Jurnal Elektronik Ilmu Komputer Udayana
Volume 9, No 1. August 2020

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Penerapan Metode Adaboost Untuk Multi-Label Classification Pada Dokumen Teks

I Gede Angga Purnajiwa Arimbawa[a1], Dr. Ngurah Agus Sanjaya ER, S.Kom., M.Kom [a2]

[a]Department Computer Science, Udayana University Bali, Indonesia

[1]arimbawaangga@gmail.com
[2]agus_sanjaya@unud.ac.id

Abstract

Peningkatan jumlah data teks yang signifikan menjadikan alasan untuk menerapkan klasifikasi terhadap teks menjadi sangat jelas. Proses klasifikasi manual yang dilakukan manusia sangat tidak efisien dan efektif. Keterbatasan ini membuka peluang besar pada pengembangan klasifikasi teks secara otomatis. Pada kasus klasifikasi artikel lebih relevan menggunakan klasifikasi multi-label, dikarenakan sebuah artikel dapat dikategorikan ke dalam banyak label. Banyak pendekatan yang dapat digunakan untuk mengimplementasikan klasifikasi multi-label pada teks. Metode supervised-learning dalam bidang machine learning adalah cara yang populer untuk permasalahan ini. Dalam tinjauan yang dilakukan, terdapat jurnal yang melakukan analisis komparatif terhadap metode supervised dalam klasifikasi multi-label. Berdasarkan tinjauan yang dilakukan, Algoritma *AdaBoos*t memberikan hasil yang lebih baik dibandingkan algoritma lainnya. Penelitian ini memiliki tujuan untuk mengetahui hasil dan performa dari algoritma *AdaBoost* dengan memanfaatkan dataset artikel komputer berbahasa inggris. Proses penelitian ini dimulai dari pengumpulan data artikel, *text processing*, klasifikasi dan evaluasi. Hasil dan performa algoritma *AdaBoost* akan dibandingkan dengan 2 algoritma klasifikasi multi-label lainnya. Berdasarkan penelitian yang dilakukan algoritma *AdaBoost* memberikan hasil lebih optimal pada dataset dengan pembobotan TF-IDF dibandingkan TF. Hasil *accuracy,precision, recall* dan *f-measure* yang diberikan lebih tinggi jika dibandingkan dengan algoritma pembanding yang digunakan. Waktu komputasi yang digunakan algoritma *AdaBoost* lebih cepat dibandingkan algoritma pembanding yang digunakan.

Keywords: *AdaBoost,Klasifikasi,Multi-label, Artikel*

1. Introduction

Penggunaan internet yang semakin meningkat menyebabkan pertumbuhan konten di internet sangat cepat dan pesat. Seperti contoh Wikipedia Indonesia yang selalu mengalami peningkatan pertumbuhan artikel, saat ini jumlah artikel yang dimiliki Wikipedia Indonesia mencapai 48,320,034 dengan lebih dari satu juta label yang dihasilkan oleh kurator [1].

Dalam banyak bidang penelitian, data yang berlabel mungkin berjumlah sedikit dan tidak mencukupi dibandingkan data yang berlabel. Dalam bidang tekstual, alasan untuk menerapkan klasifikasi terhadap teks menjadi sangat jelas, dikarenakan pertumbuhan data teks seperti artikel akademik, artikel berita, buku manual, e-mail, buku atau data teks lainnya yang semakin banyak dari hari ke hari. Pertumbuhan yang terjadi lebih cepat dibandingkan kemampuan pengguna informasi untuk mencari, mencerna dan menggunakannya.

Klasifikasi data merupakan salah satu cara untuk membantu pengguna informasi. Klasifikasi adalah proses untuk menemukan model yang dapat membedakan kelas data, dengan tujuan untuk dapat

memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Proses klasifikasi manual yang dilakukan manusia sangat tidak efisien dan efektif, bukan hanya karena permasalahan biaya dan waktu, kurator manual berpotensi menghasilkan label klasifikasi yang beragam dan tidak memadai. Ditambah lagi dengan kenyataan bahwa data teks lebih sulit untuk dimengerti dan dikategorikan karena hubungan antara runtutan kata dan kontennya tidak jelas dibandingkan dengan data angka. Keterbatasan dan kendala ini membuka peluang besar pada pengembangan klasifikasi teks secara otomatis.

Pada kasus klasifikasi artikel, klasifikasi multi-label menjadi lebih relevan dibandingkan klasifikasi binary tradisional. Seperti contoh artikel “Boosting algorithm: AdaBoost” pada website Medium.com memiliki label Machine Learnig, Data Science dan Algorithms. Klasifikasi multi-label adalah sebuah permasalahan klasifikasi yang memungkinkan untuk mengasosiasikan sebuah data ke dalam beberapa label, sedangkan klasifikasi binary adalah sebuah permasalahan klasifikasi untuk mengasosiasikan sebuah data ke dalam sebuah label.

Banyak pendekatan yang dapat digunakan untuk mengimplementasikan klasifikasi multi-label pada teks. Metode supervised-learning dalam bidang machine learning adalah cara yang populer untuk permasalahan ini. Penyelesain masalah klasifikasi multi-label dengan pendekatan supervised-learning dapat dibagi menjadi 2 yaitu problem transformation dan algorithm adaptation. Problem transformation adalah pendekatan dengan cara merubah permasalahan multi-label menjadi satu atau beberapa bermasalahan single-label, sedangkan algorithm adaptation adalah sebuah pendekatan dengan memodifikasi algoritma secara langsung untuk membuat prediksi multi-label.

Dalam tinjauan yang dilakukan, terdapat jurnal yang melakukan analisis komparatif terhadap metode supervised dalam klasifikasi multi-label. Terdapat 11 metode supervised problem transformation dan 5 metode algorithm adaptation yang dianalisis. Dari perbandingan yang dilakukan, Algoritma *AdaBoost* memberikan hasil yang lebih baik dibandingkan algoritma lainnya. *AdaBoost* mampu meningkatkan akurasi dan meminimalkan Hamming loss error [2].

Dalam penelitian ini, peneliti ingin mengetahui hasil dan performa dari algoritma *AdaBoost* dengan memanfaatkan dataset yang berbeda dari tinjauan yang dilakukan yaitu berupa artikel komputer berbahasa inggris. Peneliti menggunakan 2 metode pembobotan fitur pada dataset yaitu TF & TF-IDF, alasan penggunaan 2 metode pembobotan fitur dikarenakan tidak terdapat referensi yang menyatakan pembobotan fitur terbaik untuk permasalahan klasifikasi multi-label, sehingga akan dilakukan perbandingan hasil antara 2 pembobotan fitur untuk mengetahui pembobotan fitur yang dapat memberikan hasil lebih optimal terhadap algoritma AdaBoost.

2. Metode Penelitian

Pada bagian metodologi penelitian ini menjelaskan gambaran langkah-langkah yang akan dilakukan dalam menjalankan penelitian ini, langkah-langkah tersebut meliputi pengumpulan data, alur metodologi penelitian, tahap *preprocessing*, tahap klasifikasi artikel dan tahap pengujian.

2.1. Pengumpulan Data

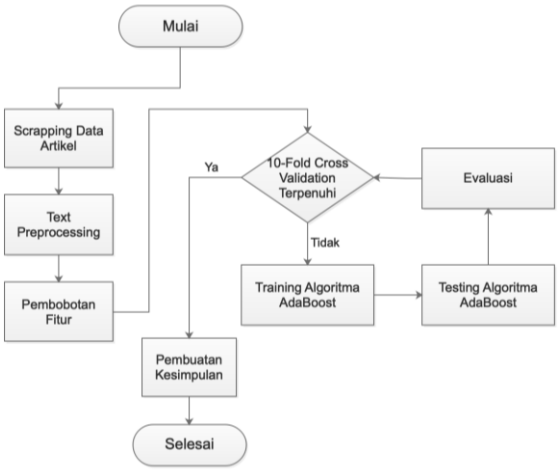
Pada pengumpulan data, data yang digunakan adalah data artikel dari website *medium.com*. Untuk mendapatkan data artikel akan dilakukan proses *scrapping*. Pengambilan data dari website *medium.com* dikarenakan data yang disediakan sudah diberikan *tag* oleh kurator atau penulis, *tag* tersebut dimanfaatkan sebagai label pada dataset. Data yang diambil adalah judul artikel, isi artikel dan label artikel. Setiap artikel bisa memiliki lebih dari 1 label. Data yang digunakan sebanyak 500 data yang dibagi 90% untuk training dan 10% untuk data testing . Setelah data – data tersebut terkumpul, lalu akan disimpan ke dalam *database.*

2.2. Alur Penelitian

Pada bagian ini akan digambarkan alur secara umum dari penelitian yang akan dilakukan penulis, yaitu dimulai dari pengumpulan data artikel dari medium.com yang kemudian setiap artikel disimpan dalam file berformat .txt, lalu data tersebut akan dilakukan tahap preprocessing, setelah itu dilakukan proses klasifikasi menggunakan algoritma *AdaBoost* dan akan

p-ISSN: 2301-5373
e-ISSN: 2654-5101

menghasilkan label prediksi, label prediksi yang dihasilkan dapat lebih dari satu label. langkah-langkah ini akan dijelaskan pada Gambar 1 .



Gambar 1. Alur Penelitian Secara Umum

2.3. Text Preprocessing

Figure 2 menjelaskan detail tentang alur text *processing* dan pembobotan fitur. Pada proses text *processing*, data artikel yang didapatkan dari proses *scrapping* dikenakan proses *transform case* untuk menyamakan jenis huruf pada data, lalu *tokenized* untuk memotong artikel menjadi *term* (kata) , lalu proses *stopword removal* untuk menghapus kata-kata yang dianggap tidak penting dan *lemmatization* untuk mengubah kata ke bentuk kata dasarnya, hasil dari proses – proses ini disebut sebagai data hasil *preprocessing*.

Setelah itu, data hasil *preprocessing* dikonversi menjadi data angka sehingga nantinya bisa diproses oleh algoritma. Data hasil *preprocessing* dikonversi menggunakan 2 metode pembobotan yaitu *Term Frequency* (TF) dan *Term Frekuensi – Invervse Document Frequency* (TF-IDF).

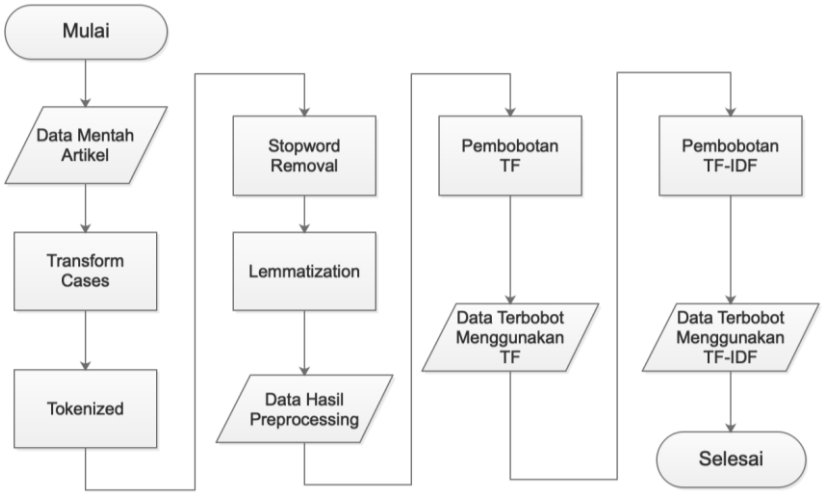


Figure 2. Text Preprocessing & Pembobotan Fitur

2.4. Proses Klasifikasi

Figure 3 menjelaskan detail proses training algoritma *AdaBoost.* Permasalahan multi-label dipecah menjadi beberapa permasalahan klasifikasi binary. Untuk menentukan sebuah data terklasifikasi ke dalam sebuah label, diperlukan beberapa model. Pada awalnya, harus ditentukan berapa jumlah model yang akan digunakan untuk mengklasifikasikan sebuah data termasuk atau tidak ke dalam sebuah label. Pada penelitian ini, model yang digunakan adalah tree. Jumlah model yang akan digunakan dinyatakan dengan simbol nT. Mulanya setiap sampel pada dataset diberikan bobot sampel yang sama. Lalu dilakukan iterasi untuk menghasilkan tree sejumlah nT. Setiap tree dibentuk berdasarkan fitur yang dapat digunakan mengklasifikasikan dataset dengan hasil terbaik. Setelah tree terbentuk, lalu menentukan Amount of Say dari tree, Amount of Say merupakan nilai kekuatan voting model pada klasifikasi final. Setelah menentukan Amount of Say, dilakukan modifikasi bobot pada tiap sampel.

Sampel yang terklasifikasi dengan salah oleh tree yang dibentuk, bobotnya ditingkatnya. Sebaliknya, sampel yang terklasifikasi dengan benar oleh tree yang dibentuk, bobotnya dikurangi. Hal ini bertujuan agar tree yang selanjutnya dibentuk dapat fokus belajar pada keslahan sebelumnya dan memperbaiki kesalahan yang dilakukan tree pendahulunya. Karena terjadi pengurangan dan penambahan bobot pada tiap sampel, maka saat sampel dijumlahkan akan terjadi anomali saat semua bobot sampel dijumlahkan (saat semua bobot dijumlahkan, hasil tidak sama dengan 1), maka dilakukan normalisasi.

Tahap selanjutnya yaitu membuat dataset baru (dengan keadaan kosong) lalu diisikan data, yang datanya diambil secara acak dari dataset lama, pengambilan data secara acak dihentikan saat ukuran dataset baru sama dengan dataset lama. Pada proses pengambilan secara acak ini, dimungkinkan untuk terjadi pengambilan data yang sudah pernah diambil sebelumnya. Pada proses pembuatan dataset baru ini, tiap sampel yang sebelumnya terklasifikasi dengan salah

p-ISSN: 2301-5373
e-ISSN: 2654-5101

akan memiliki peluang terambil lebih dari 1 kali menjadi besar, karena bobotnya yang lebih besar dibandingkan bobot sampel yang sebelumnya terklasifikasi dengan benar.
Setelah dataset baru terbentuk, lalu berikan bobot yang sama pada setiap sampel. Lalu lakukan pembentukan tree kembali hingga jumlah tree yang terbentuk sama dengan nT.

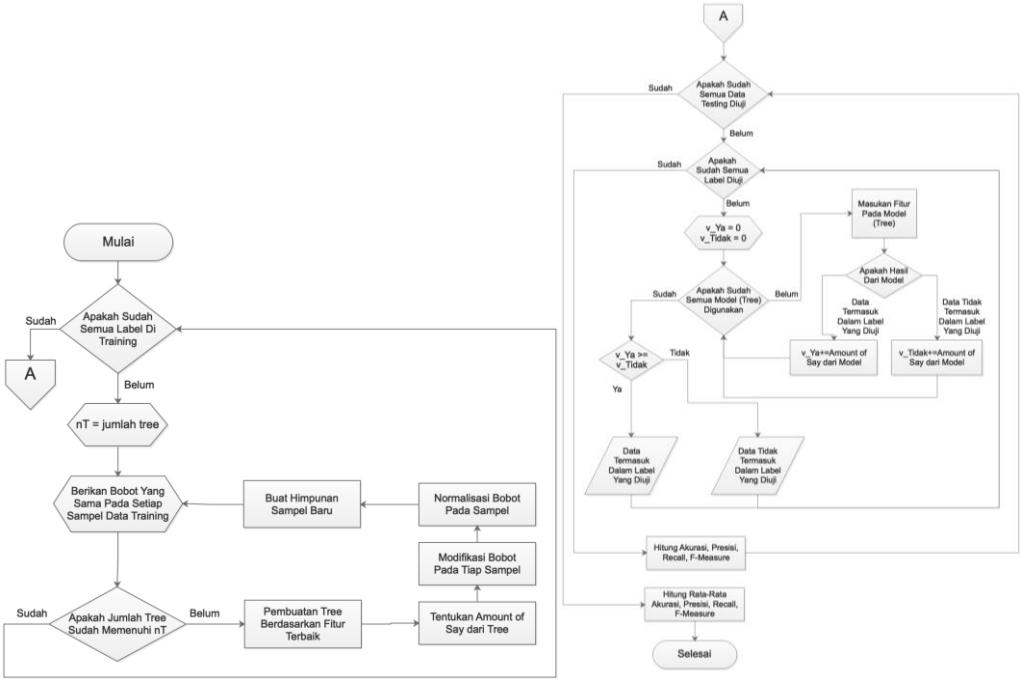


Figure 3. Training, Testing dan Evaluasi

In this eksperimen we used AdaBoost Algorithm. Algoritma *AdaBoost* dikenalkan oleh Yoav Freund dan Rober Schapire pada tahun 1995 [3]. AdaBoost is an abbreviation of Adaptive Boosting. Boosting is a technique for combining several weak classifiers to make predictions with high accuracy.
Boosting is classified as Ensemble Learning. Ensemble Learning in general, is a model that makes predictions based on several different models. By combining several models, ensemble learning tends to be more flexible and less sensitive to data. Boosting trains several individual models sequentially. Each model learns from the mistakes made by the previous model.
This algorithm is a method with the concept of supervised learning used for classification. Although the AdaBoost algorithm can now be used in many statistical models, the AdaBoost algorithm was initially applied only to the regression model. The AdaBoost algorithm is used to improve the accuracy of predictions made by using the exponential loss function. The essence of the AdaBoost algorithm is to give more weight to the weak classifier which intends to force the algorithm to concentrate on the weak classifier. Weak classifier is a classification that produces predictions that are slightly better than random guessing.

Table 1. AdaBoost Pseudocode

Pseudocode
Input : Dataset D = {(x1,y1),(x2,y2),…,(x_m,y_m)} Base Learning Algorithm DecisionTree Number of learning rounds T
Process :
D1(i) = 1/m //initialize the weight distribution
For t = 1,…,T:
H_t = DecisionTree(D,D_t) //Train a weak learner h_t from D using distribution D_t
E_t = $Pr_{i\sim Di}$ //Measure the error of h_t

$a_t = ½ \left(\frac{1- E_t}{E_t}\right)$ //Determine the weight of h_t

$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-a_t) \; if \; h_t(x_i) = y_i \\ \exp(a_t) \quad if \; h_t(x_i) \neq y_i \end{cases}$

$= \frac{D_t(i) \exp(a_t y_i h_t(x_i))}{Z_t}$ //Update the distributuion, where Z_t is a normalization factor which enables D_{t+1} be a distribution

Output : $H(x) = \text{sign}(\sum_{t=1}^{T} a_t h_t(x))$

2.5. Pengujian dan Evaluasi

Metode pengujian yang digunakan adalah *Cross Validation. Cross-validation (CV)* adalah metode statistik yang dapat digunakan untuk mengevaluasi kinerja model atau algoritma dimana data dipisahkan menjadi dua subset yaitu data proses pembelajaran dan data validasi / evaluasi. Model atau algoritma dilatih oleh subset pembelajaran dan divalidasi oleh subset validasi. Selanjutnya pemilihan jenis *Cross Validatio*n dapat didasarkan pada ukuran dataset. Biasanya *Cross Validation K-fold* digunakan karena dapat mengurangi waktu komputasi dengan tetap menjaga keakuratan estimasi.

10 *fold Cross Validation* adalah salah satu *K fold Cross Validation* yang direkomendasikan untuk pemilihan model terbaik karena cenderung memberikan estimasi akurasi yang kurang bisa dibandingkan dengan *Cross Validation* biasa, *leave-one-out Cross Validation* dan *bootstrap*. Dalam 10 *fold Cross Validation*, data dibagi menjadi 10 *fold* berukuran kira-kira sama, sehingga kita memiliki 10 subset data untuk mengevaluasi kinerja model atau algoritma. Untuk masing-masing dari 10 subset data tersebut, *Cross Validation* akan menggunakan 9 *fold* untuk pelatihan dan 1 *fold* untuk pengujian. Mesin akan mengetahui label dari data pada *fold* pelatihan untuk kebutuhan training, sedangkan mesin tidak akan mengetahui label dari data pada *fold* pengujian.

Pada penelitian ini akan dilakukan pengujian terhadap hasil klasifikasi dari data testing. Pengujian dimaksudkan untuk mengukur keakuratan hasil dari model yang digunakan. Pengukuran yang akan dilakukan adalah pengukuran *Accuracy, Precision, Recall dan F-measure. Accuracy* adalah tingkat kedekatan antara nilai aktual dengan nilai prediksi. *Precision* adalah proporsi kasus yang diprediksi positif yang juga positif benar [4] pada data sebenarnya. *Recall* adalah proporsi dari kasus positif kejadian sebenarnya yang diprediksi positif benar. *F-measure* adalah rata-rata harmonik dari nilai *precision* dan *recall [5].* Dalam pengukuran terhadap kasus multi-label, *confused metrics* tradisional tidak tepat untuk digunakan. Godbole & Sarawagi [6] mengembangkan definisi dari *Accuracy, Precision, Recall dan F-measure* seperti persamaan dibawah ini, agar dapat digunakan terhadap kasus klasifikasi multi-label. Untuk setiap objek i,

- Ci melambangkan label yang benar
- Pi melambangkan label yang di prediksi
- n melambangkan jumlah observasi
- | | melambangkan jumlah data pada himpunan label

$$Accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{|\, Ci \cap Pi \,|}{|\, Ci \cup Pi \,|} \qquad (1)$$

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{|\, Ci \cap Pi \,|}{|\, Pi \,|} \qquad (2)$$

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{|\, Ci \cap Pi \,|}{|\, Ci \,|} \qquad (3)$$

p-ISSN: 2301-5373
e-ISSN: 2654-5101

$$F - measure = \frac{1}{n}\sum_{i=1}^{n}\frac{2\mid Ci \cap Pi \mid}{\mid Ci \mid + \mid Pi \mid} \qquad (4)$$

Untuk menguji performa dilakukan pengujian dengan mengukur rata-rata waktu komputasi algoritma. Waktu komputasi algoritma *AdaBoost* akan dibandingkan dengan waktu komputasi algoritma lain yang termasuk *algorithm adaptation* [2] dengan dataset yang sama. Untuk setiap objek i, S_i melambangkan waktu komputasi.

$$\text{Performa} = \frac{1}{n}\sum_{i=1}^{n} S_i \qquad (15)$$

Algoritma yang digunakan sebagai pembanding adalah algoritma C4.5 dan *Backpropagation algorithm for Multilabel Learning (BP-MLL).*

3. Hasil dan Pembahasan

Untuk membuktikan algoritma AdaBoost bekerja dengan optimal, penulis melakukan eksmerimen dengan dataset 5 label dan 1.287 data. Label yang digunakan adalah *Algorithm, Big Data, E-Commerce, Machine Learning* dan *Programig.*

Table 2. Tabel Distribusi Pada Dataset

Label	Kemunculan
Programing	312
Machine Learning	412
Big data	319
Algorithm	303
E-Commerce	317

3.1. Performa Terbaik dari Metode Pembobotan

Eksperimen menggunakan algoritma AdaBoost dengan tujuan membandingkan dataset dengan pembobotan TF atau TF-IDF yang menghasilkan hasil yang lebih optimal pada penerapan algoritma AdaBoost. Proses eksperimental menggunakan dua variabel beragam, yaitu variabel panjang fitur yang digunakan dalam dataset dan jumlah variabel estimator yang digunakan dalam algoritma AdaBoost. Panjang fitur dataset yang digunakan dalam percobaan ini mulai dari 100, 200, 300, 400, 500, 600, 700, 800, 900 dan 1000. Jumlah penduga yang digunakan dalam percobaan ini mulai dari 10, 20, 30, 40, 50, 60, 70, 80, 90 dan 100. Hasil percobaan ditunjukkan pada Gambar 4.

Dari percobaan yang dilakukan disimpulkan bahwa penggunaan dataset dengan bobot TF-IDF selalu menghasilkan akurasi, presisi, daya ingat dan ukuran-f yang lebih tinggi dibandingkan dengan dataset dengan bobot TF.

3.2. Eksperimen Menggunakan Variasi Panjang Fitur

Eksperimen yang menggunakan algoritma AdaBoost dengan tujuan membandingkan hasil klasifikasi menggunakan berbagai panjang fitur dan mengetahui panjang fitur yang memberikan hasil optimal menggunakan klasifikasi algoritma AdaBoost. Proses eksperimental menggunakan dua variabel beragam, yaitu variabel panjang fitur yang digunakan dalam dataset dan jumlah variabel estimator yang digunakan dalam algoritma AdaBoost. Panjang fitur dataset yang digunakan dalam percobaan ini mulai dari 100, 200, 300, 400, 500, 600, 700, 800, 900 dan 1000. Jumlah penduga yang digunakan dalam percobaan ini mulai dari 10, 20, 30, 40, 50, 60, 70, 80, 90 dan 100. Hasil percobaan ditunjukkan pada Gambar 5 dan 6.

Dari percobaan yang dilakukan, diketahui bahwa peningkatan penggunaan jumlah estimator dalam algoritma AdaBoost menyebabkan peningkatan hasil akurasi, presisi, recall dan f-mengukur dalam setiap dataset dengan berbagai panjang fitur. Hasil optimal dari percobaan

yang dilakukan adalah akurasi 0,987, presisi 0,99, recall 0,99 dan f-ukuran 0,989 dengan menggunakan 100 estimator dan dataset dengan 900 fitur panjang.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

3.3. Membandingkan Hasil AdaBoost dengan C4.5 dan BPMLL

Perbandingan dibuat menggunakan hasil optimal dari masing-masing algoritma. Hasil dari algoritma AdaBoost digunakan saat menggunakan 100 estimator dan hasil BPMLL digunakan saat menggunakan 5 hidden layar, 50 epochs dan learning rate 0,1. Hasil percobaan ditunjukkan pada Gambar 6.

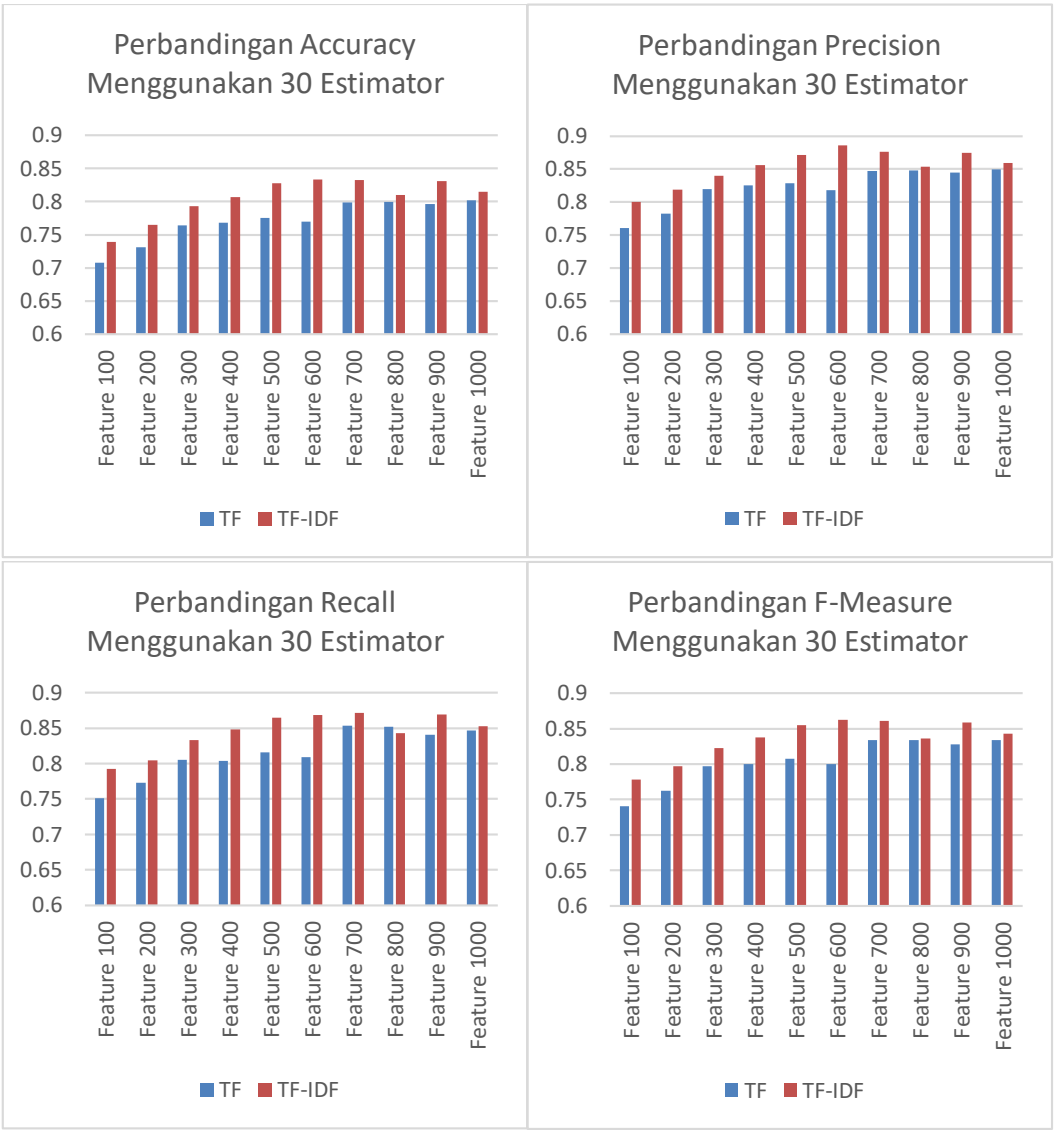


Figure 4. Grafik Accuracy, Precision, Recall and F-Measure untuk membandingkan hasil TF and IDF dengan 30 Estimators

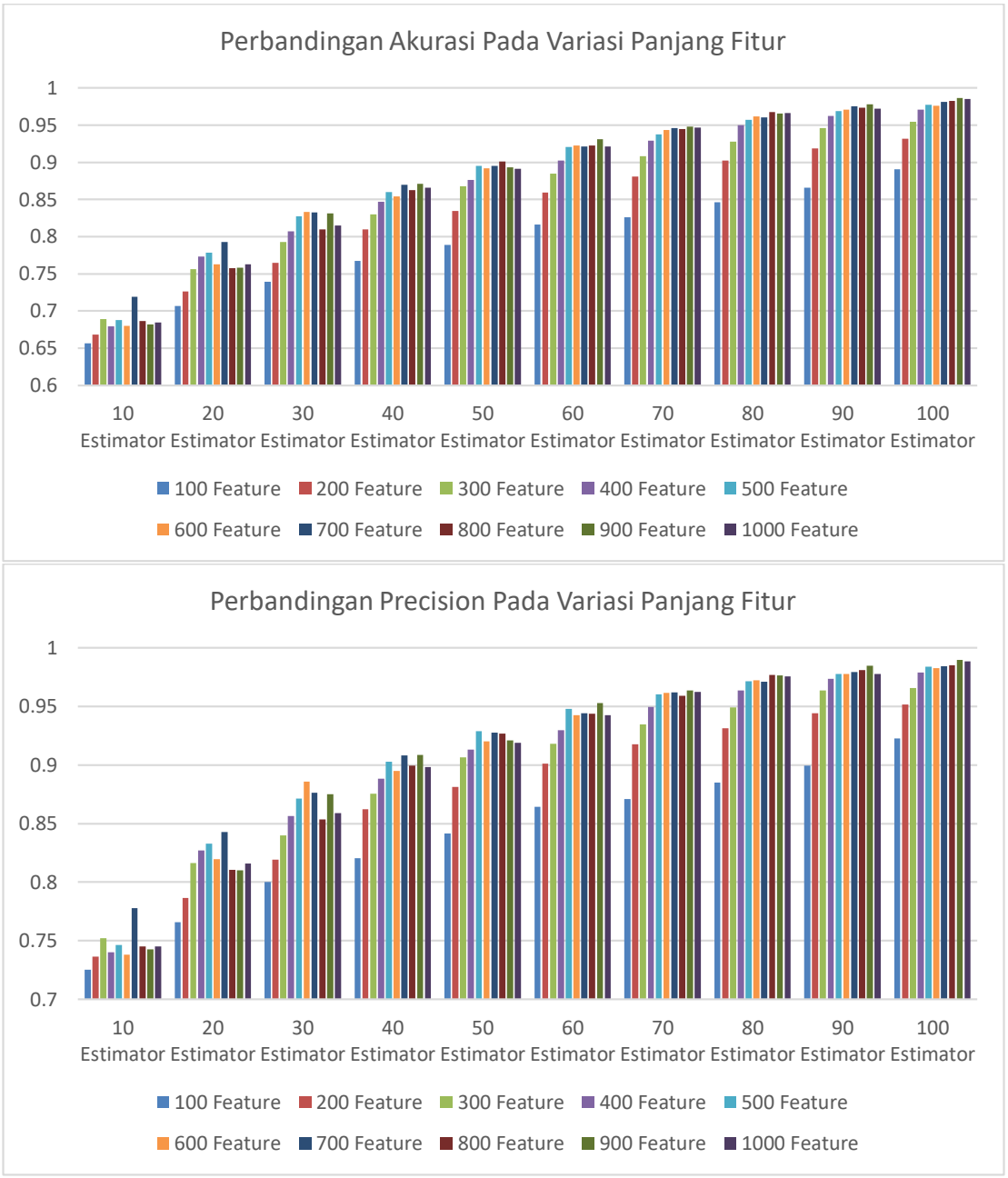


Figure 5. Grafik Accuracy dan Precision untuk membandingkan hasil dengan variasi panjang fitur

p-ISSN: 2301-5373
e-ISSN: 2654-5101

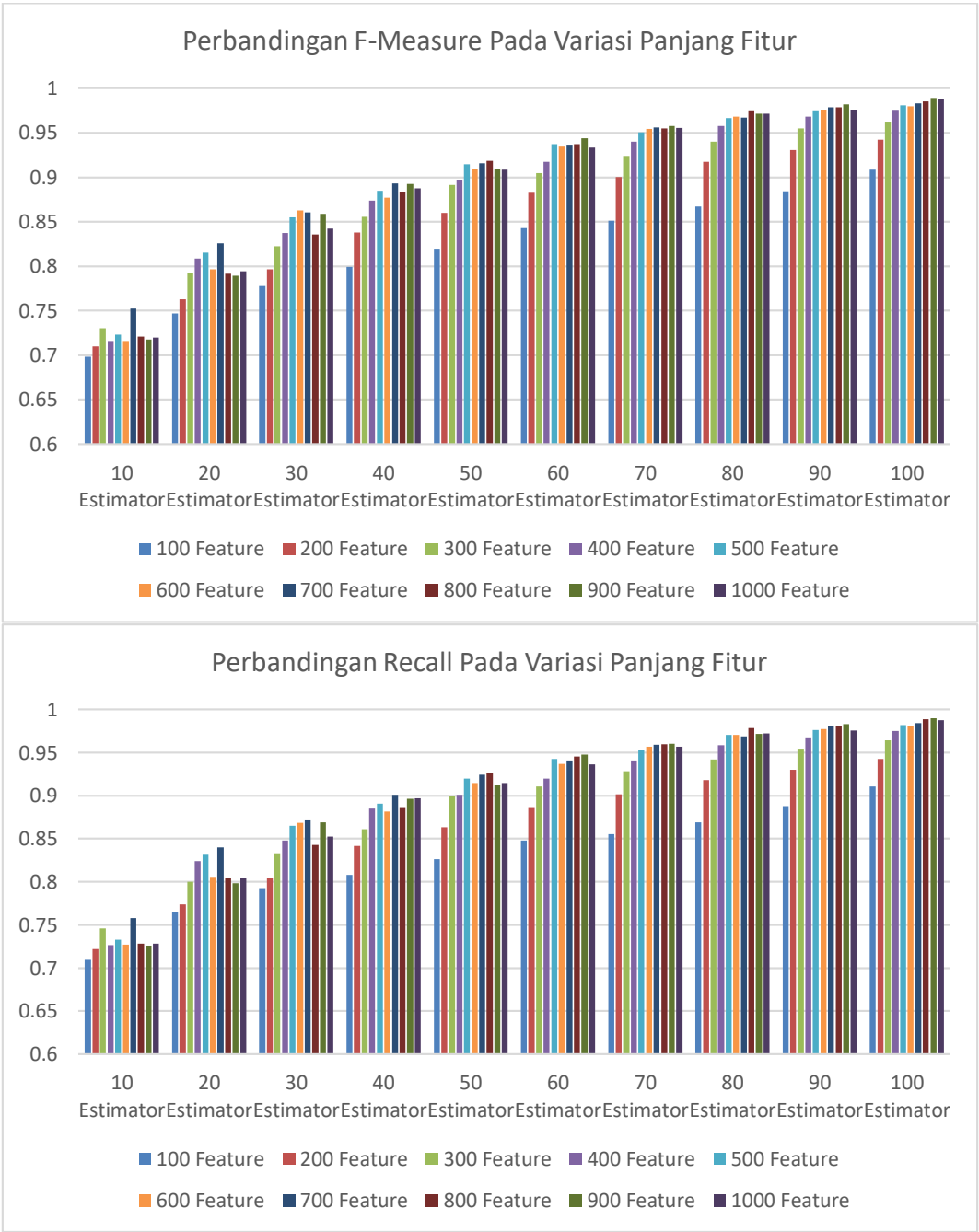


Figure 6. Grafik Recall dan F-Measure untuk membandingkan hasil dengan variasi panjang fitur

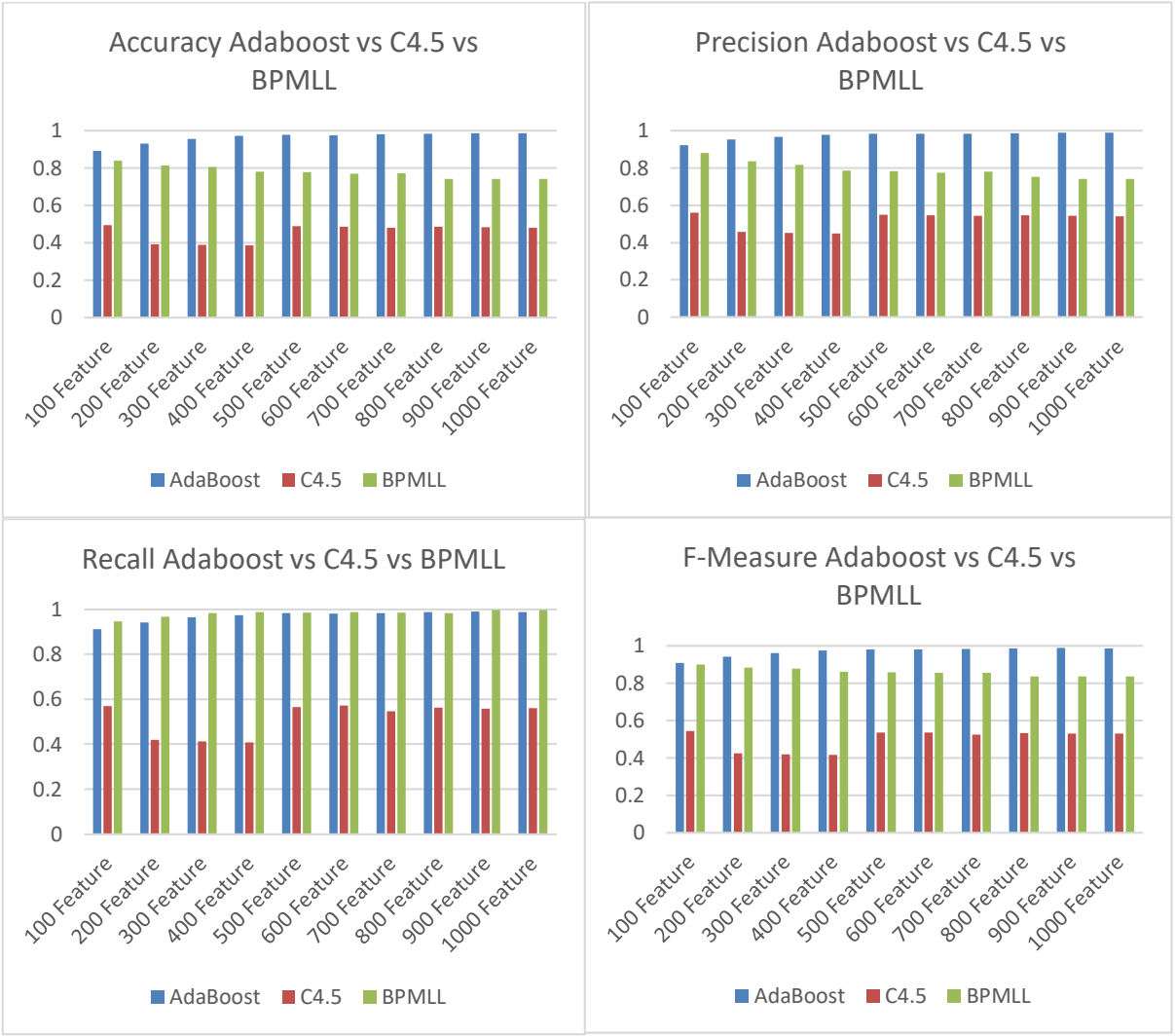


Figure 7. Grafik Accuracy, Precision, Recall, F-Measure Membandingkan AdaBoost, C4.5 dan BPMLL

Dari percobaan yang dilakukan, algoritma AdaBoost memberikan hasil akurasi, presisi dan f-mengukur yang lebih tinggi dibandingkan dengan hasil dari algoritma C4.5 dan BPMLL, sedangkan untuk hasil recall algoritma AdaBoost dan BPMLL memiliki perbedaan yang sedikit dan berada di atas 0,9 , ini berarti algoritma AdaBoost dan BPMLL memiliki tingkat keberhasilan yang baik dalam mengklasifikasikan. Hasil recall dari algoritma AdaBoost dan BPMLL lebih tinggi dari pada algoritma C4.5. Dari percobaan yang dilakukan, dapat disimpulkan bahwa algoritma AdaBoost memberikan hasil yang lebih optimal dibandingkan dengan algoritma C4.5 dan BPMLL.

3.4. Membandingkan Efisiensi AdaBoost dengan C4.5 dan BPMLL

Untuk mengetahui efisiensi dari algoritma AdaBoost dilakukan perbandingan waktu komputasi dari algoritma AdaBoost dengan algoritma waktu komputasi C4.5 dan BPMLL. Waktu komputasi dicatat dalam satuan detik fraksional (detik diwakili dalam format float). Perbandingan waktu komputasi antara algoritma AdaBoost dan C4.5 dilakukan. Hasil eksperimen yang digunakan adalah algoritma AdaBoost menggunakan 10 estimator dengan tingkat akurasi 0,65 hingga 0,68 (tingkat akurasi terendah yang diperoleh selama percobaan). Algoritma C4.5 yang digunakan memiliki tingkat akurasi 0,495 - 0,479. Dataset yang digunakan dalam percobaan ini adalah dataset dengan 5 label dan bobot TF-IDF. Hasil percobaan ditunjukkan pada Gambar 8.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

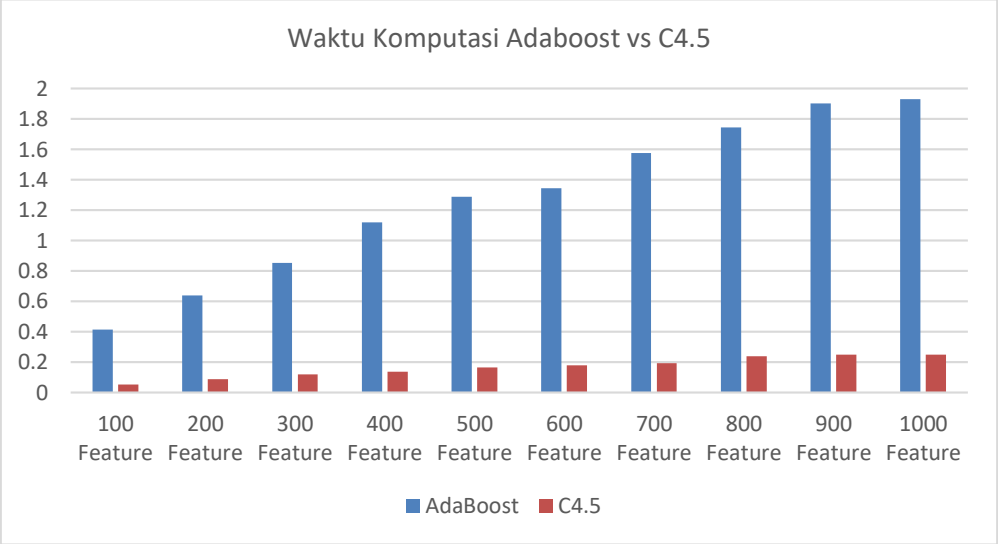


Figure 8. Grafik untuk membandingkan Waktu Komputasi AdaBoost & C4.5

Dari percobaan yang dilakukan, diketahui bahwa algoritma C4.5 memiliki waktu komputasi yang jauh lebih cepat dibandingkan dengan algoritma AdaBoost. Algoritma AdaBoost membutuhkan waktu dari 0,413 hingga 1,931 untuk mencapai akurasi 0,65 - 0,68 sedangkan C4,5 hanya membutuhkan 0,053 - 0,249 untuk mendapatkan akurasi 0,495 - 0,479. Meskipun lebih cepat, algoritma C4.5 tidak dapat menandingi hasil yang diberikan oleh algoritma AdaBoost.

Dilakukan perbandingan waktu perhitungan antara algoritma AdaBoost dan BPMLL. Hasil eksperimen yang digunakan adalah algoritma AdaBoost menggunakan 30 estimator dengan tingkat akurasi 0,73 - 0,83. Algoritma BPMLL yang digunakan memiliki tingkat akurasi 0,74 hingga 0,83 dengan 5 hidden layer, 50 epochs dan learning rate 0,1. Dataset yang digunakan dalam percobaan ini adalah dataset dengan 5 label dan bobot TF-IDF. Hasil percobaan ditunjukkan pada Gambar 9.

Dari percobaan yang dilakukan, diketahui bahwa algoritma AdaBoost memiliki waktu komputasi yang jauh lebih cepat dibandingkan dengan algoritma BPMLL. Algoritma AdaBoost membutuhkan waktu dari 1.501 hingga 7.188 untuk mencapai akurasi 0.73 - 0.83 sementara BPMLL membutuhkan 52.12 - 86.54 untuk mendapatkan akurasi 0.74 - 0.83. Dengan demikian dapat disimpulkan bahwa algoritma AdaBoost membutuhkan waktu komputasi yang lebih sedikit daripada BPMLL untuk mencapai akurasi yang sama.

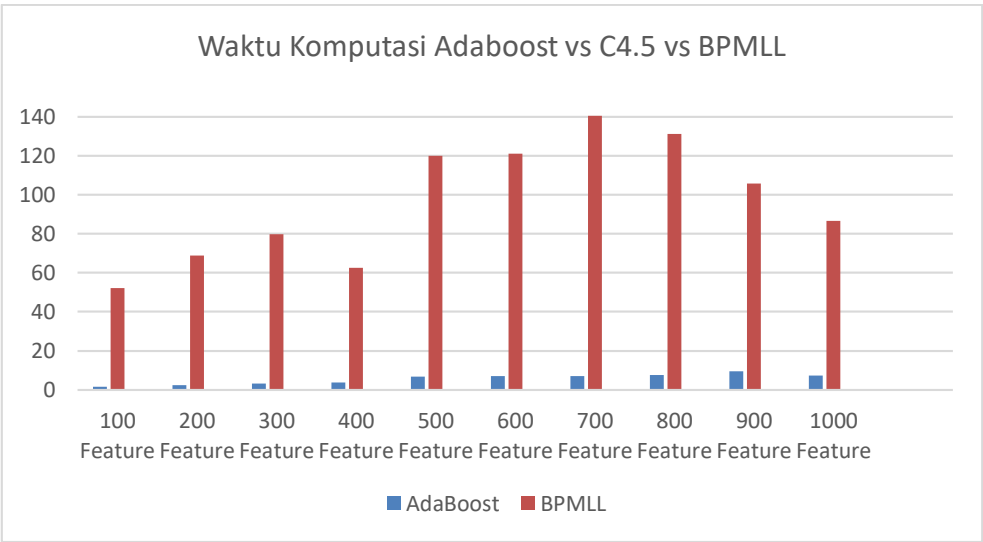


Figure 9. Grafik untuk membandingkan Waktu Komputasi AdaBoost & BPMLL

4. Conclusion

Dari penelitian yang telah dilakukan didapatkan kesimpulan sebagai berikut:

a. Penggunaan Dataset TF-IDF menghasilkan hasil *accuracy, precision, recall* dan *f-measure* yang lebih tinggi dibandingkan dataste TF pada klasifikasi Algoritma *AdaBoost.*

b. Peningkatan jumlah estimator pada algoritma *AdaBoost* menyebabkan peningkatan hasil *accuracy, precision, recall* dan *f-measure* pada klasifikasi.

c. Dari eksperimen yang dilakukan, penggunaan 100 estimator pada algoritma *AdaBoost* dan menggunakan dataset TF-IDF dengan panjang fitur 900 memberikan hasil yang teroptimal.

d. Algoritma *AdaBoost* memberikan hasil yang lebih optimal dibandingkan algoritma C4.5 dan BPMLL.

e. Algoritma *AdaBoost* memerlukan waktu komputasi yang lebih sedikit dibandingkan BPMLL untuk mencapai tingkat *accuracy* yang setara. Sedangkan Algoritma *AdaBoost* memerlukan waktu komputasi yang lebih banyak dibandingkan algoritma C4.5, namun hasil *accuracy, precision, recall* dan *f-measure* pada algoritma C4.5 tidak mampu menyaingi algoritma *AdaBoost.*

Referensi

[1] Wikipedia. *Wikipedia:Statistik*. Retrieved from Wikipedia Ensiklopedia Bebas: https://id.wikipedia.org/wiki/Wikipedia:Statistik . 2019 (1)

[2] Dharmadhikari, S. C., Ingle, M., & Kulkarni, P. A Comparative Analysis of Supervised Multi-label Text Classification Methods. *International Journal of Engineering Research and Applications (IJERA)*. 2011. (2)

[3] Schapire, R. E., & Singer, Y. BoosTexter: A boosting-based system for text categorization. *Machine learning*, 135-168. 2000. (3)

[4] Powers, D. M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011. (4)

[5] Han, J., & Kamber, M. Data Mining: Concepts and Techniques. 2012. (5)

[6] Godbole, S., & Sarawagi, S. Discriminative methods for multi-labeled classification. *Pacific-Asia conference on knowledge discovery and data mining* (pp. 22-30). Springer. 2004. (6)