ALGORITMA PREDIKSI *OUTLIER* MENGGUNAKAN *BORDER SOLVING SET*

Barry Nuqoba[1)], **Arif Djunaidy**[2)]

[1)]Program Studi Sistem Informasi FST Universitas Airlangga
[2)]Program Studi Sistem Informasi FTIf Institut Teknologi Sepuluh Nopember Surabaya
Email : barry.nu@gmail.com[1)], adjunaidy@its.ac.id[2)]

ABSTRAK

Prediksi outlier penting untuk menjaga validitas data. Algoritma prediksi outlier konvensional memiliki kelemahan dalam hal efisiensi karena harus membandingkan data yang akan diprediksi dengan seluruh data dalam data set. Konsep baru yang melibatkan *solving set* muncul sebagai solusi atas permasalahan efisiensi dalam prediksi outlier. Dengan menggunakan *solving set*, waktu prediksi menjadi lebih cepat tetapi akurasi prediksi menjadi lebih jelek. Dalam penelitian ini dikembangkan suatu algoritma prediksi outlier baru yang efisien dalam melakukan prediksi tetapi tidak mengorbankan akurasi hasil prediksi. Algoritma baru ini merupakan inovasi terhadap konsep *solving set* yang sudah dikembangkan sebelumnya. Dalam penelitian sebelumnya, *solving set* didefinisikan sebagai subset dari data set yang beranggotakan data yang menjadi *top* n-outlier sebagai representasi data set. Sedangkan dalam penelitian ini, *solving set* didefinisikan ulang sebagai subset dari data set yang merupakan data tepi klaster beserta pusat klasternya sebagai representasi data set, atau selanjutnya disebut *border solving set*. Data tepi klaster dideteksi menggunakan algoritma BORDER yang telah terbukti dapat mendeteksi data tepi klaster secara efisien, dan algoritma klasterisasi berbasis hirarki digunakan untuk melakukan klasterisasi data tepi yang telah terdeteksi. Selanjutnya, pusat masing-masing klaster dicari dengan menghitung nilai median dari data tepi pada masing-masing klaster. Algoritma Prediksi Outlier dalam penelitian ini dilakukan dengan membandingkan jarak antara data yang akan diprediksi (*query data)* dengan pusat klaster dan jarak antara *query data* dengan data tepi klaster yang terdekat. Algoritma Prediksi Outlier pada penelitian ini selanjutnya disebut APOTEK (Algoritma Prediksi Outlier menggunakan TEpi Klaster). Setelah dilakukan beberapa percobaan terhadap beberapa data set dengan distribusi normal dan seragam, APOTEK terbukti dapat melakukan perbaikan terhadap algoritma prediksi outlier yang sudah ada sebelumnya. Dalam aspek akurasi prediksi, APOTEK berhasil melakukan peningkatan sebesar 5% dibandingkan dengan algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006), untuk data set berdistribusi normal.

Kata Kunci: Analisis *Outlier*, Prediksi *Outlier*, Data Tepi Klaster, *Solving Set*, Data *Mining*.

PENDAHULUAN

Sebagian besar algoritma terkait dengan outlier yang dipublikasikan oleh para peneliti berhubungan dengan deteksi outlier. Belum banyak yang membahas tentang prediksi outlier. Prediksi outlier penting untuk menjaga validitas data dalam tempat penyimpanan data (repository). Algoritma prediksi outlier konvensional memiliki kelemahan dalam hal efisiensi karena harus membandingkan data yang akan diprediksi dengan seluruh data dalam data set. Angiulli et. al (2006) berhasil menemukan solusi dari permasalahan tersebut dengan memperkenalkan suatu konsep baru, yaitu *solving set*. *Solving set* adalah subset dari data set yang dapat merepresentasikan data set. Angiulli et. al. (2006) mendefinisikan bahwa *solving set* beranggotakan data yang merupakan top n-outlier pada data set. Dengan *solving set*, data yang akan diprediksi cukup dibandingkan dengan data pada *solving set*, tidak dengan seluruh data pada data set. Konsep *solving set* yang ditawarkan oleh Angiulli et. al. (2006) memang berhasil mengatasi masalah efisiensi dalam melakukan prediksi outlier. Sayangnya, di sisi lain penggunaan *solving set* untuk melakukan prediksi outlier mengakibatkan turunnya akurasi hasil prediksi.

Penelitian ini memperbaiki kelemahan algoritma prediksi outlier Angiulli dari sisi akurasi dan sekaligus meningkatkan efisiensi. Algoritma prediksi outlier Angiulli diperbaiki dengan melakukan definisi ulang terhadap konsep *solving set* yang dibuat oleh Angiulli. Selanjutnya, dikembangkan algoritma prediksi outlier dengan menggunakan *solving set* baru tersebut. Angiulli et.

al. (2006) mendefinisikan bahwa anggota *solving set* adalah subset dari data set yang merupakan top n-outlier. Pada penelitian ini, *solving set* didefinisikan sebagai subset dari data set yang beranggotakan data tepi klaster beserta pusat klasternya sebagai representasi data set, selanjutnya disebut dengan *border solving set*. Algoritma prediksi outlier pada penelitian ini terdiri dari dua proses utama, yaitu pembentukan *border solving set* dan prediksi outlier menggunakan *border solving set*. Proses pembentukan *border solving set* terdiri dari beberapa langkah. Langkah pertama yaitu data tepi klaster dideteksi menggunakan algoritma BORDER (Xia et. al., 2006). Selanjutnya, algoritma Single Link Hierarchical Klastering (Lubsa, 2005) digunakan untuk klasterisasi data tepi. Kemudian pusat dari masing-masing klaster dicari dengan menghitung nilai median dari data tepi pada masing-masing klaster. Data tepi beserta pusat klasternya inilah yang menjadi anggota *border solving set*. Algoritma prediksi outlier pada penelitian ini bertumpu pada *border solving set* tersebut. Oleh karena itu, selanjutnya algoritma pada penelitian ini disebut dengan APOTEK (Algoritma Prediksi Outlier menggunakan TEpi Klaster).

LANDASAN TEORI

k-Nearest Neighbor (kNN)

Algoritma *k-nearest neighbor* (kNN) adalah sebuah metode untuk melakukan klasifikasi terhadap obyek berdasarkan data pembelajaran yang jaraknya paling dekat dengan obyek tersebut. Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas *c* jika kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat titk tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean.

Pada fase pembelajaran, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi dari data pembelajaran. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data test (yang klasifikasinya tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor data pembelajaran dihitung, dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Nilai *k* yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai *k* yang tinggi mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, *k* = 1) disebut algoritma *nearest neighbor*.

Ketepatan algoritma kNN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik.

Algoritma kNN memiliki konsistensi yang kuat. Ketika jumlah data mendekati tak hingga, algoritma ini menjamin *error rate* yang tidak lebih dari dua kali *Bayes error rate* (*error rate* minimum untuk distribusi data tertentu).

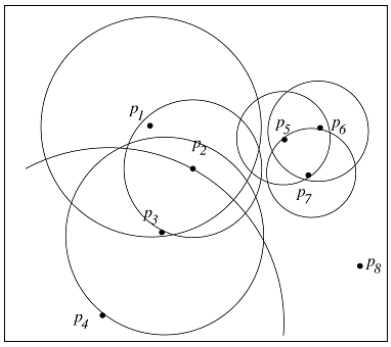
Reverse k-Nearest Neighbor (RkNN)

Reverse k-Nearest neighbor (RkNN) telah menjadi pusat perhatian dalam beberapa tahun terakhir ini. Beberapa penelitian telah menyorot pentingnya reverse k-nearest neighbor (RkNN) dalam *decision support systems*, *profile-based marketing*, *document repositories*, dan manajemen peralatan mobile. Secara formal, RkNN didefinisikan sebagai berikut:

Definisi (Reverse k-Nearest Neighbor). Diberikan suatu data set DB, suatu titik query p, suatu bilangan bulat positif k, dan suatu metrik jarak l(), reverse k nearest neighbors dari p, disimbolkan dengan $RkNN_p(k)$, adalah suatu himpunan titik p_i sedemikian hingga $p_i \in$ DB dan untuk setiap p_i, p $\in$ $kNN_{pi}(k)$, dimana $kNN_{pi}(k)$ adalah k-nearest neighbors dari titik p_i.

RkNN mempunyai sifat-sifat unik yang berbeda dengan k-nearest neighbors (kNN) yang konvensional:

- RkNN tidak terlokalisasi pada *neighborhood* dari titik query.
- Kardinalitas RkNN dari suatu titik bervariasi tergantung pada distribusi data.



Gambar 1. Contoh RkNN

Gambar 2.1 memberikan suatu contoh dari RkNN. Misalkan p_2 adalah titik query dan k=2. Dari diagram, terlihat bahwa p_2 adalah satu dari 2-nearest neighbors titik p_1, p_3, dan p_4. Oleh karena itu, reverse 2-nearest neighbors dari p_2 adalah p_1, p_3, dan p_4. Catat bahwa, meskipun p_4 letaknya jauh dari titik query p_2, itu masih suatu R2NN dari p_2.

Sebaliknya, p_5 dan p_7 dekat dengan p_2, tetapi mereka bukanlah jawaban dari query R2NN untuk p_2. Lebih lanjut, p_2 mempunyai tiga reverse 2-nearest neighbors, sementara p_4 dan p_8 mempunyai 0 reverse 2-nearest neighbor.

Sifat RkNN tersebut mempunyai potensi yang besar untuk diterapkan dalam area data mining. Bagaimanapun, kompleksitas dari RkNN menjadi suatu kendala. Solusi naïf untuk mencari RkNN, pertama menghitung k nearest neighbors untuk tiap titik p dalam data set, kemudian meretrieve titik-titik yang memiliki q sebagai salah satu k nearest neighborsnya dan mengeluarkannya sebagai jawaban. Kompleksitas dari solusi naïf tersebut adalah O(NlogN) untuk data yang terindex dengan menggunakan beberapa struruktur hirarki, seperti R-tree. Sedangkan untuk data yang tidak terindex kompleksitasnya adalah $O(N^2)$, dimana N adalah kardinalitas dari data set.

Metode untuk memproses RkNN query mengutitlisasi sifat-sifat geometris dari RkNN untuk mencari sejumlah kecil titik data sebagai kandidat dan kemudian memverifikasinya dengan nearest-neighbor query atau range query. Bagaimanapun, teknik-teknik tersebut tidak *scalable* dengan dimensi data dan nilai k. Sebagai tambahan, analisa data yang mengutilisasi RkNN biasanya menjalankan sekumpulan RkNN query untuk semua titik dalam suatu data set. Oleh karena itu, metode-metode untuk suatu RkNN query tunggal adalah tidak efisien untuk RkNN query yang berorientasi himpunan. Pendekatan BORDER mengutilisasi suatu operasi efisien yang disebut dengan kNN-Join untuk menghitung RkNN query.

k-Nearest Neighbor Join (kNN Join)

kNN-join mengkombinasikan tiap titik dalam suatu data set R dengan k-nearest neighbors nya dalam data set S yang lain. R disebut dengan outer data set (atau query data set dimana semua titik-titik dalam R adalah titik-titik query) dan S disebut sebagai inner data set. Ketika outer dan inner data set sama, kNN-join juga dikenal sebagai kNN self-join. kNN-join dapat digunakan untuk mendukung berbagai aplikasi dimana terdapat banyak kNN query yang terlibat.

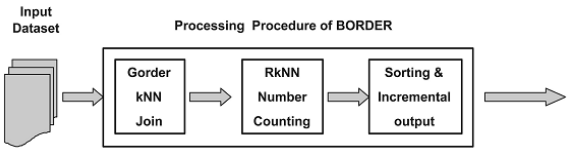
Definisi (kNN-join). Diberikan dua data set R dan S, suatu bilangan bulat k, dan similarity metric dist(), kNN-join dari R dan S, disimbolkan sebagai R kNN S, mengembalikan sepasang titik (p_i,q_j) sedemikian hingga p_i adalah dari outer data set R dan q_j dari inner data set S dan q_j adalah satu k-nearest neighbors dari pi. kNN join dapat digunakan untuk menjawab RkNN query yang berorientasi himpunan (the RkNN join).

kNN join dapat digunakan untuk menjawab RkNN query yang berorientasi himpunan (the RkNN join). Berikut adalah relasi yang ada antara kNNjoin dan RkNN join:

Lemma. *Reverse k-nearest neighbors dari semua titik dalam data set DB dapat diturunkan dari k-nearest neighbors dari semua titik dalam DB. Dengan membalik semua pasangan (pi, pj) yang dihasilkan oleh kNN join, didapatkan himpunan lengkap dari pasangan (pj, pi) dimana pi adalah reverse k nearest neighbor dari pj.*

Algoritma BORDER

Titik-titik batas adalah titik-titik data yang terletak pada tepi distribusi data yang padat, misalnya klaster. Algoritma BORDER bertujuan untuk mendeteksi titik-titik pada data set yang merupakan titik batas. Ide dasar dari algoritma ini adalah bahwa titik-titik yang merupakan titik batas cenderung memiliki jumlah RkNN yang kecil.



Gambar 2. Prosedur Algoritma BORDER

Dari gambar 2, terlihat bahwa algoritma BORDER terdiri dari tiga bagian utama, yaitu:

1. Suatu operasi kNN-join dengan Gorder untuk menemukan k-nearest neighbors untuk tiap titik dalam data set.
2. Suatu RkNN counter untuk mendapatkan jumlah RkNN untuk tiap titik (kardinalitas himpunan jawaban RkNN untuk tiap titik).
3. Titik-titik diurutkan berdasarkan jumlah RkNN nya. Titik-titik yang jumlah kNN nya lebih kecil dari suatu nilai threshold yang ditentukan oleh user akan dikeluarkan secara urut sebagai titik-titik batas.

Dalam penelitian yang dilakukan oleh Xia et. al (2006) telah dibuktikan bahwa algoritma BORDER dapat bekerja secara efektif. Berikut adalah hasil evaluasi yang telah dilakukan sehingga dapat disimpulkan bahwa algoritma BORDER dapat bekerja secara efektif dalam melakukan deteksi titik-titik batas klaster.

Algoritma Hierarchical Clustering

Algoritma hierarchical clustering pertama kali didefinisikan oleh S.C. Johnson dalam Hierarchical Clustering Schemes, Psychometrika 1967. Diberikan suatu himpunan dari N item yang akan diklaster, dan suatu matriks jarak N * N, proses dasar dari hierarchical clustering adalah sebagai berikut:

Langkah 1: Tentukan k sebagai jumlah klaster yang akan dibentuk

Langkah 2: Dimulai dengan menjadikan tiap-tiap item menjadi satu klaster, jadi jika kamu mempunyai N item, kamu sekarang punya N klaster, tiap klaster hanya berisi satu item. Jarak antar klaster adalah sama dengan jarak dari item yang di dalamnya.

Langkah 3: Cari pasangan klaster yang paling dekat dan satukan mereka menjadi satu klaster. Jadi sekarang kamu memiliki kurang satu klaster.

Langkah 4: Hitung jarak antara klaster baru dengan tiap-tiap klaster yang lama.

Langkah 5: Jika n>k, ulangi dari langkah 2

Algoritma Angiulli

Angiulli et. al. (2006) memperkenalkan suatu konsep baru yang disebut dengan *solving set*. *Solving set* adalah subset dari data set yang dapat merepresentasikan data set secara keseluruhan. *Solving set* yang diusulkan beranggotakan titik-titik data yang merupakan top-n outlier dalam data set.

Diberikan suatu data set D yang berisi obyek-obyek, suatu obyek p dapat diasosiasikan dengan suatu bobot atau skor, yaitu suatu fungsi dari jarak k nearest-neighbors. Secara intuitif, bobot mengukur seberapa besar suatu obyek tidak mirip dengan tetangga-tetangganya. Jika w* adalah bobot terbesar ke-n dari suatu obyek dalam D. Suatu outlier terhadap D adalah suatu obyek yang memiliki bobot lebih besar atau sama dengan w* terhadap D.

Algoritma prediksi outlier yang diusulkan oleh Angiulli et. al dimulai dengan mencari titik-titik data pada data set yang merupakan top n-outlier, yaitu n titik dalam data set yang memiliki bobot terbesar. Titik-titik tersebut selanjutnya disebut sebagai anggota *solving set*. Selanjutnya, prediksi outlier dilakukan untuk mengetahui apakah data baru yang masuk merupakan outlier atau bukan. Caranya yaitu dengan menghitung bobot dari obyek yang baru masuk terhadap *solving set*. Apabila bobotnya melebihi w* maka obyek tersebut merupakan outlier.

Algoritma SOLVINGSET berperan dalam pembentukan *solving set*. Algoritma tersebut menghitung bobot-bobot dari obyek data set dengan cara membandingkan tiap-tiap obyek dengan subset yang terpilih dari keseluruhan data set, disebut juga Cand, dan menyimpan k-neighbours terhadap Cand yang sudah ditemukan sejauh ini. Bobot terkini dari suatu obyek adalah batas atas terhadap bobot sebenarnya dari obyek tersebut. Obyek-obyek yang memiliki suatu bobot lebih rendah dari bobot terbesar ke-n yang sudah dihitung sejauh ini disebut nonaktif, sementara yang lain disebut aktif. Saat permulaan, Cand berisi obyek-obyek yang dipilih secara acak dari D, sementara pada setiap tahapnya, Cand dibangun dengan cara memilih dari titik-titik aktif dalam data set yang belum dipilih menjadi anggota Cand pada tahap sebelumnya.

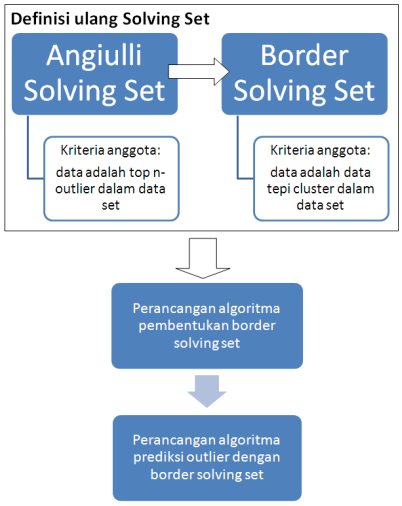
Selama eksekusi, apabila suatu obyek menjadi nonaktif maka obyek tersebut tidak lagi dipertimbangkan dalam proses pembentukan Cand, karena obyek tersebut tidak mungkin merupakan outlier. Algoritma berhenti saat tidak ada lagi obyek yang dapat diperiksa, yaitu semua obyek yang belum dimasukkan ke dalam Cand adalah nonaktif. Dengan demikian Cand menjadi kosong. *Solving set* adalah gabungan (union) dari himpunan Cand yang diperoleh pada setiap tahap. Kompleksitas dari algoritma SOLVINGSET adalah O(D2).

METODA PENELITIAN

Desain Algoritma

Sebagaimana yang terlihat pada gambar3, dalam rangka menyusun algoritma prediksi outlier menggunakan *border solving set*, ada 3 tahap yang dilakukan, yaitu:

1. Definisi ulang terhadap konsep *solving set*
2. Perancangan algoritma untuk membentuk *solving set* yang sudah didefiniskan ulang (*border solving set*)
3. Perancangan algoritma prediksi outlier dengan menggunakan *border solving set*



Gambar 3. Alur Desain Algoritma

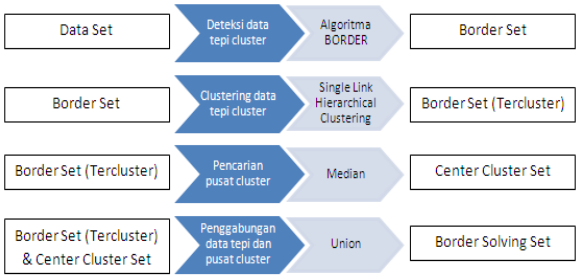
Definisi Ulang Solving Set

Sebagaimana terlihat pada Gambar 3, definisi ulang terhadap konsep *solving set* dilakukan dengan mengubah kriteria untuk menjadi anggota *solving set*. Angiulli mendefinisikan bahwa data yang ada dalam data set menjadi anggota *solving set* apabila merupakan top n-outlier. Sedangkan pada penelitian ini, kriteria untuk menjadi anggota *solving set* adalah data tersebut haruslah merupakan data tepi klaster.

Perubahan kriteria dalam penentuan anggota *solving set* menyebabkan perubahan pada algoritma pembentukan *solving set*. Oleh karena itu, penelitian ini merancang algoritma untuk pembentukan *border solving set*. Berikut adalah proses yang ada dalam pembentukan *border solving set*.

Pembentukan Border Solving Set

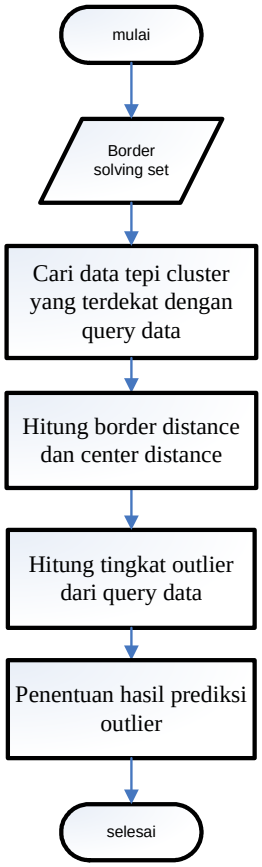
Ada 4 proses yang harus dilalui agar dapat membentuk *border solving set*. Gambar 4 berikut memperlihatkan rangkaian proses, metode yang digunakan, serta hasil tiap tahapan proses dalam pembentukan *border solving set*.



Gambar 4. Pembentukan *border solving set*

Perancangan Algoritma Prediksi Outlier

Algoritma prediksi outlier dirancang dengan menggunakan *border solving set* yang sudah terbentuk pada tahap sebelumnya. Gambar 5 berikut menampilkan proses dalam melakukan prediksi outlier dengan menggunakan *border solving set.*

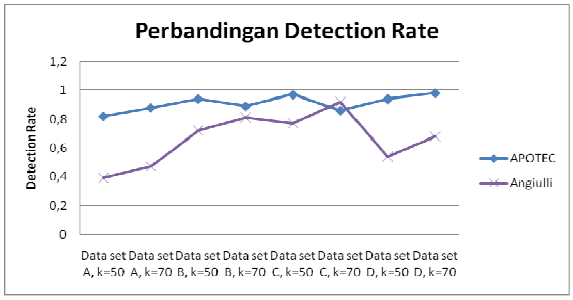


Gambar 5. Prediksi Outlier menggunakan *border solving set*

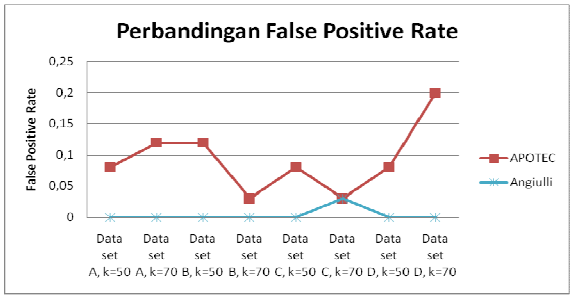
HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan hasil dan pembahasan terhadap ujicoba yang dilakukan dalam penelitian. Uji coba dilakukan untuk mengevaluasi akurasi algoritma yang dikembangkan dalam penelitian ini dibandingkan dengan algoritma sebelumnya (Angiulli et.al). Ujicoba dilakukan pada beberapa data set sintetis dengan sebaran data normal yang digenerate dengan menggunakan MATLAB. Algoritma prediksi outlier pada penelitian ini, dan algoritma prediksi outlier yang dikembangkan oleh Angiulli at. al. (2006) serta algoritma prediksi outlier *non-solving set* akan dijalankan sesuai dengan skenario ujicoba yang sudah dirancang. Hasil pelaksanaan skenario ujicoba tersebut akan menjadi tolok ukur keberhasilan APOTEK dalam melakukan perbaikan terhadap algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006).

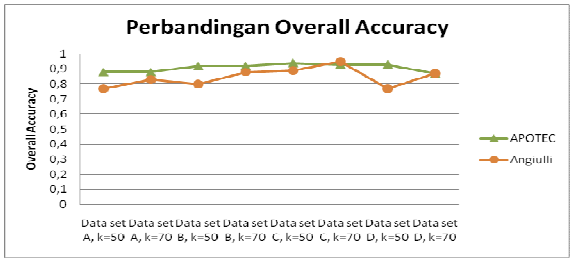
Gambar 6,7, dan 8 berikut menampilkan hasil ujicoba yang membandingkan detection rate, false positive rate, dan overall accuracy dari Algoritma penelitian ini dan algoritma Angiulli.



Gambar 6. Perbandingan Detection Rate



Gambar 7. Perbandingan False Positive Rate



Gambar 8. Perbandingan Overall Accuracy

1. Rata-rata detection rate APOTEK sebesar 91% sedangkan algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) sebesar 66%.
2. Rata-rata false positive rate APOTEK sebesar 9% sedangkan algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) sebesar 0%. Dalam hal rata-rata false positive rate, APOTEK memang sedikit lebih buruk dari algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006). Hal ini adalah dampak logis dari jauh lebih tingginya rata-rata detection rate APOTEK dari algoritma prediksi outlier yang dikembangkan oleh Angiulli. Oleh karena kelebihan rata-rata detection rate.
3. Rata-rata keakuratan total dari APOTEK sebesar 91% sedangkan algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) sebesar 85%.

Fakta-fakta di atas menunjukkan bahwa APOTEK lebih baik dari algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) untuk data set berdistribusi normal. Di samping itu, kurva yang ada pada Gambar 8 juga menunjukkan bahwa APOTEK memiliki tingkat keakuratan total yang lebih baik dari algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) untuk semua data set berdistribusi normal yang digunakan dalam ujicoba.

KESIMPULAN DAN SARAN

Kesimpulan

Dari hasil uji coba dapat ditarik kesimpulan sebagai berikut:

a. Algoritma prediksi outlier yang dikembangkan dalam penelitian ini, yaitu APOTEK telah berhasil memperbaiki algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006).
b. Hasil ujicoba akurasi prediksi untuk data set berdistribusi normal menunjukkan bahwa APOTEK menghasilkan rata-rata akurasi sebesar 91%, sedangkan algoritma prediksi outlier yang dikembangkan oleh Angiulli et. al. (2006) sebesar 85%.

Saran

Algoritma dalam penelitian ini sudah teruji bekerja dengan baik pada data set berdistribusi normal. Selanjutnya akan dikembangkan lagi sehingga algoritma ini juga dapat bekerja dengan baik pada jenis data set berdistribusi yang lain.

DAFTAR PUSTAKA

[1] Chenyi Xia, Wynne Hsu, Mong Li Lee dan Beng Chin Ooi, (2006), "BORDER: Efficient Computation of Boundary Points", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, hal. 289-303.

[2] Fabrizio Angiulli, Stefano Basta dan Clara Pizzutti, (2006), "Distance-Based Detection and Prediction of Outliers", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, hal. 145-160.

[3] Hui Xiong, Gaurav Pandey, Michael Steinbach dan Vipin Kumar, (2006), "Enhancing Data Analysis with Noise Removal", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, hal. 304-319.

[4] Lubsa, Dana Avram, (2005), "Unsupervised Single-Link Hierarchical Clustering", *Studia Univ. Babes-Bolyai, Informatica*, vol. 1, no. 2.

[5] Pang-Ning Tan, Michael Steinbach dan Vipin Kumar, (2006), *Introduction to Data Mining*, Pearson Education, Inc., Boston.