BAB 3
METODA PENELITIAN

3.1 Langkah-Langkah Penelitian

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. Untuk proses ekstraksi fitur, akan digunakan lima macam metode yang akan dibahas pada subbab 3.3.4:

   - Metode GA Select
   - Metode GE Global
   - Metode GE Multi
   - Metode GE Tatami
   - Metode GE Gavrilis

3. **Menyusun rancangan pengujian system.**

   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 *decision-tree classifier* akan digunakan sebagai

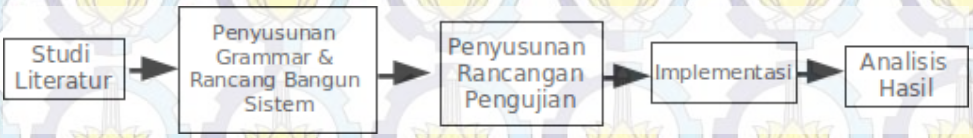
perbandingan. Khusus pada dataset synthesis 03, akan digunakan SVM dengan kernel linear sebagai pembanding.

4. **Mengimplementasikan system.**

   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:

1. Input dataset.

   Dataset yang ada ditransformasikan ke dalam bentuk yang sesuai, sehingga bisa diproses lebih lanjut oleh program

2. Menggunakan metode-metode pada subbab 3.1 langkah ke 2 guna mengekstraksi fitur.

   Semua metode yang ada (GA Select, GE Global, GE Multi, GE Tatami, dan GE Gavrilis) akan digunakan untuk mengekstraksi fitur pada data yang telah diinputkan. Langkah ini dibagi dalam beberapa tahapan

   a. *Generate* genotip.

      Genotip individu pada semua metode berupa string biner yang di-*generate* secara acak. Pada semua metode selain

GA Select, genotip-genotip ini akan ditransformasikan menjadi fenotip.

b. Mengubah genotip menjadi fenotip (fitur-fitur baru).

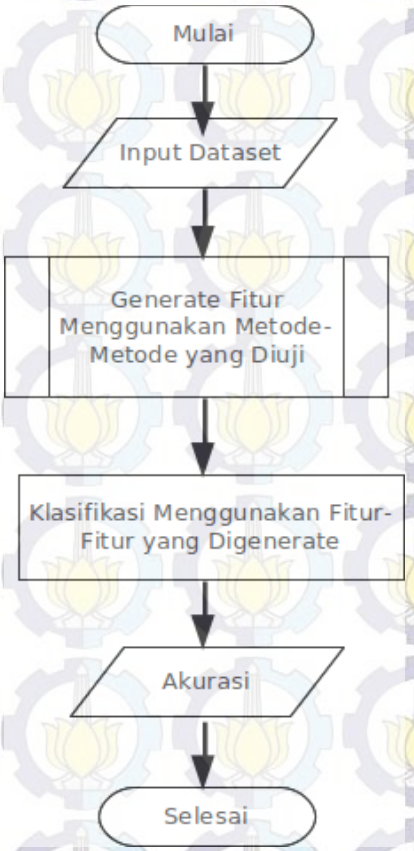
Sesuai dengan aturan *grammar* yang disediakan, setiap genotip akandiubah menjadi fenotip yang mewakili satu sebuah atau satu set fitur.

c. Menghitung nilai *fitness* dari setiap fenotip (fitur-fitur baru) terhadap setiap kelas.

Nilai *fitness* dari masing-masing fitur atau set fitur didapatkan dengan menggunakan akurasi decision tree pada ruang fitur yang tercipta.

d. Memilih fitur-fitur terbaik.

3. Membangun *feature space* berdasarkan fitur-fitur terbaik, dan melakukan proses klasifikasi.



Gambar 3.2 Flowchart Sistem.

3.3 Detail Sistem

Dalam subbab ini akan dijelaskan lebih lanjung mengenai detail rancangan sistem yang meliputi pendefinisian *grammar*, pembuatan fitur, normalisasi proyeksi data dan pemilihan fitur terbaik.

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:

3.3.1 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 disajikan dalam gambar 3.3 (diimplementasikan dalam bahasa pemrograman *Python*):

```
1.  self.variables = self.features
2.  self.grammar = {
3.          '<expr>' : ['<var>', '<stmt>'],
4.          '<stmt>' : ['(<expr>) <op>
    (<expr>)','<func>(<expr>)','sqrt(sqr(<expr>+<expr>)/2)'],
5.          '<var>'  : self.variables,
6.          '<op>'   : ['+','-','*','/'],
7.          '<func>' : ['exp','abs','sigmoid','sqr','sqrt','-']
8.      }
```

Gambar 3.3*Grammar* yang Digunakan.

Grammar tersebut diharapkan dapat men-*generate* berbagai macam variasi matematikan sederhana pada fitur-fitur original. 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>

Perhitungan srt(sqr(<expr>+<expr>)/2) digunakan untuk mengkombinasikan dua buah fitur menjadi sebuah fitur baru berupa garis dengan sudut 45 derajat terhadap kedua fitur sebelumnya.

3.3.2 Pembuatan Fitur

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

Detail program ditunjukkan dalam gambar 3.4. 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, di-*generate* sebuah fitur (fenotip) baru, Fenotip tersebut berupa string yang berisi potongan kode program dalam bahasa Python.

```
1.  def _transform(self, gene):
2.  # caching:
3.      if gene in self.genotype_dictionary:
4.          return self.genotype_dictionary[gene]
5.  # kedalaman maksimum = 20 (mencegah infinite loop)
6.      depth = 20
7.      gene_index = 0
8.      expr = self._start_node
9.  # dimulai dari level 0
10.     level = 0
11.     while level < depth:
12.         i=0
13.         new_expr = ''
14. # parsing setiap karakter pada expr
15.         while i<len(expr):
16.             found = False
17.             for key in self._grammar:
18. # ubah keyword berdasarkan akturan produksi
19.                 if (expr[i:i+len(key)] == key):
20.                     found = True
21. # jumlah kemungkinan transformasi
22.                     possibility = len(self._grammar[key])
23. # jumlah digit biner utk possibility
24.                     digit_needed =
    utils.bin_digit_needed(possibility)
25. # jika akhir gen sudah tercapai
26.                     if(gene_index+digit_needed)>len(gene):
27. # mulai dari depan lagi
28.                         gene_index = 0
29. # bagian gen utk transformasi
30.                     used_gene =
    gene[gene_index:gene_index+digit_needed]
31.                     gene_index = gene_index + digit_needed
32.                     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
# tambahkan ke cache
    self.genotype_dictionary[gene] = expr
    return expr
```

Gambar 3.4 Fungsi Transformasi untuk Mengubah Genotip Menjadi Fitur.

3.3.3 Normalisasi Proyeksi Data

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
    # pastikan isi projection hanya berupa angka (int atau float)
        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

3.3.4 Pemilihan Fitur Terbaik

Pemilihan fitur terbaik diperoleh dengan memanfaatkan lima buah metode. GE Multi dan GE Tatami merupakan metode yang diusulkan, sedangkan metode GA Select, GE Global dan GE Gavrilis digunakan sebagai pembanding.

Untuk pengukuran *fitness* individu, digunakan formula *(true_positive/(true_positive+false_negative)+true_negative/(true_negative+false*

_positive)) – 1. *True positive* adalah jumlah data yang oleh *classifier* diprediksi berada di dalam kelas tertentu dan ternyata memang benar berada dalam kelas tersebut. *True negative* adalah jumlah data yang oleh *classifier* diprediksi tidak berada di dalam kelas tertentu dan ternyata memang benar tidak berada dalam kelas tersebut. *False positive* adalah jumlah data yang oleh *classifier* diprediksi berada di dalam kelas tertentu namun ternyata tidak berada dalam kelas tersebut. *False negative* adalah jumlah data yang oleh *classifier* diprediksi tidak berada di dalam kelas tertentu namun ternyata berada dalam kelas tersebut. Dalam tabel 3.1 ditunjukkan hubungan antara *true positive*, *true negative*, *false positive*, *false negative* dan prediksi *classifier*.

Tabel 3.1 Prediksi *Classifier* dan Fakta

Prediksi *Classifier* VS Fakta	Berada dalam Kelas Tertentu	Tidak Berada dalam Kelas Tertentu
Diprediksi Berada dalam Kelas Tertentu	*True Positive*	*False Positive*
Diprediksi Tidak Berada dalam Kelas Tertentu	*False Negative*	*True Negative*

Penjelasan rinci mengenai masing-masing metode disajikan dalam subbab berikut:

3.3.4.1 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.

Setiap individu dalam GA Select terdiri dari binary string. Jika karakter pertama pada binary string adalah 1, maka fitur asli pertama digunakan, sebaliknya jika karakter pertama pada binary string adalah 0, maka fitur pertama tidak digunakan. Demikian untuk karakter-karakter selanjutnya.

Semisal sebuah individu terdiri dari binary string 01110, dan terdapat 5 fitur f1-f5, maka:

a. f1 tidak digunakan, karena karakter pertama adalah 0

b. f2 digunakan, karena karakter kedua adalah 1
c. f3 digunakan, karena karakter ketiga adalah 1
d. f4 digunakan, karena karakter keempat adalah 1
e. f5 tidak digunakan, karena karakter kelima adalah 0

Banyaknya fitur yang dapat di-*generate* dengan metode ini berkisar antara nol sampai dengan jumlah fitur original.

3.3.4.2 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 akurasi *classifier* terhadap data dengan menggunakan fenotip yang di-*generate* oleh grammatical evolution (proses grammatical evolution dijelaskan pada subbab 2.2). Metode GE Global merupakan implementasi dari penelitian sebelumnya (Gunawan, Gosaria, & Arifin, 2012)

Metode ini diharapkan menghasilkan 1 fitur terbaik yang dapat memisahkan semua kelas dalam data.

3.3.4.3 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 di-*generate* dalam metode ini adalah sebanyak jumlah kelas yang ada.

Semisal dalam sebuah dataset terdapat n buah kelas {C1, C2, C3,... Cn}, maka dalam GE Multi, akan terdapat *n fitness value* {f1, f2, f3,... fn}. Masing-masing fitness value menunjukkan keterpisahan sebuah kelas terhadap semua kelas lain. Dalam proses penentuan *fitness value*, kelas-kelas yang tidak selain kelas target akan disatukan kedalam satu kelas. Kemudian dilakukan proses klasifikasi dengan *classifier decision tree*. Proses ini dilakukan sebanyak n kali, sehingga diperoleh n *fitness* value untuk masing-masing individu.

Untuk menghitung f1, maka kelas {C2, C3,.. Cn} digabungkan menjadi C~1. *Classifier* akan memisahkan C1 dan C~1. Selanjutnya, akurasi *classifier*akan

dijadikan nilai *fitness* f1. Untuk menghitung f2, maka kelas {C1, C3,.. Cn} digabungkan menjadi C~2. *Classifier* akan memisahkan C2 dan C~2. Selanjutnya, akurasi *classifier* akan dijadikan nilai *fitness* f2. Demikian seterusnya sampai fn.

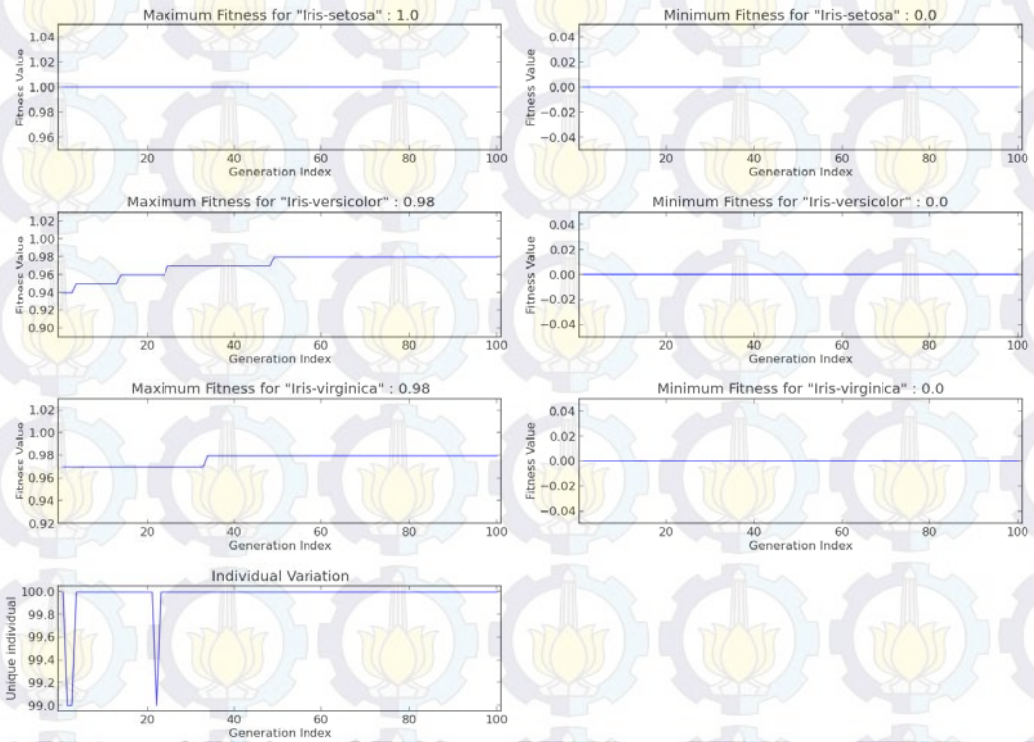
Di akhir proses GE Multi, dipilih satu individu yang memiliki f1 terbaik, 1 individu yang memiliki f2 terbaik, dan seterusnya, sehingga ditemukan n individu terbaik. Fenotip dari individu-individu terbaik ini selanjutnya digunakan sebagai fitur baru untuk proses klasifikasi.

Sebagai contoh, untuk dataset Iris yang terdiri dari 4 fitur (petal length, sepal length, petal width, dan sepal width) serta 3 kelas (iris-setosa, iris-virginica dan iris-versicolor), GE Multi menghasilkan 3 buah fitur:

- sepal_length
- (exp(sepal_width)) * ((sepal_length) / (petal_width))
- sqrt(sqr(abs(sqrt(sqr(((sepal_width) + ((sepal_width) - (sqrt(sqr((petal_length) - (sepal_length) + sepal_width)/2)))) - (petal_width) + petal_width) / 2)) + sepal_length) / 2)

Fitur pertama memiliki *fitness* value tertinggi (bernilai 1,0) untuk memisahkan iris-setosa dengan kedua kelas lain. Fitur kedua memiliki *fitnessvalue* tertinggi (bernilai 0,98) untuk memisahkan iris-versicolor dengan kedua kelas lain. Fitur ketiga memiliki *fitness value* tertinggi untuk memisahkan iris-virginica (bernilai 0,98) dengan ketiga kelas lain.

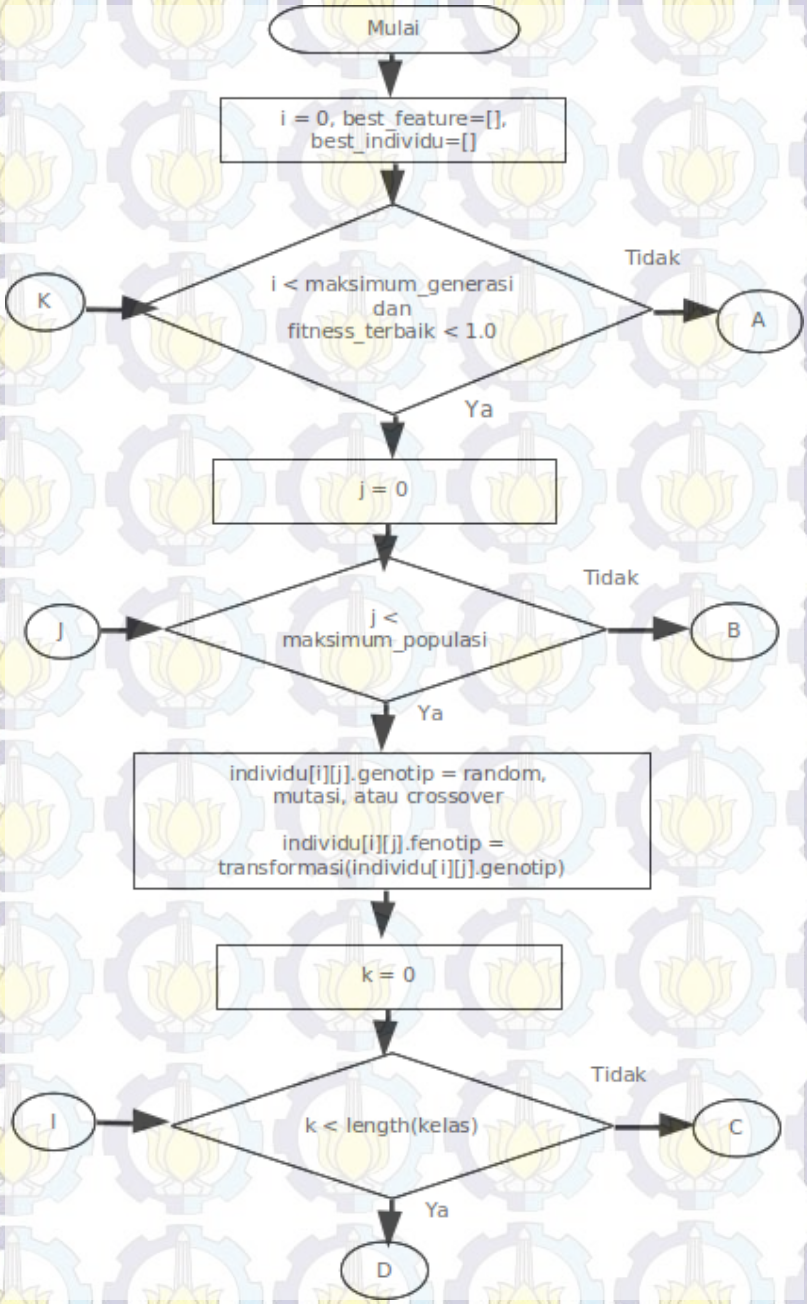
Data yang ada kemudian ditransformasi dalam menggunakan fitur-fitur baru yang telah di-*generate* GE Multi. Data hasil transformasi tadi kemudian digunakan sebagai input bagi *decision tree classifier*. Penggunaan ketiga fitur ini mengakibatkan akurasi decision tree meningkat menjadi 98,67%, dibandingkan dengan penggunaan fitur asli (sepal length, sepal width, petal length, dan petal width) yang hanya memberikan akurasi sebesar 96%.



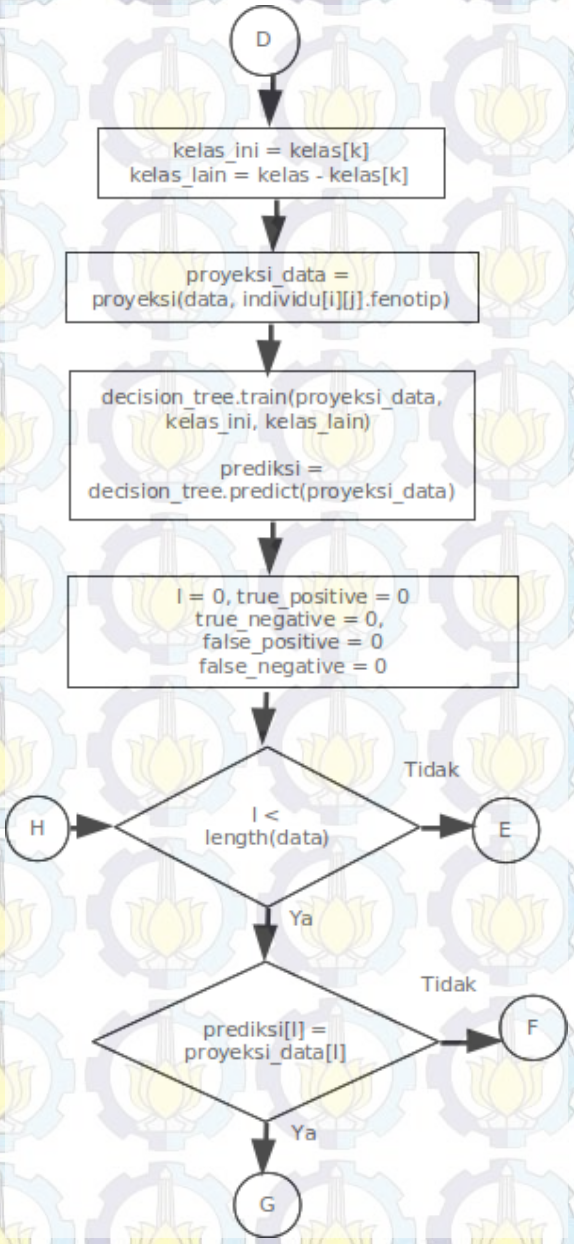
Gambar 3.6 Perubahan Nilai *Fitness* Maksimum dalam 100 Generasi

Gambar 3.6 menunjukkan perubahan nilai *fitness* terbaik pada tiap generasi. Panel-panel di bagian kiri menunjukkan nilai *fitness* maksimum yang didapat dari setiap generasi. Pada gambar tersebut tampak bahwa iris-setosa dapat terpisah secara mutlak dari kedua kelas lainnya, sedangkan iris-virginica dan iris-versicolor tidak dapat terpisah secara mutlak dari kelas lainnya.

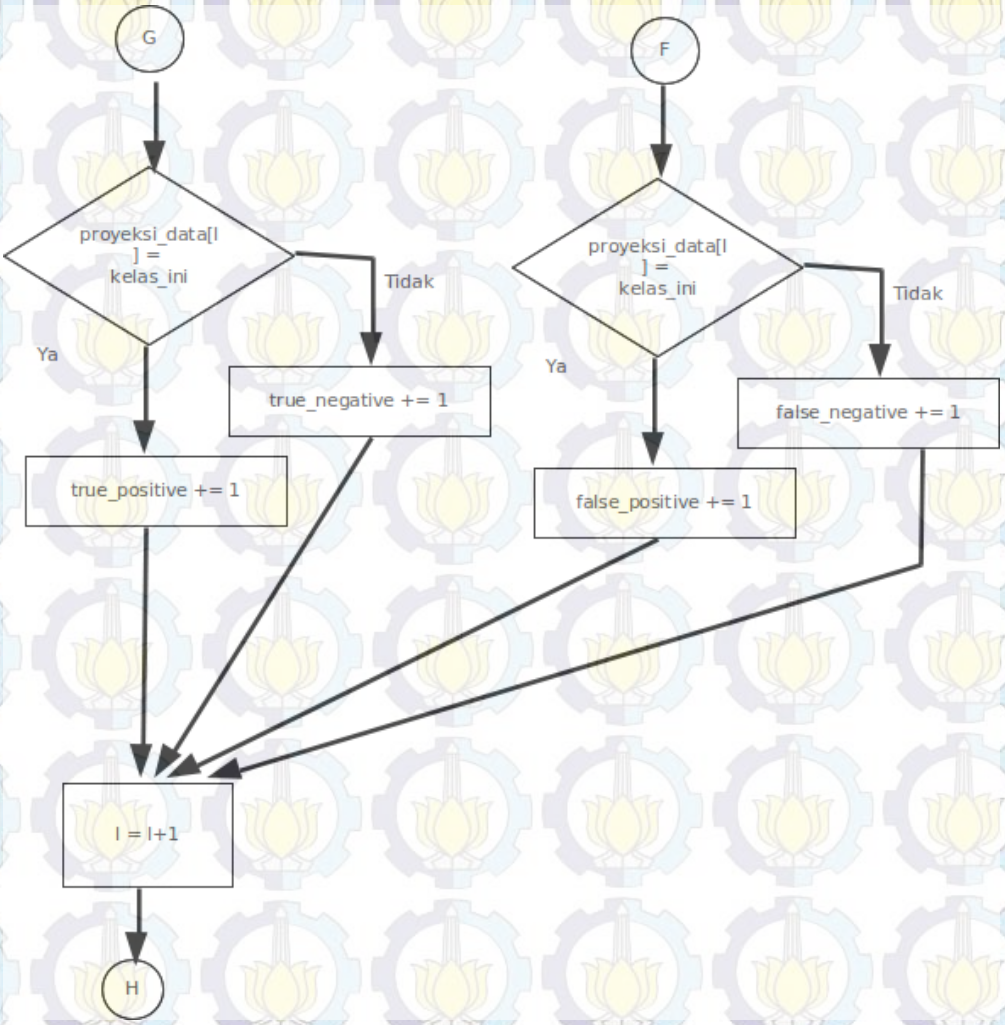
Flowchart metode GE Multi disajikan pada gambar 3.7 sampai gambar 3.10.



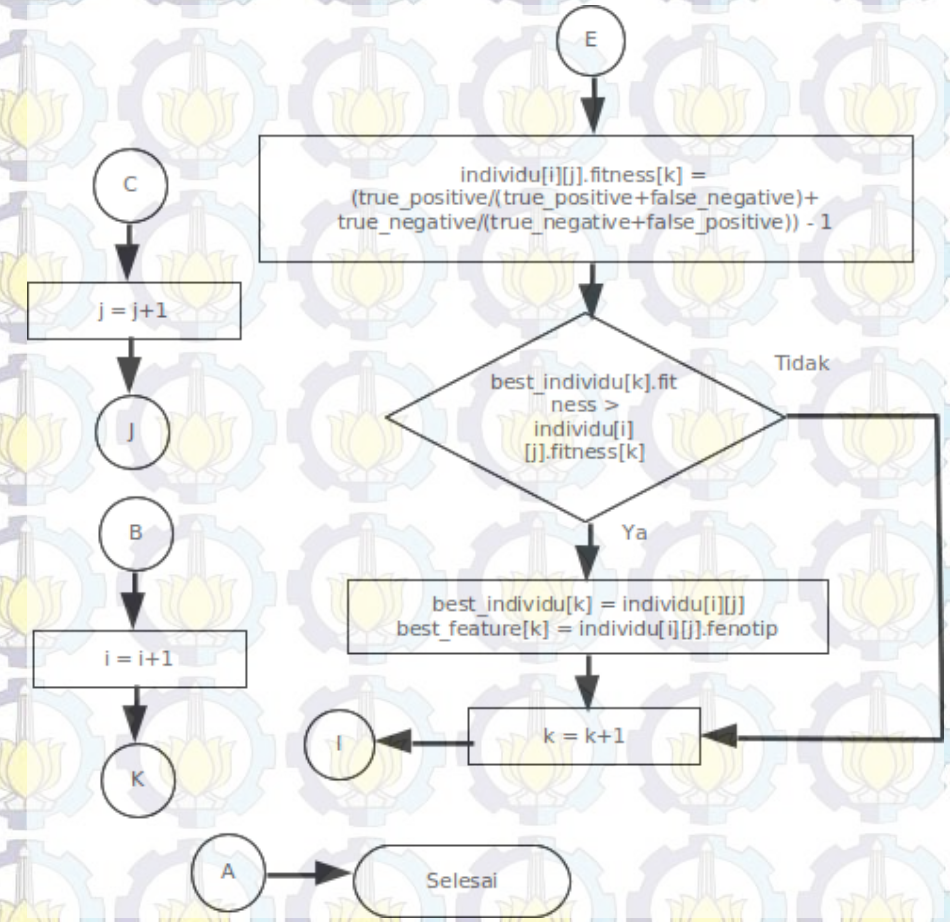
Gambar 3.7 Flowchart GE Multi (Bagian 1)



Gambar 3.8 Flowchart GE Multi (Bagian 2)



Gambar 3.9 Flowchart GE Multi (Bagian 3)



Gambar 3.10 Flowchart GE Multi (Bagian 4)

3.3.4.4 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.

Dasar pemikiran yang melandasi GE Tatami adalah bahwa *n-1* buah dimensi sebenarnya telah cukup untuk memisahkan *n* kelas pada data yang terpisah secara hirarkikal (penjelasan lebih lanjut pada subbab 4.8. Dalam ruang fitur yang terbentuk, akan dibutuhkan *n-1* buah garis linear sederhana untuk

memisahkan *n* kelas tersebut. Gambar 1.1 pada Bab I menunjukkan bagaimana 3 buah kelas dapat terpisah dalam sebuah ruang fitur dua dimensi.

Dalam GE Tatami, untuk n buah kelas{*C1, C2, C3,... Cn*}, dilakukan proses GE Multi untuk mencari individu-individu terbaik. Seperti yang telah dibahas dalam subbab sebelumnya, setiap individu dalam GE Multi memiliki *n* buah *fitnessvalue* {*f1, f2, f3,... fn*}. Di sini dipilih satu *fitness value* terbesar dari semua individu. *Fitnessvalue* terbesar dari semua individu merepresentasikan kelas yang paling terpisah dari kelas lain. Kelas ini selanjutnya disimbolkan sebagai *C*1*. Fenotip dari individu terbaik dengan nilai *fitness* tertinggi selanjutnya disebut F1, dan merupakan bagian dari fitur baru yang akan digunakan dalam proses klasifikasi.

Dalam proses selanjutnya, data-data dalam kelas *C*1* dihilangkan, sehingga diperoleh subset data baru yang terdiri dari *{C1, C2, C3,... Cn} – C*1*. Subset data baru ini akan memiliki n-1 buah kelas. Kemudian kembali dilakukan GE Multi seperti sebelumnya. Perulangan kedua ini akan menghasilkan F2 yang juga merupakan bagian dari fitur baru yang akan digunakan dalam proses klasifkiasi. Perulangan akan dilakukan sebanyak *n-1* kali. Dalam setiap perulangan, jumlah kelas yang terlibat akan berkurang satu, sehingga pada perulangan ke *n-1* hanya akan tersisa 2 kelas yang kemudian dipisahkan oleh *Fn-1*.

Sebagai contoh, untuk dataset Iris yang terdiri dari 4 fitur (petal length, sepal length, petal width, dan sepal width) serta 3 kelas (iris-setosa, iris-virginica dan iris-versicolor), GE Tatami menghasilkan 2 buah fitur:

- sepal_length
- (petal_length) - (sqrt(sqr(sepal_length + sqrt(sqr(abs(petal_length) + (sqrt(sqr(petal_length + petal_width) / 2)) - (abs(- ((sepal_width) - (petal_width))))) / 2)) / 2))

Fitur pertama memiliki *fitnessvalue* tertinggi (bernilai 1,0) untuk memisahkan iris-setosa dengan kedua kelas lain. Fitur kedua memiliki *fitness value* tertinggi (bernilai 0,96) untuk memisahkan iris-virginica dan iris-versicolor (kelas iris-setosa diabaikan karena sudah terpisah pada fitur pertama).

Fitur pertama pada GE Tatami didapatkan dengan cara menjalankan proses GE Multi, seperti yang telah dijelaskan dalam subbab 3.3.4.3. Proses GE Multi tersebut menghasilkan 3 buah individu dengan 3 nilai *fitness* terbaik untuk setiap kelas:

- sepal_length
  memiliki nilai *fitness* terbaik sebesar 1.0 untuk pemisahan antara iris setosa dengan kedua kelas lain
- (exp(sepal_width)) * ((sepal_length) / (petal_width))
  memiliki nilai *fitness* terbaik sebesar 0.98 untuk pemisahan antara iris versicolor dengan kedua kelas lain.
- sqrt(sqr(abs(sqrt(sqr(((sepal_width) + ((sepal_width) - (sqrt(sqr((petal_length) - (sepal_length) + sepal_width)/2)))) - (petal_width) + petal_width) / 2)) + sepal_length) / 2)
  memiliki nilai *fitness* terbaik sebesar 0.98 untuk pemisahan antara iris virginica dengan kedua kelas lain.

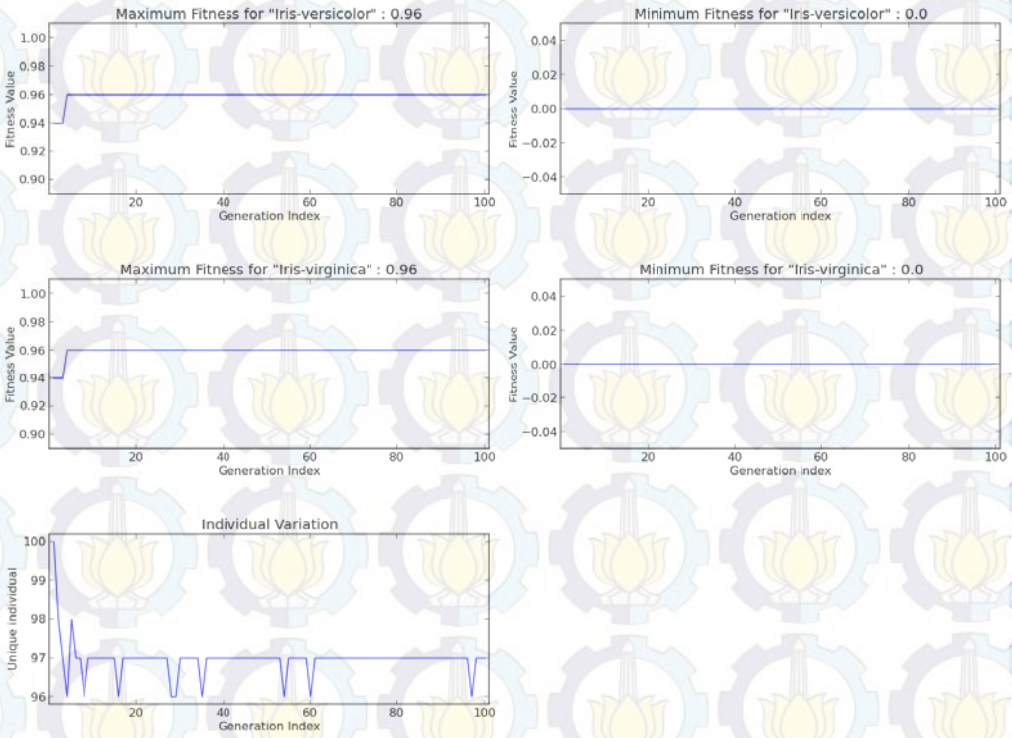
Pada proses ini, diperoleh bahwa nilai *fitness* tertinggi adalah 1.0 untuk memisahkan iris setosa dengan kedua kelas lain. Nilai *fitness* ini dimiliki oleh individu dengan fenotip sepal_length. Maka fitur sepal_length ditambahkan sebagai fitur baru dalam GE Tatami.

Kemudian proses ekstraksi fitur dilanjutkan dengan menghilangkan data yang memiliki kelas iris-setosa, dengan asumsi bahwa fitur sepal_length pada langkah pertama tadi telah memisahkan iris-setosa dengan kedua kelas lainnya. Maka data yang baru hanya terdiri dari dua buah kelas, yakni iris versicolor dan iris virginica.

Proses GE Multi kembali dijalankan pada data yang kini hanya terdiri dari dua kelas (iris virginica dan iris versicolor). Proses ini menghasilkan dua individu dengan dua nilai *fitness* terbaik. Kedua individu tersebut memiliki fenotip yang sama, yakni *(petal_length) - (sqrt(sqr(sepal_length + sqrt(sqr(abs(petal_length) + (sqrt(sqr(petal_length + petal_width) / 2)) - (abs(- ((sepal_width) - (petal_width))))) / 2)) / 2))* yang memiliki nilai *fitness* 0,96. Angka 0,96 merupakan nilai akurasi decision tree jika fitur tersebut digunakan pada data yang

kelas iris-setosa nya telah dihilangkan. Oleh sebab itu, terjadi penurunan nilai jika dibandingkan dengan *fitness* value yang dibuat oleh GE Multi pada langkah pertama. Ini wajar, karena jumlah data *false negative* ikut berkurang seiring dengan penghilangan kelas iris-setosa.

Langkah pertama pada GE Tatami, pada dasarnya adalah sama dengan GE Multi. Oleh sebab itu proses perubahan nilai *fitness* dari setiap generasi adalah sama dengan yang telah tergambar pada gambar 3.6. Sedangkan perubahan nilai *fitness* dari setiap generasi pada langkah kedua digambarkan pada gambar 3.11

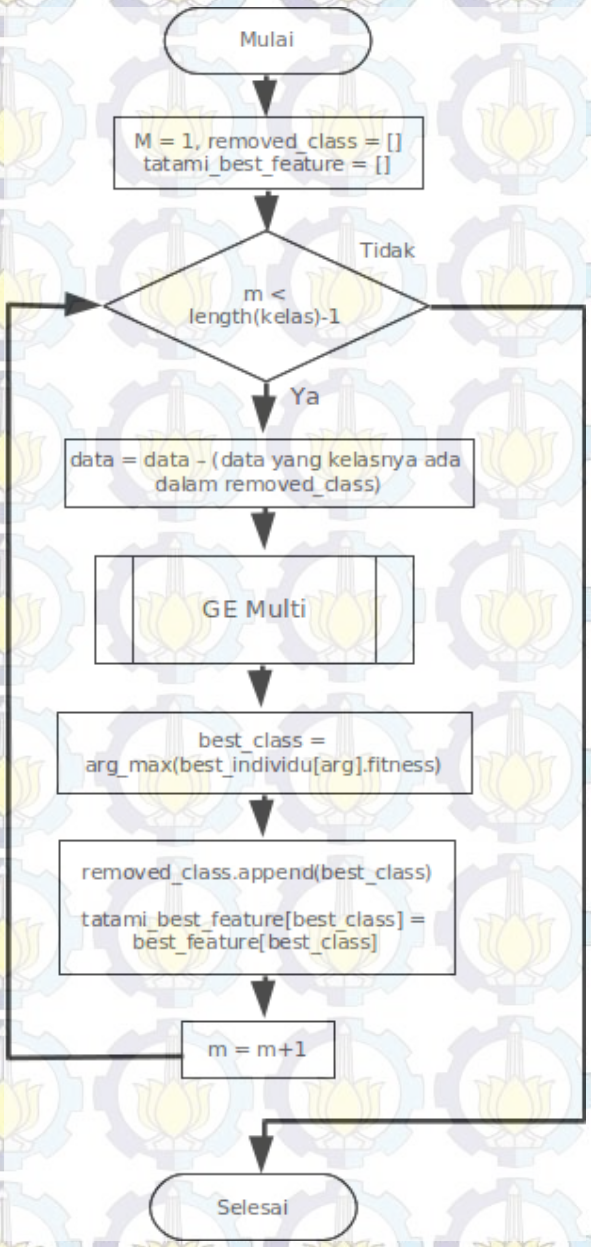


Gambar 3.11 Perubahan Nilai *Fitness* Maksimum dalam 100 Generasi pada Perulangan Kedua GE Tatami

Data yang ada kemudian ditransformasi dalam menggunakan fitur-fitur baru yang telah di-*generate* GE Tatami. Data hasil transformasi tadi kemudian digunakan sebagai input bagi *decision tree classifier*. Penggunaan ketiga fitur ini mengakibatkan akurasi decision tree meningkat menjadi 98,67%, dibandingkan dengan penggunaan fitur asli (sepal length, sepal width, petal length, dan petal

width) yang hanya memberikan akurasi sebesar 96%. Dalam kasus ini, penggunaan GE Tatami dan GE Multi menghasilkan nilai akurasi yang sama. Namun fitur yang dibuat oleh GE Tatami lebih sedikit (2 fitur saja).

Flowchart GE Tatami disajikan pada gambar 3.12.



Gambar 3.12 Flowchart GE Tatami.

3.3.4.5 Metode GE Gavrilis

Metode GE Gavrilis merupakan implementasi dari penelitian yang dilakukan oleh Gavrilis (Gavrilis, Tsoulous, & Dermatas, Selecting and Constructing Features Using Grammatical Evolution, 2008). Dalam metode ini akan di-*generate* set-set fitur yang masing-masing diwakili oleh satu individu. Berbeda dengan GE Multi dan GE Tatami, di sini satu individu mewakili satu set fitur.

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

Jumlah fitur yang di-*generate* dalam GE Gavrilis akan berkisar antara nol sampai tak terhingga.

3.3.5 Pengukuran Performa Fitur

Untuk pengukuran performa fitur sebagai acuan perbandingan atas semua metode di subbab 3.3.4, digunakan *Decision Tree classifier* yang merupakan salah satu algoritma umum dalam permasalahan klasifikasi. Semua metode yang telah dibahas pada subbab sebelumnya akan digunakan untuk men-*generate* 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/)