Forum Statistika dan Komputasi, Oktober 2010 p : 10-15 Vol 15 No.2
ISSN : 0853-8115

PENGENALAN ALGORITMA GENETIK UNTUK PEMILIHAN PEUBAH PENJELAS DALAM MODEL REGRESI MENGGUNAKAN SAS/IML

(An Introduction to Variables Selection in Regression Analysis using Genetic Algorithm with SAS/IML)

Bagus Sartono
Peneliti di Data Mining Center – Departemen Statistika IPB
E-mail : bagusco4@yahoo.com, bagusco@ipb.ac.id

Abstract

Genetic algorithm has been a popular alternative in the various fields of optimization problem. This paper describes some basic ideas of this algorithm and its application for selecting significant variables in the regression analysis. Simple SAS/IML commands are presented in order to emphasize how the algorithm works. It is also available to do some modification in some parts of those commands.

Keyword: *data mining, cross-over*

PENDAHULUAN

Pengambil keputusan sering harus menghadapi permasalahan optimasi yang sulit atau terlalu mahal secara komputasi untuk memperoleh solusi eksak bagi tercapainya kondisi optimum. Ini terutama jika permasalahannya terkait dengan masalah kombinatorika dengan jumlah kemungkinan yang sangat banyak. Sebut saja misalnya permasalahan penentuan urutan pengerjaan 10 tugas sehingga diperoleh hasil terbaik. Jika setiap tugas bisa dilakukan sebelum atau sesudah tugas yang lain, maka akan ada 10! atau lebih dari 3.6 juta kemungkinan urutan yang harus dievaluasi dan kemudian ditentukan mana yang terbaik. Seandainya ada 20 tugas, maka angka banyaknya kemungkinan urutan membengkak mencapai lebih dari 3.4×10^{18}. Suatu angka yang hampir pasti terlalu besar untuk mengevaluasi seluruhnya.

Metode-metode optimasi meta-heuristik berupaya untuk mengurangi biaya komputasi ini tanpa melakukan evaluasi terhadap seluruh kemungkinan solusi. Meskipun teknik ini tidak dapat diandalkan untuk mendapatkan solusi optimum global, namun dalam banyak situasi mampu memberikan solusi yang mendekati nilai tersebut. Salah satu algoritma optimasi yang dapat digunakan adalah algoritma genetik (*genetic algorithm*) dan selanjutnya dalam tulisan ini akan banyak disebut dengan GA.

Algoritma ini telah tercatat berhasil memberikan solusi yang sangat baik di berbagai kasus seperti yang antara lain dilaporkan dalam beberapa literatur. Leardi (2001) memberikan ulasan tentang penggunaan GA serta menampilkan daftar literatur yang cukup panjang tentang penerapan algoritma ini di bidang kimia khususnya dalam kemometrika. Radhakrishnan *et. al* (2009) menerapkan GA dalam bidang manajemen penyimpanan barang dan melaporkan bahwa dengan GA mereka mendapatkan solusi yang mampu menekan biaya dan kejadian kelangkaan. Sementara diskusi mengenai penggunaan GA untuk optimasi di bidang keuangan dapat dilihat dalam Pereira (2000). Dalam kaitan dengan analisis statistika, GA juga dapat diterapkan dalam pemilihan peubah penjelas pada analisis regresi pada saat kita berhadapan dengan data yang terdiri atas ratusan atau bahkan ribuan peubah seperti dalam banyak kasus di *data mining*.

Meskipun telah banyak tulisan mengenai topik tersebut, dalam tulisan ini penulis mencoba memberikan penjelasan sederhana bagaiamana GA bekerja dalam pemilihan peubah penjelas. Beberapa perintah dasar menggunakan software SAS/IML juga ditampilkan dengan tujuan memperjelas uraian yang ada. Perlu diketahui, SAS/IML telah menyediakan beberapa fungsi terkait dengan GA, namun kami tidak menggunakan fungsi-fungsi built-in tersebut dan memilih untuk menggunakan perintah-perintah dasar yang lebih mudah diikuti oleh pembaca yang tidak memiliki latar belakang statistika dan komputasi.

Tulisan ini terbagi dalam beberapa bagian yang secara berurutan diawali dengan gambaran tentang GA secara umum hingga bagaimana implementasinya untuk kasus pemilihan peubah penjelas dalam analisis regresi.

ALGORITMA GENETIK

Algoritma ini bekerja dengan meniru perilaku dalam proses evolusi yang dialami makhluk hidup dimana dari generasi ke generasi hanya mereka yang memiliki kemampuan lebih yang dapat bertahan. Di dalamnya juga berlangsung proses perubahan genetik individu anak yang diturunkan dari sifat genetik tetuanya melalui kromosom.

Penjelasan sederhana dari proses evolusi ini dapat diberikan sebagai berikut.

Pada awalnya populasi makhluk hidup terdiri atas individu-individu dengan karakteristik yang sangat heterogen. Kondisi lingkungan menyebabkan hanya individu yang mampu mengatasi masalah yang dapat bertahan. Individu tersebut dapat disebut sebagai individu terbaik dan yang paling tepat dengan kondisi yang ada. Selanjutnya terjadi proses perkawinan yang menghasilkan individu anak dan menjadi generasi berikutnya yang menggantikan generasi pendahulunya. Karakteristik genetik anak merupakan gabungan (*recombinant*) dari sifat genetik yang ada dalam kromosom tetuanya melalui proses cross-over. Berikutnya kembali terjadi proses seleksi yang menentukan individu mana yang bisa bertahan. Begitu seterusnya terjadi berulang-ulang dari generasi ke generasi. Yang juga terjadi adalah proses mutasi kromosom, berupa perubahan kode kromosom karena pengaruh eksternal. Tingkat kejadian mutasi ini umumnya sangat rendah.

Proses yang digambarkan di atas itulah yang diadopsi dalam algoritma genetik. Dreo *et al* (2006) menyebutkan bahwa algoritma ini awalnya dikembangkan oleh JH Holland pada tahun 1975. Namun, baru mendapat perhatian luas dan popular digunakan setelah D. E. Goldberg menuliskannya dalam buku "Genetic Algorithms in Search, Optimization and Machine Learning" tahun 1989.

Meskipun tidak selalu demikian, dalam tulisan ini setiap solusi yang mungkin dari suatu permasalahan dalam bentuk barisan angka 1 dan 0, dan dipandang sebagai kromosom dari individu tersebut. Selanjutnya, langkah-langkah dari GA ini adalah:

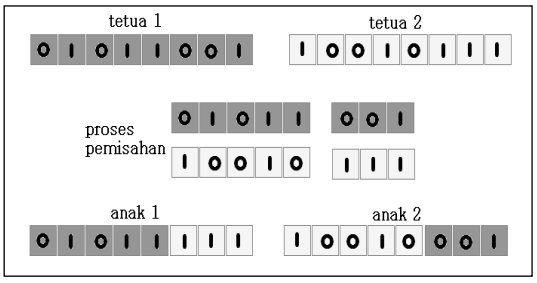
1. Pembangkitan populasi
2. Seleksi individu untuk proses regenerasi maupun untuk yang dapat bertahan
3. Persilangan/cross-over antar individu untuk menghasilkan anak bagi generasi selanjutnya
4. Mutasi, yang berupa perubahan nilai pada kromosom dengan laju perubahan yang sangat rendah
5. Pergantian generasi, dan proses berulang untuk langkah 2 hingga langkah 4 hingga berulang dalam beberapa generasi yang banyaknya ditentukan sebelumnya.

Proses seleksi didasarkan pada kualitas individu menggunakan besaran yang ingin dioptimalkan. Pada kasus *travel salesman problem* untuk menentukan urutan kota yang dikunjungi misalnya, total waktu yang diperlukan adalah besaran yang dipakai untuk mengukur kualitas setiap individu solusi. Sedangkan pada kasus pemilihan peubah dalam regresi, besaran seperti koefisien determinasi dan fungsi kemungkinan bisa menjadi pilihan.

Proses seleksi sendiri bisa dilakukan dengan berbagai cara. Yang paling mudah adalah mengambil beberapa individu terbaik. Selain itu juga dapat dilakukan dengan proses pengambilan acak proporsional, dengan proporsi setara dengan fungsi kualitasnya. Artinya, individu yang kualitasnya lebih baik memiliki peluang terpilih lebih besar dan pengambilan dilakukan dengan pemulihan. Cara yang lain lagi adalah menggunakan teknik turnamen. Dengan teknik ini, dilakukan pemilihan beberapa individu secara acak dan individu terbaik dari subset itu dinyatakan sebagai individu terpilih. Proses diulang dengan pemulihan beberapa kali hingga diperoleh n_{sel} individu. Sebanyak n_{sel} individu ini selanjutnya digunakan dalam proses pembangkitan generasi baru melalui proses perkawinan dan cross-over.

Generasi baru dalam konsep GA adalah individu yang dihasilkan dari tetua dengan kualitas yang baik. Proses seleksi pada tahap sebelumnya diharapkan mampu memenuhi persyaratan tersebut. Cross-over merupakan upaya memperoleh individu baru dari gabungan dua tetua dan diharapkan mampu mewariskan sifat baik para tetuanya. Pada prinsipnya, proses ini adalah menukar kode kromosom dari tetua pertama ke tetua kedua, dan sebaliknya. Sehingga individu anak mewarisi sebagian kode dari tetua pertama dan sebagian dari tetua kedua.

Sebagai ilustrasi sederhana, Gambar 1 menampilkan proses *single point cross-over*. Masing-masing kromosom tetua dibagi menjadi dua bagian, dan selanjutnya membentuk dua anak baru yang merupakan persilangan dari dua bagian tersebut. Teknik *croos-over* lain adalah *double point cross-over* dan *uniform cross-over* (Dreo, 2006).



Gambar 1. Ilustrasi proses *single point cross-over* dalam GA.

Pada tahap selanjutnya suatu individu juga mengalami proses mutasi, berupa perubahan kode dari 0 ke 1 dan sebaliknya pada beberapa lokasi. Tingkat terjadinya mutasi ini umumnya sangat kecil, yaitu sekitar 1% atau kurang. Mutasi tidak bisa diatur terlalu besar karena perubahan dari generasi ke generasi akan menjadi seperti langkah acak dan akan terlalu lama mencapai konvergensi. Bentuk proses mutasi dapat terjadi dengan beragam

cara tergantung bagaimana pengguna GA mendefinisikannya.

Terdapat dua cara yang popular dalam proses pergantian. Perbedaan dari keduanya terletak pada apakah tetua yang terpilih dari generasi sebelumnya dipertahankan atau tidak. Jika dipertahankan maka generasi yang baru terdiri dari tetua dengan kualitas baik dan anak hasil perkawinan tetua tersebut. Itu adalah cara yang pertama. Sedangkan cara yang kedua adalah hanya menyertakan individu hasil persilangan saja dalam generasi yang baru.

Banyaknya generasi yang dibangkitkan sangat tergantung pada kompleksitas permasalahan dan kecepatan konvergensi. Pada program-program tertentu, ditampilkan grafik yang memonitor pergerakan kualitas dari setiap generasi sehingga pengguna dapat menghentikan kapan saja jika menganggap kualitasnya sudah cukup atau tidak terlihat perubahan kualitas setelah beberapa lama.

Sebagaimana disebutkan dalam bagian Pendahuluan, GA telah banyak diterapkan dan salah satunya adalah dalam pemilihan peubah penjelas pada analisis regresi.

PERMASALAHAN PEMILIHAN PEUBAH DALAM REGRESI

Analisis regresi merupakan salah satu metode analisis data yang sangat popular. Analisis ini menghasilkan model statistik yang menyatakan dugaan hubungan antara segugus peubah penjelas dengan gugus peubah respon. Dengan model yang dihasilkan tersebut, ada dua manfaat yang ingin diperoleh. Yang pertama adalah menyelidiki pengaruh, baik arah maupun besarannya, dari peubah penjelas terhadap peubah respon. Manfaat yang kedua adalah melakukan pendugaan atau prediksi nilai peubah respon jika nilai-nilai peubah penjelas dari suatu individu telah diketahui. Adanya manfaat kedua ini yang menjadi alasan memasukkan analisis regresi sebagai predictive analysis dalam berbagai modul datamining.

Terkait dengan hal tersebut, prinsip yang perlu dipegang dalam pembuatan model adalah prinsip kesederhanaan. Prinsip ini menyukai model yang melibatkan parameter yang lebih sedikit atau dengan demikian jumlah peubah yang lebih sedikit. Peubah-peubah penjelas yang memang mampu memberikan kontribusi dalam pendugaan model saja yang sebaiknya dilibatkan, kecuali ada landasan teori terkait dalam penyusunannya. Sehingga jelas bahwa tahapan pemilihan peubah penjelas merupakan salah satu hal utama dalam melakukan analisis regresi.

Metode-metode pemilihan peubah seperti *forward*, *backward*, dan *stepwise* adalah metode yang paling sering kita temui dilakukan para analis. Seleksi menggunakan pendekatan ini dapat mengaburkan kesimpulan karena ada peubah penjelas yang saling berkorelasi akan terbuang meskipun sebenarnya memiliki keterkaitan kuat dengan peubah respon. Upaya lain yang bisa diterapkan adalah dengan menerapkan konsep *penalized* atau *regularization*, seperti dalam metode *ridge regression* dan LASSO. Juga, yang bisa dilakukan adalah melakukan pemilihan peubah dengan menerapkan GA seperti yang akan diuraikan berikut. Metode GA terutama dapat dijadikan pilihan ketika peubah penjelas yang ada berjumlah sangat banyak.

PENERAPAN ALGORITMA GENETIK DALAM PEMILIHAN PEUBAH

Andaikan permasalahan yang dihadapi melibatkan gugus yang terdiri atas p buah peubah penjelas. Setiap peubah diberi nomor mulai dari 1 hingga p. Sebuah vektor biner berukuran p x 1 yang hanya berunsurkan nilai 1 dan 0 dapat dipandang sebagai vektor indikator yang menunjukkan apakah peubah-peubah penjelas ada di dalam model, dengan kode 1 pada unsur ke-i jika peubah nomor i ada dalam model. Dua vektor yang berbeda mewakili dua model dengan komposisi peubah penjelas yang berbeda. Penotasian inilah yang digunakan dalam mengimplementasikan GA dalam proses pemilihan peubah untuk analisis regresi. Vektor biner tersebut dapat dipandang sebagai kromosom individu tertentu dan selanjutnya dapat dinilai kualitasnya.

Berikut ini akan dipaparkan implementasi dari algoritma GA untuk pemilihan variabel dalam analisis regresi, sekaligus disajikan perintah SAS/IML untuk melakukannya (lihat Lampiran 1). Pada ilustrasi program ini, diasumsikan permasalahan yang dihadapai adalah memilih peubah untuk model regresi linear dari 400 peubah penjelas yang ada. Nilai-nilai peubah penjelas ada pada kolom 1 hingga 400 dari data dengan nama DATA, sedangkan peubah respon disimpan pada kolom ke 401 pada data yang sama. Banyaknya pengamatan dalam kasus ini adalah 200.

Dalam program ini, jumlah individu pada populasi di setiap generasi dinotasikan "*npol*" dan jumlah individu yang terambil dalam setiap tahap seleksi dinotasikan "*nsel*". Sementara itu, banyaknya generasi yang dievaluasi adalah "*ngen*" seperti yang dapat dilihat pada penggalam program di bagian LANGKAH 0.

Langkah pertama dari algoritma GA adalah membangkitkan populasi yang terdiri atas "*npol*" individu. Pada ilustrasi ini populasi dibangkitkan secara acak. Karena vektor biner hanya berunsur 1 dan 0 kita bisa menggunakan pembangkitan bilangan acak Bernoulli. Penentuan peluang dalam sebaran Bernoulli ditentukan berdasarkan jumlah peubah yang kita inginkan ada dalam setiap model. Seandainya kita menginginkan ada sekitar

20 peubah di setiap model (berarti 5% dari total banyaknya peubah yang ada), maka penggunaan 0.05 dapat dijadikan pilihan. Individu-individu yang ada, yaitu model-model dengan peubah penjelas yang berbeda tergantung pada di posisi mana angka 1 muncul pada kromosom, selanjutnya dievaluasi nilai koefisien determinasinya (R-sq). Dalam ilustrasi ini R-sq digunakan hanya untuk memudahkan pemahaman jalannya algoritma. Pembaca dapat memodifikasi menggunakan statistik lain.

Selanjutnya, setelah memperoleh nilai kualitas dari setiap model dalam populasi, dilakukan pemilihan atau seleksi untuk memperoleh "*nsel*" model yang bertahan. Meskipun banyak cara melakukan proses seleksi, dalam ilustrasi model ini digunakan teknik pemeringkatan sederhana. Sebanyak "*nsel*" model dengan R-sq terbesar adalah model-model yang dipertahankan. Model-model tersebut selanjutnya disebut sebagai tetua dan memasuki tahap perkawinan silang antar mereka untuk memperoleh anak.

Metode cross-over sederhana dilakukan untuk menghasilkan kromosom anak dalam ilustrasi program. Setiap anak mendapat setengah bagian dari kromosom dua tetua. Dengan demikian dari dua tetua diperoleh dua buah anak. Dan, karena ada "*nsel*" tetua maka ada 2(*nsel*)(*nsel* – 1) anak yang dihasilkan, dan ini menjadi populasi generasi berikutnya. Masing-masing individu dalam generasi baru hasil cross-over tersebut selanjutnya mengalami proses mutasi kromosom. Mutasi merupakan proses perubahan kromosom. Jika suatu individu mengalami mutasi maka kode kromosom tertentu yang semula bernilai 1 berubah menjadi 0 atau sebaliknya. Proses mutasi yang diberikan dalam ilustrasi ini dapat dibaca pada perintah yang ada. Seandainya bilangan acak yang menandakan mutasi terjadi tepat pada kode kromosom yang bernilai 1, maka kode kromosomnya berubah menjadi 0 dan kode di sebelah kanannya menjadi 1. Begitu proses mutasi selesai, setiap individu kembali dievaluasi kualitasnya melalui besaran R-sq dan proses seleksi kembali dilakukan untuk mendapatkan generasi-generasi berikutnya. Proses iterasi ini berhenti setelah sebanyak ngen generasi telah dibangkitkan.

Setelah mengalami evolusi dalam beberapa generasi, kita bisa mengharapkan bahwa di generasi yang paling akhir populasi terdiri atas model-model dengan kualitas yang bagus. Berdasarkan kondisi populasi model dari generasi terakhir tersebut kita bisa menentukan peubah-peubah apa saja yang sebaiknya dipilih sehingga menghasilkan model yang memuaskan. Salah satu cara yang dapat dilakukan adalah melihat peubah apa yang paling sering muncul atau paling sering terlibat pada generasi tersebut. Selain tentu kita bisa urutkan model mana yang memberikan R-sq paling besar. Dari file data dengan nama "*genterakhir*" kita bisa mendapatkan jawaban mengenai hal tersebut.

Kami melakukan percobaan terhadap program yang disusun terhadap data rekaan yang dibangkitkan menggunakan program berikut.

```
%macro buatdata;
data DATA;
do i = 1 to 200;
%do j = 1 %to 400;
x&j = rannor(0);
;
%end;
y = x1+x20+x29+x50+x90+x91;
output;
end;
run;

data DATA;
set DATA;
drop i;
run;
%mend;
%buatdata;
```

Macro "buatdata" menghasilkan data dengan nama DATA yang terdiri atas dan dengan isi seperti yang dijelaskan sebelumnya untuk keperluan sesuai dengan program yang telah disusun. Peubah respon Y merupakan penjumlahan dari peubah penjelas dengan nomor 1, 20, 29, 50, 90, dan 91. Dengan demikian, jika dilakukan proses pemilihan peubah menggunakan GA, maka sangat diharapkan keenam peubah tersebut teridentifikasi sebagai peubah penting dalam membangun model regresi. Program yang telah dideskripsikan kami jalankan dengan jumlah generasi sebanyak 10000. Dibutuhkan waktu sekitar 26 menit menggunakan SAS 9.2 pada komputer personal dengan spesifikasi processor berkecepatan 2.13 GHz dan RAM 2 GB. Pada generasi terakhir, peubah-peubah yang muncul 40 kali atau lebih dari 90 model yang ada adalah peubah dengan nomor 1, 20, 29, 50, 90, 91, 201, 229, 250, dan 291. Keenam peubah yang memang diatur sebagai peubah yang penting dapat teridentifikasi seluruhnya oleh GA.

KESIMPULAN

Dalam tulisan ini telah dipaparkan konsep dasar algoritma genetik dan penerapannya dalam pemilihan peubah penjelas pada kasus analisis regresi yang memiliki gugus kandidat peubah yang jumlahnya sangat banyak. Implementasi program menggunakan SAS/IML dengan operator-operator algoritma yang sederhana juga ditampilkan dengan harapan dapat dimodifikasi dengan mudah oleh pengguna. Ilustrasi menggunakan data rekaan menunjukkan bahwa algoritma ini sangat efektif dalam menemukan peubah penting dalam model regresi.

DAFTAR PUSTAKA

Dr´eo J., P´etrowski, A., Siarry P., &. Taillard E. 2006 Metaheuristics for Hard Optimization. Springer-Verlag Berlin Heidelberg

Leardi R. 2001. Genetic algorithms in chemometrics and chemistry: a review. *J. Chemometrics* 15: 559–569

Pereira, R., 2000. *Genetic Algorithm Optimisation for Finance and Investment*. Papers 00.02, La Trobe - Department of Economics.

Radhakrishnan, P., Prasad V.M., and Gopalan M.R. 2009. Optimizing Inventory Using Genetic Algorithm for Efficient Supply Chain Management. *Journal of Computer Science* 5 (3): 233-241.

Lampiran 1. Implementasi algoritma genetik sederhana dalam SAS/IML

```
proc IML;
/* LANGKAH 0: pengaturan banyaknya individu dan jumlah generasi yang dibangkitkan
*/

npop = 90;  /* banyaknya individu populasi */
nsel = 10;  /* banyaknya individu hasil seleksi */
ngen = 10; /* banyaknya generasi yang dibangkitkan */

/* membaca data peubah penjelas dan peubah respon*/
use DATA;
read all into A;
close DATA;
Y = A[,401];
X = A[,1:400];

/* LANGKAH 1: Pembangkitan populasi awal sebanyak npop individu secara acak dan
penilain kualitas model menggunakan R-sq */
populasi = j(npop,400,.);
call randgen(populasi,'BERN',0.05);

/* menghitung R-sq dari setiap individu */
do i = 1 to npop;
xvar = j(200,1,1);
do j = 1 to 400;
if populasi[i,j] = 1 then xvar = xvar || X[,j];
end;
residual = y - xvar * inv(xvar` * xvar) * xvar` * y;
ssresidual = ssq(residual);    /* residual SS  */
cssy=ssq(y-sum(y)/200);         /* corrected total SS  */
rsquare=(cssy-ssresidual)/cssy;      /* RSQUARE             */
performance = performance//rsquare;
end;

populasi = populasi || performance;

/* menghitung rata-rata R-sq populasi dari suatu generasi*/
averageperformance = sum(performance)/npop;
/* menghitung nilai maksimum R-sq populasi dari suatu generasi */
maxperformance = performance[<>,];
/* menyimpan nilai rata-rata dan maksimum R-sq dari setiap generasi */
/* pada tahap selanjutnya, informasi ini dijadikan bahan pembuatan plot */
poolperformance = {0}||averageperformance||maxperformance;

do repeat = 1 to ngen;

/* LANGKAH 2: pemilihan individu terbaik (sebanyak nsel)*/
call sort(populasi, {401} , {401});
selected = populasi[1:nsel,1:400];

/* menghapus data populasi dan performanya. populasi pada generasi selanjutnya
adalah hasil dari cross-over dan mutasi */
data = npop * 401;
populasi = remove (populasi, 1:data);
performance = remove (performance, 1:npop);
```

```
/* LANGKAH 3: simple crossover */
do i = 1 to nrow(selected)-1;
do j = (i+1) to nrow(selected);
populasi = populasi // (selected[i,1:200] || selected[j,201:400]);
populasi = populasi // (selected[i,201:400] || selected[j,1:200]);
end;
end;
/* LANGKAH 4: mutasi */
mutation = j(npop,400,.);
call randgen(mutation,'BERN',0.01);
do i = 1 to npop;
do j = 1 to 400;
if mutation[i,j] = 1 & populasi[i,j] = 1 then do;
 populasi[i,j] = 0;
 if j = 400 then populasi[i,1] = 1;
 else populasi[i,j+1] = 1;
 end;
end;
end;

/* menghitung R-sq dari setiap individu */
do i = 1 to npop;
xvar = j(200,1,1);
do j = 1 to 400;
if populasi[i,j] = 1 then xvar = xvar || X[,j];
end;
residual = y - xvar * inv(xvar` * xvar) * xvar` * y;
ssresidual = ssq(residual);
cssy=ssq(y-sum(y)/200);
rsquare=(cssy-ssresidual)/cssy;
performance = performance//rsquare;
end;

populasi = populasi || performance;
averageperformance = sum(performance)/npop;
maxperformance = performance[<>,];
poolperformance = poolperformance//(repeat||averageperformance||maxperformance);
end;

importance = populasi[+,1:400];
importance = importance`;

call sort(populasi, {401} , {401});

create perf from poolperformance; append from poolperformance;
create impo from importance; append from importance;
create genterakhir from populasi; append from populasi;

quit;

/* plot perkembangan rata-rata R-sq tiap generasi */
proc plot data=perf; plot col2*col1; run;
/* plot perkembangan max R-sq tiap generasi */
proc plot data=perf; plot col3*col1; run;

data impo;
set impo;
i = _n_;
run;

/* plot frekuensi kemunculan masing2 peubah di generasi terakhir*/
proc plot data=impo; plot col1*i; run;
quit;
/* daftar peubah yang muncul 40 kali atau lebih di generasi terakhir*/
proc print data=impo; where col1 >= 40; run;
```