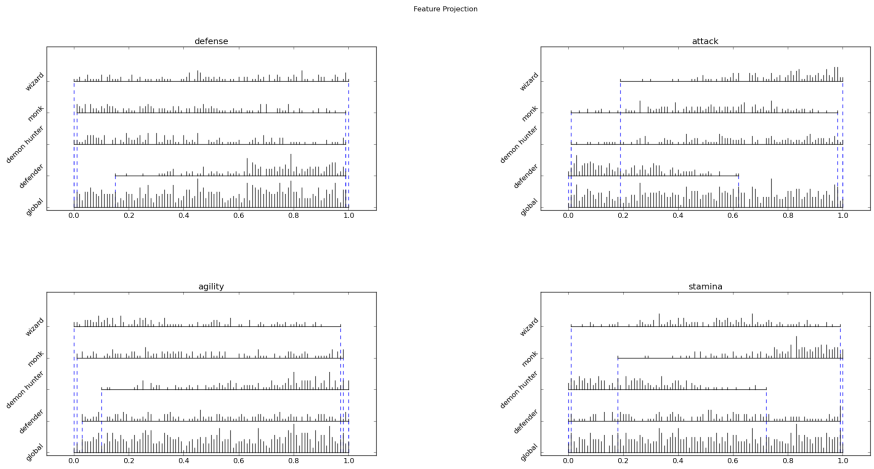
Dalam kasus data ecoli, tampak bahwa GE Tatami tidak menunjukkan hasil yang terlalu baik. Adapun demikian, masih tampak bahwa metode ini unggul pada fold ke dua. Dibandingkan dengan kasus-kasus lain, selisih jumlah antara fitur yang digenerate dan fitur original dalam kasus ecoli adalah paling banyak. GE Tatami mengenerate 4 fitur, sedangkan fitur original yang tersedia ada 7 buah.

Dimungkinkan bahwa dalam kasus ecoli, grammar yang dipakai gagal untuk merangkum semua informasi yang tersedia. Atau dimungkinkan pula kegagalan tersebut dikarenakan terlalu sedikitnya jumlah data. Kemungkinan lain adalah adanya ketidaksesuaian classifier. Classifier yang digunakan adalah gaussian naive bayes yang mengasumsikan pola sebaran data normal.

Untuk pembahasan lebih lanjut, akan diambil contoh data synthesis_02. Keterangan lengkap mengenai data ini telah dibahas pada subbab 4.2

4.5.1. Pembahasan Skenario GA Select

Pada data synthesis 2, Skenario GA Select memilih fitur yang persis sama dengan fitur original. Proyeksi masing-masing kelas terhadap masing-masing fitur disajikan pada gambar 4.1



Gambar 4.1 Proyeksi Data Synthesis 02 pada Fitur yang Digenerate GA Select

Hasil proyeksi tersebut menunjukkan bahwa data-data yang ada tidak dapat terpisahkan secara visual hanya dengan memanfaatkan salah satu fitur original saja. Adapun demikian, saat keempat fitur tersebut digabungkan, maka classifier masih bisa melakukan klasifikasi dengan keakuratan cukup tinggi sebesar 85.80%.

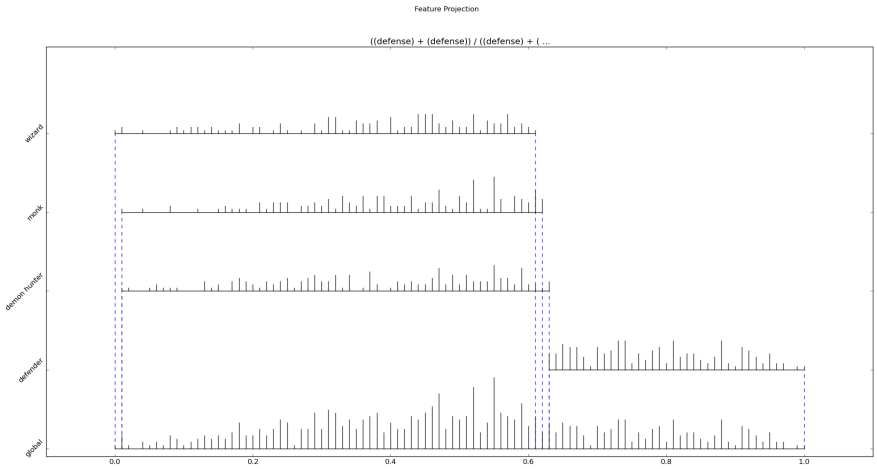
Skenario GA Select menunjukkan hasil terbaik pada data e-coli, yakni dengan akurasi sebesar 88.09%.

Skenario ini sesuai untuk kasus-kasus yang bisa diselesaikan dengan feature selection.

4.5.2. Pembahasan Skenario GE Global

Skenario GE Global hanya memungkinkan pengkonstruksian 1 fitur terbaik untuk memisahkan semua kelas. Skenario ini merekonstruksi fitur

((defense) + (defense)) / ((defense) + ((attack) + ((((stamina) - (stamina)) / (attack)) * (defense))))). Pemanfaatan fitur tersebut memberikan akurasi 56.64%.

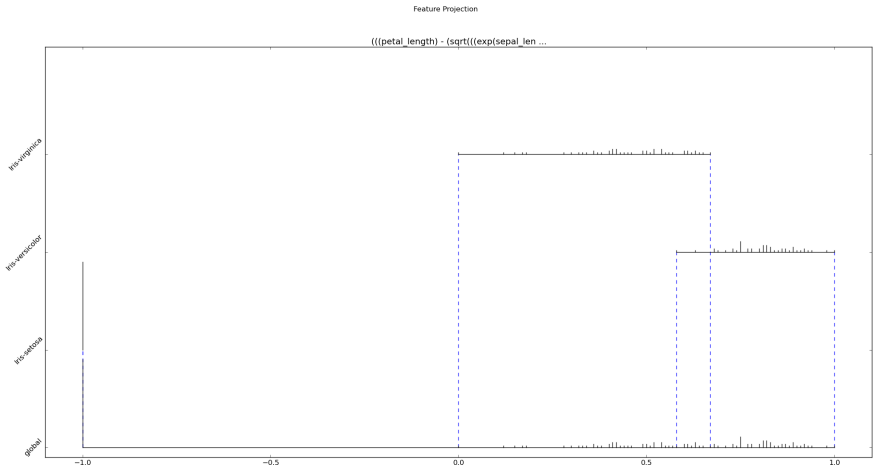


Gambar 4.2 Proyeksi Data Synthesis 02 pada Fitur yang digenerate GE Global

Pada gambar 4.2, tampak bahwa fitur yang digenerate oleh GE Global bisa memisahkan kelas *defender* dengan ketiga kelas lain. Namun demikian, fitur tersebut tak berhasil memisahkan ketiga kelas yang lain (*demon hunter*, *monk* dan *wizard*)

Walaupun pada kasus synthesis 02, GE Global tidak bisa memisahkan fitur dengan baik, namun pada kasus iris, ternyata metode ini bisa menunjukkan hasil yang cukup baik, yakni sebesar 98%.

Skenario ini tampaknya sesuai untuk data yang memiliki kemungkinan terciptanya 1 fitur untuk memisahkan semua kelas, seperti data iris.



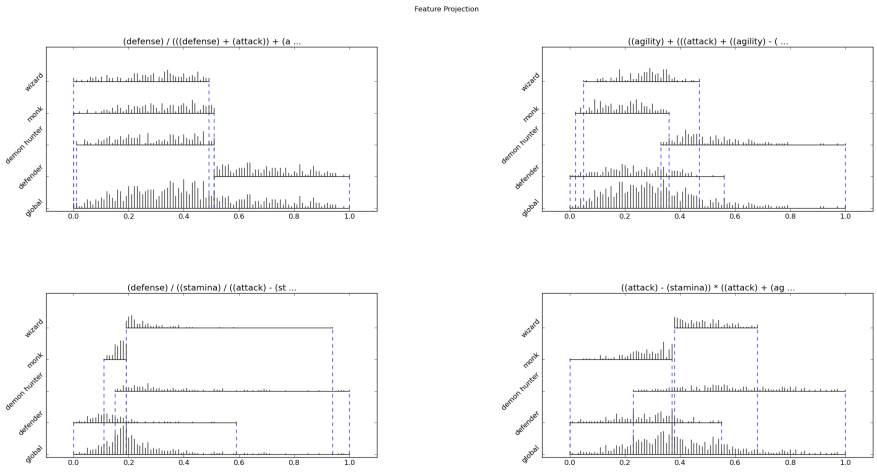
Gambar 4.3 Proyeksi Data Iris pada Fitur yang Digenerate GE Global

4.5.3. Pembahasan Skenario GE Local

Skenario GE Local memungkinkan pengkonstruksian fitur-fitur terbaik sejumlah banyaknya kelas. Masing-masing fitur bertugas untuk memisahkan satu kelas dengan semua kelas lain.

Pada data synthesis 02, skenario GE Local mengenerate fitur-fitur berikut:

- (defense) / (((defense) + (attack)) + (attack))
- ((agility) + (((attack) + ((agility) - ((defense) - (attack)))) / (stamina))) – (stamina)
- (defense) / ((stamina) / ((attack) – (stamina)))
- ((attack) - (stamina)) * ((attack) + (agility))



Gambar 4.4 Proyeksi Data Synthesis 02 pada Fitur yang digenerate GE Local

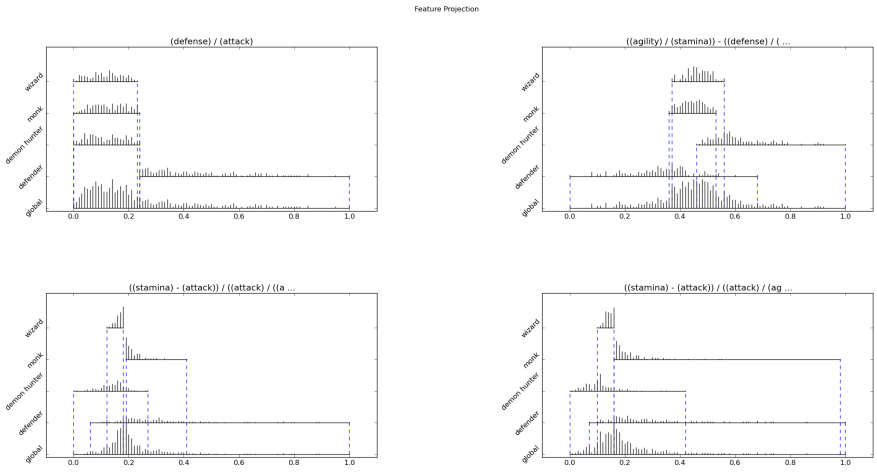
Penggunaan fitur-fitur yang digenerate oleh GE Local menghasilkan akurasi sebesar 83.9%

4.5.4. Pembahasan Skenario GE Multi

Skenario GE Multi menghasilkan fitur sebanyak jumlah kelas dengan akurasi sebesar, 87.60%

Fitur yang di generate adalah sebagai berikut:

- (defense) / (attack)
- ((agility) / (stamina)) - ((defense) / (attack))
- ((stamina) - (attack)) / ((attack) / ((agility) * (defense)))
- ((stamina) - (attack)) / ((attack) / (agility))



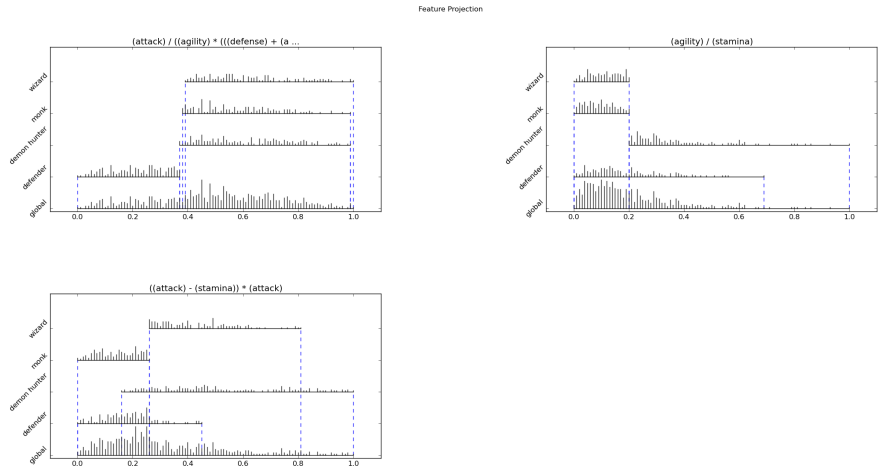
Gambar 4.5 Proyeksi Data Synthesis 02 pada Fitur yang digenerate GE Multi

4.5.5. Pembahasan Skenario GE Tatami

Skenario GE Tatami menghasilkan fitur sebanyak jumlah kelas-1. Setiap kelas memisahkan satu kelas dengan kelas-kelas lain yang belum terpisah. Penggunaan fitur-fitur yang dihasilkan oleh skenario GE Tatami memberikan akurasi sebesar 91.03%

Fitur yang digenerate dengan metode ini adalah:

- (attack) / ((agility) * (((defense) + (attack)) / (agility)))
- (agility) / (stamina)
- ((attack) - (stamina)) * (attack)



Gambar 4.6 Proyeksi Data Synthesis 02 pada Fitur yang digenerate GE Tatami

Fitur *(attack) / ((agility) * (((defense) + (attack)) / (agility)))* berhasil memisahkan *defender* dan ketiga kelas lain (*demon hunter*, *monk* dan *wizard*). Fitur *(agility) / (stamina)* berhasil memisahkan kelas *demon hunter* dengan kelas *monk* dan *wizard*. Fitur *((attack) - (stamina)) * (attack)* berhasil memisahkan *monk* dan *wizard*.

Adapun demikian, skenario GE Tatami tidak berhasil memisahkan data dengan baik pada skenario ecoli. Kegagalan ini dikarenakan tidak adanya fitur yang berhasil memisahkan minimal satu kelas dengan semua kelas lainnya. Hasil eksperimen menunjukkan bahwa nilai fitness keterpisahan terbesar pada langkah pertama hanya sebesar 0.21.

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.

Hasil penelitian juga menunjukkan bahwa skenario GA Select menunjukkan hasil yang paling stabil.

5.2 Saran

Metode-metode yang berbasis grammatical evolution sangat tergantung pada bilangan random yang digunakan untuk mengubah genotip menjadi fenotip, serta grammar yang dipakai. Penggunaan grammar yang lebih kompleks tentunya akan menambah kompleksitas, namun memungkinkan adanya peluang terbentuknya fitur-fitur baru yang lebih baik. Penggunaan grammar perlu disesuaikan dengan kasus yang ada. Untuk kasus-kasus yang berbeda, disarankan penggunaan grammar yang berbeda-beda pula.

Untuk penggunaan dalam kasus real, disarankan untuk melakukan perbandingan terlebih dulu antara akurasi dari fitur-fitur yang dibuat dengan GE Tatami atau skenario lain dengan GA Select. Jika akurasinya tidak lebih baik, maka lebih baik digunakan fitur original.

Selain penggunaan grammar yang disesuaikan dengan kasus, penggunaan fitness function yang lebih baik juga disarankan.

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