Deteksi *Outlier* Menggunakan Algoritma *Block-based Nested Loop* (Studi Kasus: Data Akademik Mahasiswa Prodi PS Universitas XYZ)

Fiona Endah Kwa[1], Paulina H. Prima Rosa[2]
[1, 2]Jurusan Teknik Informatika, Universitas Sanata Dharma Yogyakarta
fiona.endah@gmail.com, rosa@usd.ac.id

***Abstrak* — Dalam makalah ini dijabarkan salah satu metode *data mining* yaitu deteksi *outlier* pada data numerik dengan mengimplementasikan algoritma *Block-based Nested-Loop*. Sistem yang dibangun diujikan terhadap 3 jenis dataset yang merupakan data akademik mahasiswa Prodi PS Angkatan 2007 dan 2008 dari Universitas XYZ.**

Kesimpulan dari penelitian ini adalah : Sistem Deteksi *Outlier* Menggunakan Algoritma *Block-based Nested-Loop* yang merupakan alat bantu yang digunakan untuk melakukan deteksi *outlier* pada sekumpulan data numerik telah berhasil dibangun; algoritma *Block-based Nested-Loop* terbukti dapat mendeteksi *outlier* dalam *dataset* dan hasilnya disetujui oleh Ketua Program Studi PS; pemilihan nilai atribut M dan D sangat berpengaruh terhadap hasil deteksi *outlier*; dan penggunaan blok-blok data mempercepat proses deteksi *outlier*.

***Kata kunci* — Deteksi *outlier, Block-based Nested-Loop, data mining*.**

I. PENDAHULUAN

Penelitian dalam bidang pendidikan menggunakan teknik penambangan data telah banyak dilakukan saat ini. Penambangan data dalam bidang pendidikan (*educational data mining*) berfokus pada pengembangan metode-metode untuk mengekstrak *knowledge* dari data pendidikan. Data pendidikan dapat berupa data pribadi atau akademik. Selain itu data pendidikan juga dapat berasal dari *e-learning system* yang memiliki data dalam jumlah besar yang digunakan oleh banyak institusi [1].

Salah satu teknik dalam penambangan data yang menarik adalah perihal deteksi *outlier*, yang dilakukan untuk menemukan data yang tidak konsisten dengan data lainnya. Data dianggap tidak konsisten (*outlier*) apabila data tersebut tidak memiliki tingkat kemiripan yang sesuai dengan data lainnya [2]. Dengan adanya deteksi *outlier*, dapat dikenali adanya kesalahan dalam memasukkan data, kecurangan dalam menggunakan data (*fraud detection*) atau adanya sebuah kejadian langka yang memiliki makna tertentu dan perlu dianalisis lebih lanjut (*rare events analysis*).

Algoritma *Block-based Nested-Loop* merupakan contoh algoritma yang memiliki kemampuan untuk mendeteksi *outlier* dalam sekumpulan data numerik. Algoritma ini juga memiliki unjuk kerja yang baik saat diimplementasikan pada kumpulan data dengan jumlah atribut yang banyak atau *high dimensional datasets* [3].

Salah satu contoh data numerik yaitu data akademik mahasiswa yang berupa hasil tes masuk universitas dan indeks prestasi semester (IPS). Pada makalah ini diuraikan proses deteksi *outlier* terhadap data akademik mahasiswa yang terdiri atas nilai tes masuk dan Indeks Prestasi Semester (IPS). Maksud penelitian ini adalah untuk mengidentifikasi mahasiswa manakah yang memiliki data akademik yang unik atau berbeda pada tiap semester berdasarkan nilai tes masuk dan IPS.

Berdasarkan hasil deteksi *outlier*, pihak universitas dapat memperoleh informasi mengenai mahasiswa dengan data akademik yang berbeda atau unik dibandingkan mahasiswa lainnya. Data akademik yang unik dapat dihasilkan dari nilai IPS mahasiswa yang sangat tinggi atau sangat rendah pada setiap semester. Selain itu, data akademik yang unik juga berasal dari tinggi rendahnya nilai tes masuk mahasiswa. Berdasarkan data *outlier* tersebut, universitas dapat menindaklanjuti dengan tindakan yang diperlukan bagi para mahasiswa yang diidentifikasi berbeda dari kebanyakan mahasiswa lainnya.

II. DASAR TEORI

A. *Pengertian Outlier*

Outlier dalam sekumpulan data merupakan data yang dianggap tidak mirip atau tidak konsisten dengan data lainnya. *Outlier* merupakan hasil observasi (data pengukuran) dalam suatu kumpulan data yang nilainya sangat berbeda jika dibandingkan dengan sekumpulan data dari pengukuran lain [4]. *Outlier* juga merupakan data yang tidak mengikuti pola umum atau model dari data lainnya yang berada dalam kumpulan yang sama. *Outlier* terlihat berbeda jauh dan tidak konsisten dengan data lain [2].

Munculnya *outlier* dapat disebabkan oleh data pengukuran yang salah, data pengukuran yang berasal dari populasi lain, maupun data yang berasal dari pengukuran yang benar tetapi mewakili peristiwa atau keadaan unik yang jarang terjadi.

Sebagian besar algoritma penambangan data berfokus untuk meminimalkan pengaruh *outlier* atau mengeliminasi *outlier* tersebut. Hal ini dapat mengakibatkan hilangnya informasi penting yang tersembunyi dibalik *outlier* tersebut. Namun *outlier* sesungguhnya dapat menjadi hal yang menarik untuk dianalisis lebih lanjut.

B. *Deteksi Outlier*

Deteksi *outlier* (*outlier detection*) adalah deteksi yang dilakukan pada sekumpulan objek untuk menemukan objek yang memiliki tingkat kemiripan yang sangat rendah

dibandingkan dengan objek lainnya. Deteksi *outlier* umumya digunakan untuk menemukan kejanggalan dalam data, deteksi kecurangan data atau untuk mengetahui adanya pola khusus dalam sekumpulan data. Deteksi *outlier* sering dimanfaatkan untuk mendeteksi kecurangan penggunaan kredit atau layanan telekomunikasi. Deteksi *outlier* juga berguna dalam bidang pemasaran, yaitu untuk mengidentifikasi perilaku belanja konsumen dengan tingkat pendapatan yang tinggi atau rendah. Dalam dunia kesehatan, deteksi *outlier* digunakan untuk menemukan respon yang tidak biasanya atau berbeda terhadap berbagai perawatan kesehatan [2]. Di bidang pendidikan, deteksi *outlier* dapat digunakan untuk mengetahui prestasi akademik mahasiswa yang berbeda secara signifikan dari mahasiswa lainnya dalam universitas yang sama [1].

Ada beberapa metode yang dapat digunakan untuk mendeteksi *outlier*. Metode-metode tersebut dibagi menjadi empat pendekatan yaitu: pendekatan *statistical distribution-based*, pendekatan *distance-based*, pendekatan *density-based local outlier* dan pendekatan *deviation-based* [2].

Menurut pendekatan *distance-based*, sebuah objek *O* dalam *dataset T* merupakan *DB (p,D)-outlier* jika setidaknya ada *p* objek dalam *T* terletak pada jarak yang lebih dari *D* terhadap *O*. Diasumsikan *N* adalah jumlah objek dalam *dataset T*, dan *F* merupakan fungsi jarak yang digunakan untuk menghitung jarak antar objek dalam *T*. Untuk sebuah objek *O*, *D-neighbourhood* dari *O* memuat sekumpulan objek $Q \in T$ yang terletak pada jarak kurang dari *D* terhadap *O* sehingga $\{ Q \in T \mid F(O, Q) \leq D \}$. Nilai *fraction p* merupakan persentase jumlah objek minimum dalam *T* yang harus ada di luar *D-neighbourhood* dari sebuah *outlier*. Diasumsikan *M* merupakan jumlah objek maksimum dalam *D-neighbourhood* dari sebuah *outlier* sehingga $M = N(1 - p)$ [6]. Sebagai contoh, terdapat sebuah *dataset T* dengan jumlah objek *N* sebanyak 100 dan persentase jumlah *outlier* dalam data sekitar 10%. Berdasarkan data tersebut maka persentase data bukan *outlier p* adalah 100 – 10% yaitu 90% atau 0.9. Untuk memperoleh nilai *M* maka menggunakan perhitungan $N(1 - p)$ sehingga $M = 100(1\text{-}0.9)$ yaitu 10. Artinya, pada *dataset T* jumlah objek maksimum dalam *D-neighbourhood* sebuah *outlier* adalah 10 objek. Apabila sebuah objek *A* dalam *dataset T* dapat menemukan lebih dari 10 objek yang jaraknya kurang dari *D* terhadap dirinya maka *A* akan menjadi data bukan *outlier* sebaliknya A akan menjadi *outlier*. Nilai *M* adalah jumlah objek maksimum dalam *D-neighbourhood* sebuah *outlier* sehingga dapat dikurangi menjadi 9, 8, 7 atau bahkan 1 sampai diperoleh hasil *outlier* yang dianggap paling tepat. Pada pendekatan *distance-based* terdapat beberapa algoritma yang dapat digunakan antara lain *Index-Based*, *Nested-Loop* dan *Cell-Based* [2].

Jarak antar dua objek dalam sebuah kumpulan data memiliki arti yang penting dalam deteksi *outlier* menggunakan pendekatan *distance-based*. Jarak antar dua objek menjadi nilai yang akan dibandingkan dengan nilai parameter *D*. Untuk menghitung jarak antar objek harus disesuaikan dengan tipe variabel yang dimiliki. Tipe variabel yang berupa nilai numerik dan sifatnya multidimensi, dapat dihitung jaraknya menggunakan pengukuran *Euclidean distance* [2] yang didefinisikan sebagai berikut

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2} \qquad (1)$$

di mana i = objek pertama, j = objek kedua, x_i = nilai objek pertama, x_j = nilai objek kedua dan n = dimensi objek.

Pendekatan *distance-based* melibatkan penggunaan dua buah parameter sebagai input. Penentuan nilai parameter *p* dan *D* dapat melibatkan beberapa kali percobaan (*trial and error*) hingga mendapatkan nilai yang paling tepat.

C. *Algoritma Block-based Nested Loop*

Algoritma *Block-based Nested-Loop* merupakan pengembangan dari algoritma *Index-Based*. Algoritma ini menghindari penggunaan memori dalam membentuk indeks untuk menemukan seluruh *DB(p,D)-outliers*. Algoritma ini menggunakan prinsip *block-oriented, nested-loop design*.

Diasumsikan total *free memory* dari komputer yang digunakan sebanyak *B* bytes. Algoritma ini selanjutnya membagi *free memory* komputer dalam dua bagian yang disebut *first array* sebanyak ½*B* dan *second array* sebanyak ½*B*. Data kemudian dibagi ke dalam blok-blok tertentu. Algoritma *Nested-Loop* akan mengatur proses masuk keluarnya blok ke dalam memori. Setiap data yang ada dalam blok tertentu akan secara langsung dihitung jaraknya dengan data lainnya. Untuk setiap data *t* dalam *first array*, jumlah tetangganya akan dihitung. Dua data dikatakan saling bertetangga apabila jarak keduanya kurang dari atau sama dengan *D*. Perhitungan jumlah tetangga untuk setiap objek akan berhenti saat nilainya telah melebihi *M* [5]. *M* merupakan jumlah tetangga maksimum dalam *D-neighbourhood* sebuah *outlier*. Jika jumlah tetangga sebuah data telah melebihi *M* artinya data tersebut tidak termasuk *outlier*.

Pseudocode algoritma *Block-based Nested-Loop* adalah sebagai berikut [5]:

1. Isi *first array* (½*B*) dengan sebuah blok dari *T*
2. Untuk setiap objek t_i, pada *first array*, do :
    a. $count_i \leftarrow 0$
    b. Untuk setiap objek t_j dalam *first array*, jika $dist(t_i, t_j) \leq D$;
        Tambahkan nilai $count_i$ dengan 1. Jika $count_i > M$, tandai t_i sebagai *non-outlier* dan proses dilanjutkan ke t_i berikutnya.
3. Selama masih ada blok yang tersisa untuk dibandingkan dengan *first array*, do:
    a. Isi *second array* dengan blok lainnya (tetapi pastikan blok yang belum pernah dimasukkan ke dalam *first array* berada pada urutan terakhir)
    b. Untuk setiap objek t_i dalam *first array* yang belum ditandai (*unmarked*) do:

Untuk setiap objek t_j dalam *second array*, jika dist(t_i ,t_j) ≤ D:

Tambahkan nilai $count_i$ dengan 1. Jika $count_i > M$, tandai t_i sebagai *non-outlier* dan proses dilanjutkan ke t_i berikutnya.

4. Untuk setiap objek t_i dalam *first array* yang belum ditandai (*unmarked*), tandai t_i sebagai *outlier*.
5. Jika blok dalam *second array* sudah pernah dimasukkan ke dalam *first array* sebelumnya, berhenti; jika belum, tukar posisi nama dari *first array* dan *second array* dan kembali ke langkah 2.

III. METODOLOGI

Pada penelitian ini data yang digunakan adalah data akademik mahasiswa Prodi PS Angkatan 2007 dan 2008 Universitas XYZ. Data ini diperoleh dari gudang data akademik mahasiswa hasil penelitian Rosa dkk [6]. Data diperoleh dalam bentuk skrip *.sql*. Dari skrip tersebut, data yang digunakan dalam penelitian ini adalah data nilai hasil seleksi masuk mahasiswa (jalur tes dan jalur prestasi) dan nilai indeks prestasi semester (IPS) pada semester satu sampai semester empat. Total data akademik yang digunakan dalam penelitian ini sebanyak 126 buah.

Data yang didapat mengalami proses sebagaimana diuraikan dalam bagian berikut ini.

D. Pemrosesan Awal

Dalam tahap ini dilakukan eksekusi skrip gudang data yang berisi tabel-tabel dim_angkatan, dim_daftarsmu, dim_fakultas, dim_jeniskel, dim_kabupaten, dim_prodi, dim_prodifaks, dim_statustes dan fact_lengkap2.

E. Seleksi Data

Data yang akan dipakai adalah kolom ips1, ips2, ips3, ips4, ips4, nil11, nil12, nil13, nil14, nil15 dan final, yang seluruhnya berasal dari tabel fact_lengkap2. Kolom-kolom tersebut kemudian diseleksi lagi barisnya yaitu diambil hanya baris dengan sk_prodi = 27, yaitu prodi PS. Data tersebut yang dipilih karena memiliki tipe numerik sehingga sesuai untuk dikenai proses deteksi *outlier* sesuai tujuan penelitian. Tabel I berikut menunjukkan nama data beserta jangkauan nilai setiap data.

TABEL I. NAMA DAN NILAI ATRIBUT DATA MAHASISWA

Nama Atribut	Penjelasan	Nilai
Final	Nilai final mahasiswa saat seleksi penerimaan mahasiswa baru	0 – 100
Nil11	Hasil tes penalaran mekanik	0 – 10
Nil12	Hasil tes penalaran verbal	0 – 10
Nil13	Hasil tes hubungan ruang	0 – 10
Nil14	Hasil tes Bahasa Inggris	0 – 10
Nil15	Hasil tes kemampuan numerik	0 – 10
Ips1	IPS mahasiswa semester 1	0 – 4
Ips2	IPS mahasiswa semester 2	0 – 4
Ips3	IPS mahasiswa semester 3	0 – 4
Ips4	IPS mahasiswa semester 4	0 – 4

F. Transformasi Data

Data numerik yang terpilih memiliki jangkauan nilai yang berbeda, sehingga dilakukan proses transformasi data untuk menyamakan jangkauan nilai, dengan menggunakan metode normalisasi *min-max normalization* [2] dengan rumus sebagai berikut:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A \quad (2)$$

di mana v = nilai awal, min_A = nilai minimum atribut A sebelum normalisasi, max_A = nilai maksimum atribut A sebelum normalisasi, new_max_A = nilai maksimum atribut A setelah normalisasi dan new_min_A = nilai minimum atribut A setelah normalisasi.

G. Penambangan Data

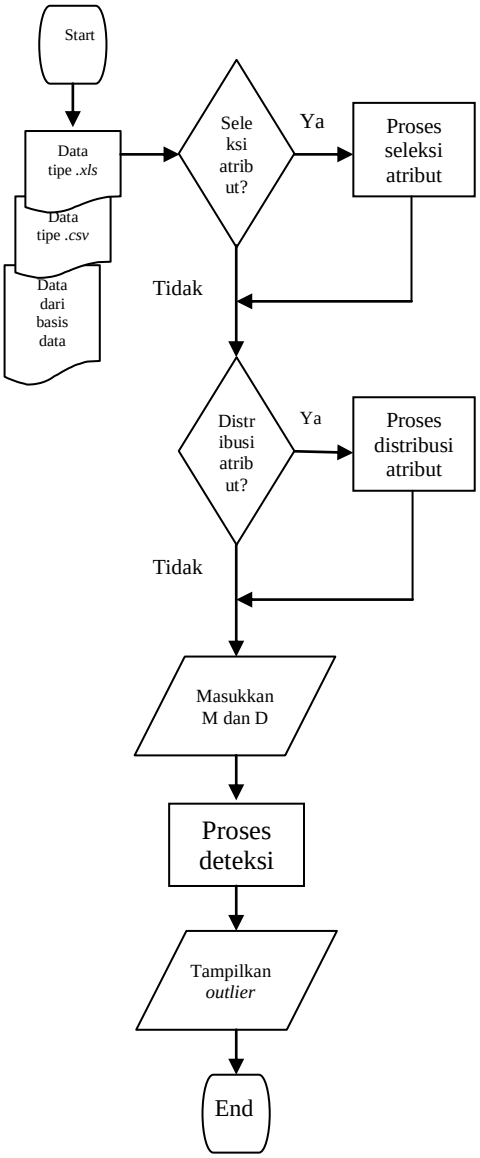
Data yang telah melalui proses transformasi data selanjutnya dicari *outlier*nya menggunakan algoritma *Block-based Nested-Loop*, dengan mengembangkan suatu sistem perangkat lunak deteksi *outlier* dengan deskripsi umum seperti dalam gambar 1.

Selain membutuhkan data berupa file bertipe xls/csv/database, sistem deteksi outlier yang dibangun juga membutuhkan masukan nilai M dan D.

Nilai M menunjukkan jumlah tetangga atau objek maksimum dari sebuah *outlier* dalam ketetanggaan-D [5]. Sebuah data akan dinyatakan sebagai *outlier* apabila jumlah tetangganya kurang dari atau sama dengan M. Sebaliknya, data yang jumlah tetangganya lebih dari M akan dinyatakan sebagai bukan *outlier*.

Nilai D menunjukkan jangkauan (*range*) nilai yang menjadi dasar penentuan dua buah data merupakan tetangga atau tidak berdasarkan jarak (*euclidean distance*) kedua data tersebut. Jika jarak dua data bernilai kurang dari atau sama dengan D, maka keduanya merupakan tetangga.

Sistem akan menghasilkan keluaran berupa sejumlah data yang masuk ke dalam kelompok *outlier*, jika ada, serta waktu yang dibutuhkan untuk melakukan deteksi *outlier*.



Gambar 1. Proses Umum Sistem Deteksi *Outlier* Menggunakan Algoritma *Block-based Nested-Loop*

H. Evaluasi Pola

Pengetahuan atau pola berupa *outlier* yang didapat dari proses deteksi *outlier* akan dievaluasi dengan hipotesa yang telah dibentuk sebelumnya. Hipotesa awal mengenai mahasiswa yang masuk ke dalam kategori *outlier* yaitu mahasiswa dengan data akademik khusus atau unik dilihat dari hasil seleksi masuk dan IPS setiap semester. Kesesuaian hasil deteksi *outlier* dengan hipotesa awal menunjukKan *output* yang baik dari proses penambangan data yang dilakukan.

I. Presentasi Pengetahuan

Pada tahap ini, pola khusus yang dihasilkan (*outlier*) ditampilkan ke dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan melalui pembuatan antarmuka pengguna yang mudah dimengerti.

IV. HASIL DAN PEMBAHASAN

Hasil pengembangan sistem deteksi outlier dengan algoritma *Block-based Nested Loop* dikenai 4 jenis pengujian sebagai berikut:

1. Pengujian *black box*
2. Pengujian *review* dan validasi hasil oleh pengguna.
3. Pengujian efek perubahan nilai atribut penambangan data
4. Pengujian efek penggunaan blok data terhadap waktu deteksi *outlier*

A. Pengujian Black Box

Dalam pengujian ini dilakukan uji aspek fungsional dari perangkat lunak pendeteksi *outlier* berdasarkan diagram use case sistem yang dirancang. Berbagai macam kasus uji berbasis skenario normal dan abnormal dilakukan untuk memastikan bahwa sistem menghasilkan keluaran seperti yang diharapkan.

Berdasarkan hasil pengujian terhadap setiap fungsi sistem dengan berbagai macam kasus uji, disimpulkan bahwa sistem dapat menghasilkan keluaran yang sesuai dengan yang diharapkan pengguna dan melakukan *error handling* terhadap fungsi – fungsi yang tidak berjalan sesuai aturan.

B. Pengujian Review dan Validasi Hasil oleh Pengguna

Pengujian *review* dan validasi oleh pengguna dilakukan dengan meminta Ketua Program Studi (Kaprodi) PS untuk memberikan konfirmasi apakah keluaran sistem berupa daftar mahasiswa yang dianggap *outlier* sungguh-sungguh merupakan data *outlier* di mata Kaprodi.

Untuk keperluan pengujian ini, dipergunakan 2 buah dataset yaitu DATASET1 yang berisi 54 data mahasiswa yang berasal dari jalur prestasi, DATASET2 yang berisi 72 data mahasiswa yang berasal dari jalur tes, dan DATASET3 yang berisi 126 data mahasiswa yang merupakan gabungan data dari jalur prestasi dan jalur tes. Dengan nilai masukan M=5 dan D=1 dihasilkan keluaran sistem sebagai berikut:

TABEL II. HASIL DETEKSI *OUTLIER*

Dataset	Jumlah data	Jumlah *outlier*	No urut data *outlier*
DATASET1	54	6	Mhs-81, Mhs-97, Mhs-100, Mhs-107, Mhs-115, Mhs-119
DATASET2	72	1	Mhs-51
DATASET3	126	3	Mhs-81, Mhs-107, Mhs-115

Berdasar *review* oleh Kaprodi, seluruh data yang dinyatakan *outlier* oleh sistem pada kenyataannya memang dikenali sebagai *outlier* oleh Kaprodi karena mahasiswa pemilik data tersebut memiliki karakteristik nilai yang berbeda dibandingkan mahasiswa lainnya. Beberapa mahasiswa yang dinyatakan *outlier* bahkan akhirnya tidak dapat menyelesaikan studinya. Sekalipun demikian, beberapa mahasiswa yang dinyatakan *outlier* ada pula yang berhasil menyelesaikan studinya. Dalam kasus-kasus semacam itu, mahasiswa tersebut dikenali pernah memiliki masalah yang menghambat studinya, namun pada akhirnya bisa menyelesaikan masalahnya sehingga studi dapat berjalan lancar kembali. Dengan demikian, dapat disimpulkan bahwa sistem deteksi *outlier* ini dapat mengidentifikasi data yang pada kenyataannya memang berbeda dari data lainnya.

C. Pengujian Efek Perubahan Nilai M dan D

Pengujian terhadap Sistem Pendeteksi *Outlier* Menggunakan Algoritma *Nested-Loop* dapat dilakukan dengan mengubah nilai parameter M dan D. Tabel xx berikut ini menunjukkan variasi perubahan jumlah *outlier* akibat adanya perubahan nilai M dan D untuk DATASET1.

TABEL III. PERUBAHAN JUMLAH O*UTLIER* AKIBAT VARIASI PERUBAHAN NILAI M DAN D UNTUK DATASET1

	M = 4	M=5	M = 6	M = 7	M = 8	M = 9
D = 1	42	46	48	51	52	53
D = 2	2	2	3	4	4	4
D = 3	0	0	0	0	0	0
D = 4	0	0	0	0	0	0
D = 5	0	0	0	0	0	0

Berdasarkan hasil uji dalam tabel III, tampak bahwa:

a. Jika nilai M tetap dan nilai D bertambah, jumlah *outlier* akan tetap atau berkurang. Hal ini disebabkan jangkauan nilai (D) yang digunakan sebagai batas nilai sebuah kelompok bukan *outlier* semakin bertambah sedangkan jumlah tetangga (M) yang diperlukan tetap sehingga mengurangi kemunculan *outlier*.
b. Jika nilai D tetap dan nilai M bertambah, jumlah *outlier* akan tetap atau bertambah. Hal ini disebabkan jangkauan nilai (D) yang digunakan sebagai batas nilai sebuah kelompok bukan *outlier* tetap sedangkan jumlah tetangga (M) yang diperlukan bertambah sehingga menambah kemunculan *outlier*.

D. Pengujian Efek Penggunaan Blok Data terhadap Waktu Deteksi Outlier

Algoritma *Block-based Nested Loop* memiliki prinsip *block-oriented*. Data yang diolah menggunakan algoritma ini akan dimasukkan ke dalam blok-blok tertentu. Blok-blok data selanjutnya akan bergantian masuk dan keluar dari memori selama proses deteksi *outlier* berlangsung. Algoritma *Nested-Loop* dikhususkan untuk deteksi *outlier* pada data dalam jumlah besar. Untuk itu dalam pengujian ini dilakukan perbandingan penggunaan 1 blok dan 2 blok terhadap DATASET3 yang memiliki jumlah terbesar dibandingkan kedua dataset lainnya. Hasil pengujian dipaparkan dalam tabel IV di bawah ini. Bagian yang diarsir menunjukkan waktu yang lebih cepat.

TABEL IV. HASIL PENGUJIAN BLOK

Pengujian	**Jumlah Blok**	
	1	2
1	0.042789002 detik	**0.019838321 detik**
2	0.020603497 detik	**0.020215153 detik**
3	0.020371910 detik	0.044256818 detik
4	0.022191141 detik	**0.020106247 detik**
5	0.020390796 detik	0.053300593 detik
6	0.022181388 detik	**0.011608695 detik**
7	0.020535434 detik	0.020707078 detik
8	0.020651433 detik	**0.020180457 detik**
9	0.041769129 detik	**0.020130156 detik**
10	0.020422828 detik	0.022138491 detik
11	0.031415547 detik	**0.020115168 detik**
12	0.020981882 detik	**0.020242460 detik**
13	0.045891224 detik	**0.042321115 detik**
14	0.035943643 detik	**0.020183024 detik**
15	0.035923278 detik	**0.020449623 detik**
16	0.020498897 detik	**0.020199662 detik**
17	0.053786389 detik	**0.020142371 detik**
18	0.020850795 detik	0.041829481 detik
19	0.020401679 detik	0.045726185 detik
20	0.020346456 detik	**0.019651582 detik**

Hasil pengujian di atas menunjukkan bahwa deteksi *outlier* yang dilakukan dengan menggunakan 2 blok data menghasilkan waktu deteksi *outlier* yang lebih cepat dibandingkan dengan yang hanya menggunakan 1 blok data. Dari duapuluh kali pengujian, algoritma *Nested-Loop* yang dijalankan dengan membagi data ke dalam 2 blok ternyata mampu menghasilkan waktu deteksi *outlier* yang lebih cepat dalam empatbelas kali (70%) pengujian. Hal ini membuktikan bahwa penggunaan blok-blok data memiliki pengaruh dalam mengoptimalkan penggunaan memori. Data yang digunakan pada penelitian masih tergolong sedikit untuk kapasitas memori yang tersedia pada komputer yang digunakan sehingga perbandingan lama deteksi *outlier* tidak begitu berbeda antara yang menggunakan 1 blok dan 2 blok. Jika data yang digunakan memiliki jumlah yang lebih besar, perbandingan lama deteksi *outlier* antara yang menggunakan

blok data dengan yang tidak, kemungkinan akan lebih terlihat dengan lebih jelas.

VI. SIMPULAN

Kesimpulan yang diperoleh dari hasil penelitian ini adalah sebagai berikut:

1. Sistem Deteksi *Outlier* Menggunakan Algoritma *Block-based Nested-Loop* yang merupakan alat bantu yang digunakan untuk melakukan deteksi *outlier* pada sekumpulan data numerik telah berhasil dibangun.
2. Algoritma *Block-based Nested-Loop* terbukti dapat mendeteksi *outlier* dalam *dataset*.
3. Pemilihan nilai atribut M dan D sangat berpengaruh terhadap hasil deteksi *outlier*. Jika nilai M tetap dan nilai D bertambah, jumlah *outlier* akan tetap atau berkurang. Jika nilai D tetap dan nilai M bertambah, jumlah *outlier* akan tetap atau bertambah.
4. Penggunaan blok-blok data dalam algoritma *Block-based Nested-Loop* terbukti dapat mempercepat waktu deteksi *outlier* dalam *dataset.*

Beberapa hal yang akan dikerjakan lebih lanjut terkait penilitian tentang topik ini antara lain:

1. Pengembangan sistem agar dapat menerima masukan selain *file* berformat *.xls, .csv* dan tabel dari basis data ,, misalnya file bertipe .txt, .arf dan sebagainya.
2. Pengujian sistem untuk menangani data yang berukuran jauh lebih besar untuk melihat efektifitas penggunaan blok-blok data.

DAFTAR PUSTAKA

[1] M.M.A. Tair and Alaa M. El-Halees, "Mining Educational Data to Improve Students' Performance: A Case Study", *International Journal of Information and Communication Technology Research*, Vol. 2 No. 2, Hal. 140-146, 2012

[2] Jiawei Han and Micheline Kamber, *Data Mining : Concepts and Techniques 2nd ed*, San Francisco, CA: Morgan Kaufmann Publishers, 2006.

[3] A. Ghoting, S. Parthasarathy, M.E. Otey, "Fast Mining of Distance-Based Outliers in High Dimensional Datasets", *Journal Data Mining and Knowledge Discovery*, Vol. 16 Issue 3, hal. 349-364, 2008.

[4] D.M. Hawkins, "*Identification of Outliers".* London: Chapman and Hall, 1980.

[5] E. M. Knorr, and Raymond T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," *Proceedings of the 24rd International Conference on Very Large Data Bases*, hal. 392-403, 1998.

[6] P.H.P. Rosa, Ridowati Gunawan, dan Sri Hartati Wijono. "The Development of Academic Data Warehouse as a Basis for Decision Making : Case Study at XYZ University," *Proceeding of International Conference on Enterprise Information Systems and Applications*, Universitas Islam Indonesia, Yogyakarta, Indonesia, 2013.