Perbandingan Algoritme *Pruning* pada *Decision Tree* yang Dikembangkan dengan Algoritme CART

Martin Budi, Rindang Karyadin, Sony Hartono Wijaya

Departemen Ilmu Komputer, Institut Pertanian Bogor, Jl. Meranti Wing 20 Lv.V, Bogor, Jawa Barat, 16680

Abstract---Pruning is part of the development of decision tree. As decision tree is developed, some nodes became outliers as the results of noise data. Implementation of the decision tree pruning, can reduce the noise and outliers on the initial decision tree so that it can improve the accuracy of the data classification. Therefore the selection of proper pruning algorithm needs to be done to get the maximum results of the classification.

This experiment uses data from the company's customer credit providers. The data obtained from the data bank at the University of California. Data used in this experiment has twenty variables with two classes and 1000 instances. The data contain thirteen qualitative variables and the rest is a numeric data. The data is a good for use because it does not have a missing value.

The experiment compared three pruning algorithm, Cost Complexity Pruning (CCP), Reduced Error Pruning (REP), Error Based Pruning (EBP). Those pruning algorithms do prune to the decision tree that was developed with the Classification and Regression Tree (CART) algorithm. The comparison of those algorithms is done repeatedly on the data with different conditions both in terms of the instance number and the data variables. Comparison of the algorithm includes a comparison of the accuracy of the decision tree, and the process time of pruning algorithm.

The experiment's result shows the average error rate of that the REP algorithm will produce the smallest error rate. Although the error rate of REP algorithm is the smallest, the difference value between ERP's and EBP's error rate is only 0.5%. Even though they have almost similar error rate, EBP algorithm proposes more simple decision tree than REP algorithm does.

Keyword : Decision tree, Classification and Regression Tree (CART), Cost Complexity Pruning (CCP), Reduced Error Pruning (REP), Error Based Pruning (EBP).

PENDAHULUAN

Data mining merupakan salah satu tahapan dalam proses *Knowledge Discovery in Database* (KDD) yang melakukan ekstraksi informasi atau pola penting dalam data berukuran besar (Han & Kamber 2006). Teknik yang dapat digunakan pada implementasi *data mining* adalah klasifikasi dan prediksi, *association rule*, dan *clustering*. Klasifikasi merupakan metode yang berfungsi untuk menemukan model yang membedakan kelas data, sehingga klasifikasi dapat memperkirakan label kelas dari suatu objek yang belum diketahui. Salah satu metode klasifikasi yang sering digunakan adalah *decision tree*.

Pruning merupakan bagian dari proses pembentukan *decision tree*. Saat pembentukan *decision tree*, beberapa *node* merupakan
outlier maupun hasil dari *noise* data. Penerapan *pruning* pada *decision tree,* dapat mengurangi *outlier* maupun *noise* data pada *decision tree* awal sehingga dapat meningkatkan akurasi pada klasifikasi data (Han & Kamber 2006). Oleh sebab itu pemilihan algoritme *pruning* yang tepat perlu dilakukan untuk mendapat hasil klasifikasi yang maksimal.

Pada penelitian yang dilakukan oleh (Esposito *et al.* 1997), algoritme Reduced Error Pruning (REP) disimpulkan sebagai algoritme yang menghasilkan *subtree* terkecil dengan *error rate* minimum. Penelitian Esposito *et al.* 1997) menggunakan algoritme C4.5 untuk membangun *decision tree* yang di-*pruning*. Berbeda dengan penelitian sebelumnya, penelitian ini membandingkan pengunaan algoritme *pruning* pada *decision tree* yang dibangun dengan algoritme Classification and Regression Tree (CART). Algoritme CART biasa menggunakan Cost Complexity Pruning (CCP) sebagai algoritme *pruning*-nya. Pada penelitian ini algoritme *pruning* CCP dibandingkan dengan dua algortime *pruning* lain yaitu REP dan Error Based Pruning (EBP).

Penelitian in bertujuan untuk menerapkan teknik CCP , REP dan EBP pada metode klasifikasi *decision tree* dengan algoritme CART. Selain itu penelitian in ijuga membandingkan nilai akurasi dari *decision tree* yang terbentuk, serta waktu proses yang dihasilkan oleh algoritme *pruning* CCP , REP dan EBP.

METODE PENELITIAN

Penelitian ini menerapkan tahapan yang tertuang dalam suatu Metodologi Penelitian (Gambar 1).

A. Studi Literatur

Studi literatur dilakukan dengan memperdalam algoritme-algoritme yang akan dibandingkan. Informasi yang diperoleh berasal dari beberapa sumber seperti : jurnal, buku dan artikel di internet.

B. Pengumpulan Data

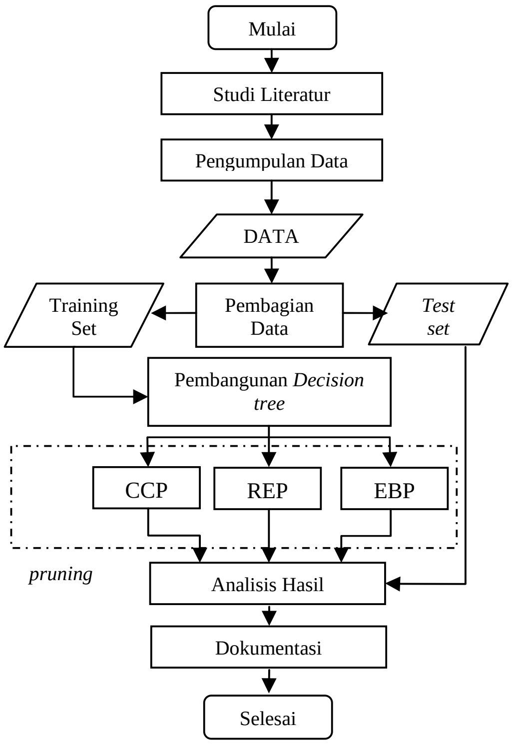
Penelitian ini menggunakan data *profile* pelanggan dari perusahaan penyedia kredit. Data tersebut diperoleh dari bank data pada University of California (Asuncion & Newman 2007). Contoh data dapat dilihat pada Lampiran

C. *Data Mining*

Teknik *data mining* dengan metode *decision tree* terdiri dari dua tahapan, yaitu:

1. Pembentukan *Tree*

Pada tahap ini akan dibentuk suatu *tree* yang terdiri dari *node* awal, *leaf* sebagai distribusi kelas dan batang yang menggambarkan hasil keluaran dari pengujian.



Gambar 1. Metode penelitian.

Pembentukan tree dilakukan dengan menerapkan algoritme CART.CART adalah metode yang menggunakan data histori untuk membangun sebuah decision tree. Decision tree yang dihasilkan kemudian digunakan untuk klasifikasi data yang baru (Timofeev 2004).

Metode CART pertama kali dikemukakan oleh Breiman pada tahun 1984. Metode CART menggunakan metode decision tree yang hanya memiliki cabang dua buah, atau yang biasa disebut dengan binary tree (Larose 2005).

CART melakukan pemisahan secara rekursif nilai dari *record* data menjadi bagian-bagian data yang memiliki kesamaan. CART membentuk pohon dengan mencari ke semua variabel untuk mencari tempat percabangan yang paling optimal dengan menggunakan Persamaan 1.

$$\text{Gini}_A(D) = \sum_{i=1}^{k} \frac{D_i}{D} \text{Gini}(D_i) \ldots\ldots\ldots\ldots\ldots\ldots..(1)$$

dengan:

$$\text{Gini}(D_i) = 1 - \sum_{j=1}^{m} \frac{D_j}{D_i}$$

$\text{Gini}_A(D)$ = nilai Gini dari data jika dipartisi dengan parameter A.

k = jumlah pembagian data, k = 2 (CART)

$\frac{D_i}{D}$ = nilai perbandingan jumlah data D dengan jumlah data partisi ke-i.

m = jumlah kelas data yang ada.

$\frac{D_j}{D_i}$ = nilai perbandingan jumlah data D_i dengan jumlah data kelas ke-j .

Atribut dengan nilai GiniA(D) paling kecil adalah atribut yang akan digunakan.

2. Pemangkasan *Tree*

Pemangkasan *tree* dilakukan dengan menggunakan tiga algoritme yaitu :

a. CCP

Proses awal dari algoritme *Cost complexity pruning* adalah menentukan nilai alpha, yaitu derajat kompleksitas sebuah pohon keputusan. Perhitungan nilai alpha menggunakan Persamaan 2 (Quinlan 1987).

$$\alpha = \frac{r(t) - r(T_t)}{|n_{T_t}| - 1} \ldots\ldots\ldots..\ldots(2)$$

dengan:

$$\mathrm{r}(\mathrm{T_t}) = \frac{\sum_{s \in \mathrm{T_t}} \mathrm{e(s)}}{\sum_{s \in \mathrm{T_t}} \mathrm{n(s)}} \ldots\ldots\ldots\ldots..(3)$$

dengan:

r(T_t) = *error rate* pada *subtree* T_t.

$e(s)$ = jumlah data dengan kelas minoritas pada *node* s.

$n(s)$ = jumlah data pada *node* s.

s = *node* anggota *subtree* T_t.

$$r(t) = \frac{e(t)}{n(t)} \ldots\ldots\ldots\ldots\ldots\ldots...(4)$$

dengan:

$r(t)$ = *error rate* dari *node* t.

$e(t)$ = jumlah data dengan kelas minoritas pada *node* t.

$n(t)$ = jumlah data pada *node* t.

α = parameter kompleksitas,

$|n_{T_t}|$ = jumlah *leaf node* pada pohon T_t.

Pada setiap *internal node* dilakukan perhitungan nilai alpha dengan menggunakan persamaan di atas. *Internal node* dengan nilai alpha yang paling kecil akan dipangkas. Pohon yang dipangkas akan dihitung nilai *error rate*. Hal tersebut diperoleh dari *test set* yang dimasukkan ke dalam aturan pohon keputusan. Perhitungan alpha terus dilakukan hingga tidak terdapat lagi *internal node* pada pohon keputusan.

Seluruh nilai *error rate* dari pemangkasan dengan nilai alpha yang mendekati serupa satu dengan yang lain akan dijumlahkan dan dihitung nilai rata-ratanya. Nilai rata-rata tersebut akan dibandingkan dengan seluruh nilai rataan *misclassification error* yang ada. Nilai alpha dengan *error rate* paling kecil akan digunakan sebagai alpha pada pemangkasan pohon keputusan.

b. REP

REP merupakan salah satu algoritme *pruning* yang dikemukakan oleh Quinlan (1987). Algoritme ini membagi data menjadi dua yaitu *train set* dan *test set.* Setiap *internal node* pada pohon yang dihasilkan, dihitung berapa nilai *error rate internal node* tersebut menggunakan Persamaan 3. Kemudian dengan Persamaan 4 dihitung nilai *error rate node* tersebut apabila *node* merupakan *leaf node.* Hasil perhitungan keduanya dibandingkan, dan *pruning* dilakukan jika *error rate* hasil Persamaan 4 lebih kecil daripada *error rate* Persamaan 3.

Apabila perubahan status pada *node dari internal node* menjadi *leaf node* memiliki nilai *error rate* yang sama atau lebih rendah daripada jika *node* tersebut menjadi *internal node* maka *pruning* dilakukan. Proses tersebut terus dilakukan hingga terbentuk pohon keputusan dengan *error rate* yang terbaik dan jumlah aturan yang optimal (Quinlan 1987).

c. EBP

Algoritme EBP biasa digunakan pada algoritme *decision tree* c4.5 . Algoritme ini mengizinkan pergantian *subtree* dengan salah satu dari *leaf node*-nya untuk membuat *decision tree* yang lebih simpel.

EBP mulai melakukan *pruning* pada *internal node* dari bagian bawah *decision tree*. Pemeriksaan setiap *internal node* dilakukan dengan menggunakan Persamaan 5.

$$e`(t) < e`(T_t) + std\big(e`(\,T_t)\big) \ldots \quad (5)$$

dengan:

$e`$(t) = *error rate internal node*,

$e`(T_t)$ = *error rate subtree* dengan *internal node t* (Ripley 1996).

$$std\big(e`(\,T_t)\big) \;=\; [e`(T_t) * (n(t) - e`(T_t)/n(t))]^{1/2}$$

dengan :

$n(t)$ = jumlah *node* yang diperiksa *error rate*-nya.

Apabila nilai *error rate* menjadi lebih kecil maka *pruning subtree* dilakukan (Quinlan 1992).

D. Analisis Hasil

Analisis dilakukan dengan membandingkan hasil *decision tree* yang dilakukan oleh algoritme *pruning* CCP, REP, EBP. Analisis dilakukan berulang-ulang pada data dengan jumlah *instance* dan variabel yang berbeda-beda. Hal yang diamati adalah nilai *error rate* dari *decision tree*, jumlah *node* yang dikurangi, serta waktu eksekusi masing-masing algoritme.

HASIL DAN PEMBAHASAN

A. Data

Data yang digunakan pada penelitian ini memiliki 20 variabel dengan dua buah kelas dan berjumlah 1000 *instance*. Dari 20 variabel yang ada pada data, 13 variabel merupakan data kualitatif dan sisanya merupakan data bertipe numerik. Data tersebut sudah baik karena tidak memiliki *missing value.*

Pada penelitian ini tidak dilakukan data *preprocessing. Preprocessing* tidak dilakukan karena data yang digunakan sudah mengalami proses tersebut.

Data dengan jumlah *instance* 5000 dan 10000 mendapat tambahan data yang berasal dari pembangkitan data secara acak. Walaupun dibangkitkan secara acak, nilai variabel kelas dari setiap *instance* yang ada diperoleh dari klasifikasi berdasarkan *decision tree* yang dibangun dengan data asli.

Proses awal dari Penelitian ini adalah pembangunan *decision tree*. *Decision tree* dibangun dengan menggunakan algoritme CART.

1. Pembentukan *Tree*

Pembentukan tree dilakukan dengan membagi data menjadi 4 bagian (S_1,..,S_4) yang memiliki jumlah merata. Secara bergantian tiga bagian data digunakan untuk membangun *decision tree* dan satu bagian lainnya sebagai data *testing*.

Pembentukan tree dilakukan berulang-ulang yaitu pada data dengan jumlah *instance* 250, 500, 1000, 5000, dan 10000. Selain itu pembentukan *tree* juga dilakukan pada data dengan jumlah variabel yang berbeda-beda. Pembentukan *tree* dilakukan pada data dengan jumlah variabel yang dikurangi sebanyak 1, 3, 5, 7, 10, dan 15 variabel.

2. Pemangkasan *Tree*

Pemangkasan tree dilakukan dengan menggunakan tiga algoritme, yaitu CCP, REP dan EBP. Penerapan algoritme CCP agak sedikit berbeda dengan dua algoritme lainnnya. Sebelum melakukan *pruning,* algoritme CCP terlebih dahulu menentukan nilai alpha. Penentuan nilai alpha dimulai dengan membagi data menjadi 10 bagian (S_1,….,S_{10}) dengan algoritme 10-*folds cross.* Secara bergantian sembilan bagian data digunakan untuk membangun *decision tree* dan satu bagian yang lain sebagai data *testing.* Dari perlakuan tersebut, CCP akan menghasilkan berbagai nilai alpha serta *error rate*, dari seluruh nilai tersebut dicari nilai alpha dengan *error rate* minimum. Nilai alpha yang akan digunakan sebagai standar *pruning* algoritme CCP.

Berbeda dengan CCP, algoritme REP dan EBP tidak perlu menentukan nilai khusus sebagai standar *pruning*. Algoritme REP dan EBP langsung melakukan *pruning* pada *decision tree* yang dihasilkan.

B. Analisis Hasil

Analisis hasil percobaan dilakukan dengan membandingkan hasil *decision tree* yang dilakukan oleh algoritme *pruning* CCP, REP, EBP.

Rataan nilai *error rate* dari *decision tree* hasil *pruning* dengan tiga buah algoritme serta *error rate decision tree* sebelum mengalami *pruning* disajikan pada Tabel 1.Dari Tabel 1 terlihat bahwa *error rate* setelah dilakukan *pruning* untuk data dengan jumlah *instance* kurang dari sama dengan 1000 lebih baik daripada nilai *error rate* sebelum dilakukan *pruning.* Namun untuk data dengan jumlah *instance* lebih dari 1000, terlihat bahwa nilai *error rate decision tree* awal lebih baik daripada nilai *error rate* setelah *tree* di-*pruning*.

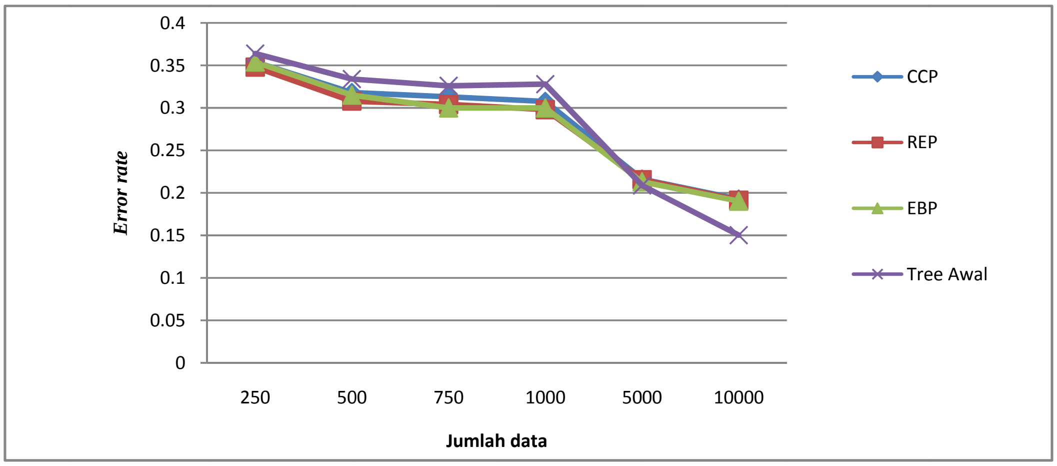
Secara global nilai *error rate* yang paling baik untuk data dengan *instance* kurang dari sama dengan 1000 dihasilkan oleh algoritme REP. Untuk data dengan jumlah *instance* lebih dari 1000, nilai *error rate* yang paling baik dihasilkan oleh algoritme EBP.

Data dengan jumlah 5000 dan 10000 diperoleh dengan cara dibangkitkan secara acak dari data yang ada. Nilai rataan ragam dari data awal adalah 996015.15. Sedangkan nilai ragam untuk 5000 dan 10000 adalah 4154527.8 dan 4142692.469. Nilai ragam dari data tersebut lebih besar dari nilai ragam data awal. Nilai ragam yang lebih besar menunjukkan bahwa data tersebut kurang seragam.

Keragamaan yang besar cenderung akan meningkatkan probabilitas terjadinya kesalahan. Oleh sebab itu nilai *error rate* pada suatu *decision tree* yang dibangun pada satu bagian data belum tentu mewakili bagian data yang lain. Karena itu perlu diterapkan standar deviasi agar nilai *error rate* yang dihasilkan lebih relevan. EBP menerapkan standar deviasi pada proses *pruning*nya dengan tujuan untuk memperoleh *error rate* yang lebih baik.

Tabel 1. *Error rate* pada *decision tree* dengan berbagai jumlah *instance* data

Algoritme *Pruning*	***Error Rate***					
	250 data	**500 data**	**750 data**	**1000 data**	**5000 data**	**10000 data**
CCP	0.3543	0.318	0.3133	0.3076	0.2194	0.1944
REP	**0.3481**	**0.3076**	0.3037	**0.2984**	0.2187	0.1931
EBP	0.3544	0.3152	**0.2997**	0.3004	**0.2169**	**0.1918**
Tree Awal	0.3639	0.3344	0.3255	0.3322	**0.2123**	**0.1506**



Gambar 2. Error rate pada decision tree dengan berbagai jumlah instance data.

Terlihat pada Tabel 1 nilai *error rate* pada data dengan nilai ragam yang besar yaitu 5000 dan 10000 data, *error rate* EBP lebih baik daripada *error rate* algoritme lainnya.

Gambar 2 merupakan grafik *error rate* pada *decision tree* dengan berbagai jumlah *instance* data. Pada Gambar 2 terlihat bahwa nilai *error rate* ketiga algoritme *pruning* semakin baik sejalan dengan jumlah data yang semakin bertambah

Pruning pada data dengan *instance* lebih dari jumlah *instance* data asli cenderung mengalami *overprune*. Hal tersebut terlihat dari nilai *error rate* yang dihasilkan lebih besar daripada *error rate decision tree* awal. *Overprune* dapat disebabkan oleh pendugaan yang salah pada saat penentuan nilai kelas data saat pembangkitan data secara *acak*. *Decision tree* yang digunakan untuk klasifikasi dibangun menggunakan data yang asli yang berjumlah 1000 *instance*. Sedangkan jumlah data yang akan diklasifikasikan berjumlah lebih dari 1000 *instance*. Hal tersebut yang memungkinkan terjadinya kesalahan klasifikasi yang mengakibatkan terjadinya *overprune*.

Gambar 3 merupakan grafik rataan nilai *error rate* pada *decision tree* dengan berbagai jumlah *instance* data. Pada grafik terlihat bahwa secara global nilai *error rate decision tree* yang mengalami *pruning* lebih baik daripada *decision tree* awal. Selain itu, Gambar 3 juga menunjukkan bahwa algoritme REP menghasilkan *decision tree* dengan *error rate* yang paling rendah dibandingkan dengan algoritme lain.

Gambar 3. Rataan error rate pada decision tree dengan berbagai jumlah instance data.

Serupa dengan Tabel 1, Tabel 2 juga memperlihatkan nilai *error rate* dari *decision tree* hasil *pruning* dengan tiga buah algoritme dan *error rate decision tree* sebelum mengalami *pruning*. Berbeda dengan Tabel 1, data yang digunakan untuk membangun *decision tree* pada Tabel 2 merupakan data dengan jumlah *instance* 1000. Serta jumlah variabel yang digunakan berbeda-beda dari jumlah variabel data asli.

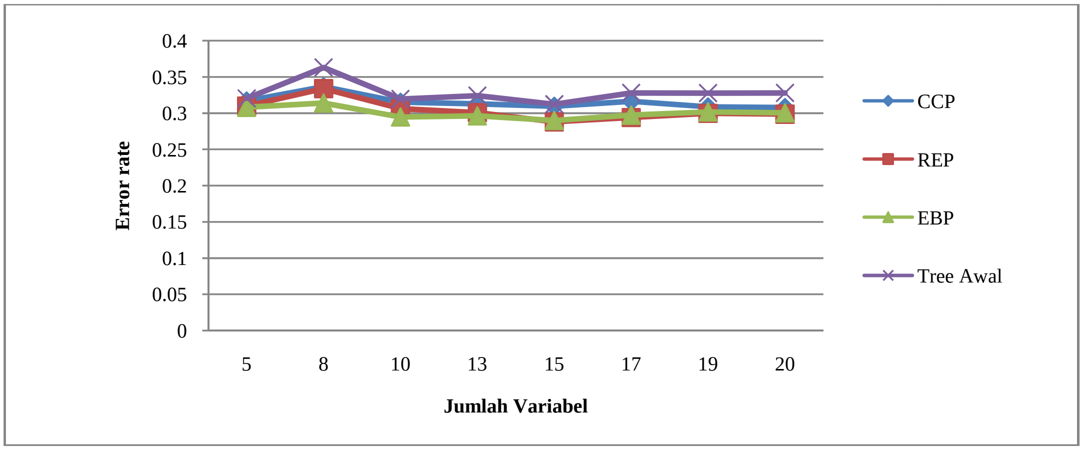
Pada Tabel 2 terlihat bahwa untuk data dengan jumlah variabel lebih dari 5 untuk ketiga algoritme *pruning* memiliki nilai *error rate* yang lebih baik daripada nilai *error rate decision tree* awal. Sedangkan pada data dengan 5 variabel, *error rate* paling kecil dihasilkan oleh *decision tree* awal yang kemudian diikuti oleh algoritme EBP, sebagai algoritme yang menghasilkan *error rate* terkecil setelah *decision tree* awal.

Semakin kecil nilai variabel akan berpengaruh pada jumlah *node* yang dihasilkan oleh *decision tree*. Semakin sedikit *node* sama saja dengan semakin sedikit aturan yang terbentuk pada *decision tree* tersebut. Jumlah aturan pada *decision tree* berpengaruh pada keakuratan klasifikasi *decision tree*. Hal tersebut berpengaruh pada proses pemilahan data saat pembangunan *decision tree.* Pemilahan data yang kurang baik menyebabkan peningkatan nilai *error rate* pada setiap *node*. Hal tersebut yang menyebabkan terjadinya *overprune*.

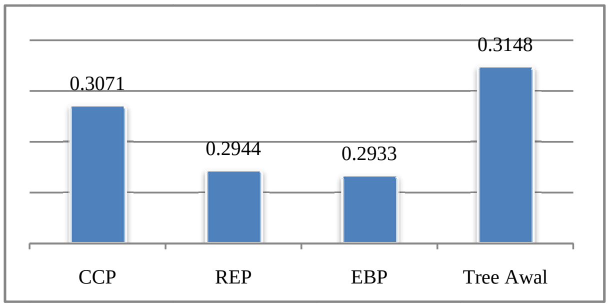
Gambar 4 menunjukkan bahwa nilai *error rate* yang paling baik untuk data dengan jumlah variabel lebih dari sama dengan 15 dihasilkan oleh algoritme REP. Untuk data dengan jumlah variabel kurang dari 15, nilai *error rate* yang paling baik dihasilkan oleh algoritme EBP.Sedangkan pada data dengan 5 variabel terjadi *overprune*, hal tersebut terlihat dari nilai *error rate* ketiga algoritme yang lebih besar daripada nilai *decision tree* awal. Dengan demikian pada kondisi tersebut, *decision tree* yang paling baik dihasilkan apabila *decision tree* tersebut tidak mengalami *pruning*.

Tabel 2. *Error rate* pada *decision tree* dengan berbagai jumlah variabel data

Algoritme *Pruning*	***Error Rate***							
	20 variabel	**19 variabel**	**17 variabel**	**15 variabel**	**13 variabel**	**10 variabel**	**8 variabel**	**5 variabel**
CCP	0.3076	0.3086	0.3164	0.3094	0.3128	0.315	0.3364	0.3168
REP	**0.2984**	**0.2996**	**0.2938**	**0.2878**	0.3006	0.3062	0.3338	0.3096
EBP	0.3004	0.3016	0.2972	0.2898	**0.2962**	**0.2946**	**0.3138**	**0.308**
Tree Awal	0.3278	0.3276	0.3278	0.3122	0.324	0.3194	0.3628	**0.3202**



Gambar 4. *Error rate* pada *decision tree* dengan berbagai jumlah variabel data.

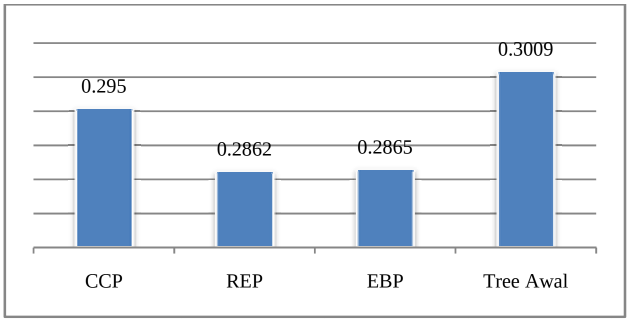


Gambar 5. Rataan *error rate* pada *decision tree* dengan berbagai jumlah variabel data.

Gambar 5 adalah grafik rataan nilai *error rate* pada *decision tree* dengan berbagai jumlah variabel data. Pada grafik terlihat bahwa nilai rataan *error rate decision tree* yang mengalami *pruning* lebih baik daripada *decision tree* awal. Hal tersebut tidak berbeda dengan percobaan yang dilakukan pada data dengan berbagai jumlah *instance*.

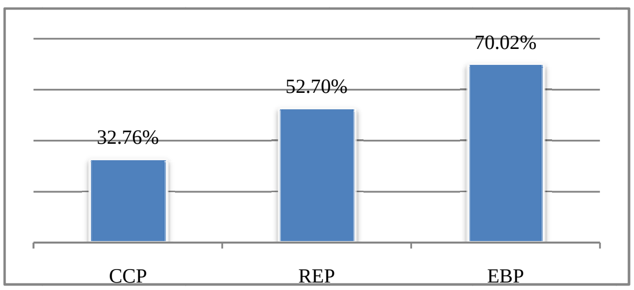
Gambar 5 juga menunjukkan bahwa rataan algoritme EBP menghasilkan *decision tree* dengan nilai *error rate* yang paling rendah dibandingkan dengan algoritme lain. Dengan demikian *pruning* dengan algoritme EBP yang dilakukan pada data dengan jumlah variabel yang berbeda, akan cenderung memiliki nilai *error rate* yang lebih baik.

Hasil rataan *error rate* secara keseluruhan dapat dilihat pada Gambar 6. Gambar 6 mewakili seluruh *error rate* yang dihasilkan, baik percobaan dengan jumlah *instance* data yang berbeda-beda maupun percobaan dengan berbagai jumlah variabel. Gambar 6 menunjukkan nilai yang tidak jauh berbeda dengan Gambar 3. Pada Gambar 6 terlihat bahwa rataan nilai *error rate decision tree* yang mengalami *pruning* lebih baik daripada *decision tree* awal.



Gambar 6. Rataan *error rate* keseluruhan perulangan.

Gambar 6 juga menunjukkan bahwa algoritme REP menghasilkan decision tree dengan error rate yang paling rendah dibandingkan dengan algoritme lain. Gambar 6 menunjukkan bahwa nilai error rate yang dihasilkan algoritme REP dan EBP tidak jauh berbeda.



Gambar 7. Rataan selisih node.

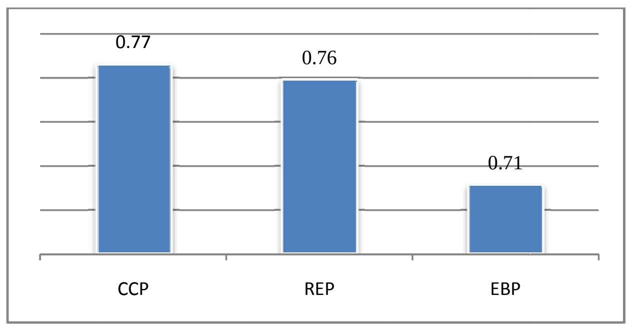
Namun dari Gambar 7 dapat dilihat bahwa rataan *node* yang di *prune* oleh masing-masing algoritme cukup jauh berbeda. Pada Gambar 7 terlihat bahwa rata-rata algoritme EBP mampu memangkas kurang lebih 70 persen dari jumlah keseluruhan *node*. Hal tersebut cukup berbeda dengan algoritme REP yang rata-rata hanya memangkas 52 persen dari *node* yang ada. Sehingga dapat dikatakan bahwa dengan nilai *error rate* yang tidak jauh berbeda, algoritme EBP mampu memangkas *node* lebih banyak daripada algoritme REP. Semakin banyak *node* yang berkurang pada *decision tree* maka akan cenderung semakin cepat *decision tree* tersebut melakukan klasifikasi.

Gambar 8 merupakan grafik nilai rataan waktu klasifikasi masing-masing *decision tree* yang telah mengalami *pruning*. Gambar 8 menunjukkan bahwa *decision tree* yang menggunakan algoritme EBP sebagai algoritme *pruning* memiliki waktu klasifikasi paling cepat.

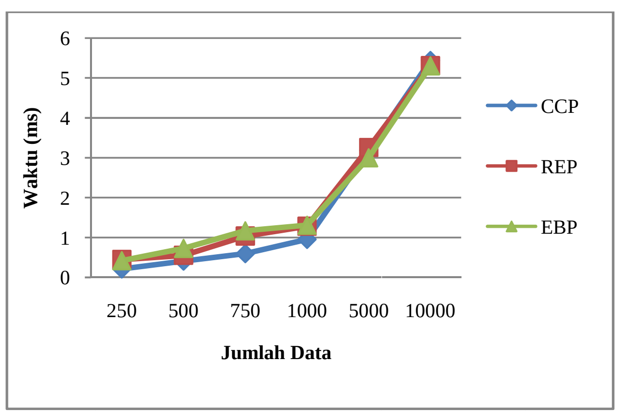
Algoritme EBP merupakan algoritme yang menghasilkan *decision tree* paling ringkas dibandingkan dengan dua algoritme lain. Sedangkan algoritme CCP merupakan algoritme yang menyisakan *node* paling banyak setelah proses *pruning.* Dengan konsidi tersebut, *decision tree* hasil *pruning* dengan algoritme CCP merupakan *decision tree* dengan waktu klasifikasi terlama. Sehingga dapat disimpulkan bahwa jumlah *node* pada *decision tree* akan berpengaruh pada waktu klasifikasi *decision tree* tersebut.

Selain mengukur *error rate* dan selisih *node*, perbandingan ketiga algoritme dapat dilihat dari waktu eksekusi ketiga algoritme *tersebut*. Waktu yang digunakan oleh masing-masing algoritme dapat dilihat pada Gambar 9 dan 10. Gambar 9 menunjukkan waktu eksekusi algoritme *pruning* pada data dengan jumlah *instance* yang berbeda-beda. Walaupun Gambar 8 memperlihatkan bahwa masing-masing algoritme cenderung memiliki waktu eksekusi yang serupa, data dengan jumlah *instance* 750 dan 1000 menunjukkan bahwa waktu eksekusi algoritme CCP lebih cepat dibandingkan dua algoritme lainnya.

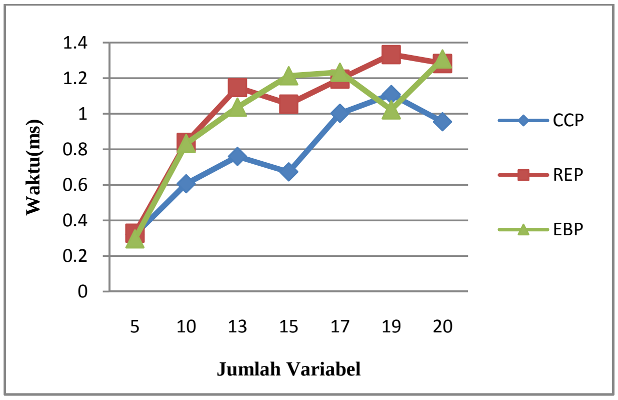
Nilai waktu eksekusi algoritme *pruning* pada data dengan jumlah variabel berbeda-beda dapat dilihat pada Gambar 10. Pada Gambar 10 terlihat bahwa meningkatnya jumlah variabel pada data akan berpengaruh pada waktu eksekusi algoritme *pruning.* Semakin banyak variabel pada data maka semakin banyak waktu yang digunakan untuk proses *pruning*. Pada grafik terlihat bahwa algoritme CCP memiliki waktu eksekusi yang lebih cepat daripada algoritme lainnya hampir pada setiap perulangan kecuali pada data dengan 5 dan 19 variabel.



Gambar 8. Rataan waktu klasifikasi *decision tree* yang telah di *pruning.*



Gambar 9. Waktu eksekusi pada data dengan berbagai jumlah *instance*.



Gambar 10. Waktu eksekusi pada data dengan berbagai variabel.

KESIMPULAN DAN SARAN

A. Kesimpulan

Penelitian ini menerapkan ketiga algoritme *pruning* pada *decision tree* yang dibangun dengan algoritme CART. Masing-masing algoritme *pruning* menghasilkan *decision tree* yang lebih simpel.

Hasil penelitian yang dilakukan menunjukkan bahwa pada rataan *error rate* seluruh percobaan, algoritme REP akan menghasilkan *error rate* paling kecil. Hasil tersebut tidak berbeda dengan penelitian sebelumnya oleh Esposito et al. (1997). Walaupun *error rate* algoritme REP lebih kecil, *error rate* tersebut hanya berbeda 0.5% dengan nilai *error rate* algoritme EBP. Dengan nilai *error rate* yang mendekati serupa, EBP menghasilkan *decision tree* yang jauh lebih simpel daripada algoritme REP.

B. Saran

Saran yang dapat dilakukan pada penelitian selanjutnya ialah :

1. Perbandingan algoritme *pruning* dilakukan pada data dengan kelas data lebih dari dua.
2. Melakukan perbandingan algoritme *pruning* pada *decision tree* dengan algoritme lainnya seperti *Supervised Learning In Quest* (SLIQ) atau *Scalable Parallelizable Induction of Decision Tree* (SPRINT), kemudian hasilnya bisa dibandingkan dengan penelitian ini.

DAFTAR PUSTAKA

Asuncion A. & Newman DJ. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

Esposito F, Donato M, Giovanni S. 1997. A Comparative Analysis of Methods for Pruning Decision Trees [catatan penelitian]. IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 19, hlm. 476-491.

Gehrke J, Ramakrishnan R, Ganti V. 1998. RainForest – A Framework for Fast Decision Tree Construction of Large Database [skripsi]. Madison : Department of Computer Sciences , University of Wisconsin.

Han J, Kamber M. 2006. *Data Mining: Concepts and Techniques*. Ed ke-2. USA: Academic Press.

Kantardzic M. 2003. *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-Interscience.

Kohavi R. 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Di dalam: Proceedings of the International Joint Conference on Artificial Intelligence; 1995. San Mateo, CA : Morgan Kaufmann. hlm 1137-1143.

Larose TD. 2005. *Discovering Knowledge in Data : an Introduction to Data Mining*. USA : John Wiley & Sons, Inc.

Lewis RJ. 2000. An Introduction to Classification and Regression Tree (CART) Analysis [tesis]. California : Department of Emergency Medicine Harbor, UCLA Medical Center.

Quinlan JR. 1987. Simplifying Decision Trees [catatan penelitian]. *International Journal of Man-Machine Studies* vol. 27, hlm. 221-234.

Quinlan JR. 1992. C4.5: *Programs for Machine Learning*. San Mateo, CA:Morgan Kaufmann.

Ripley BD. 1996. *Pattern Recognition and Neural Networks*. Cambridge : Cambridge University Press.

Timofeev R. 2004. Classification and Regression Trees (CART) Theory and Application [tesis]. Berlin : Center of Applied Statistics and Economics, Humboldt University.