KLASIFIKASI DATA TIME SERIES ARUS LALU LINTAS JANGKA PENDEK MENGGUNAKAN ALGORITMA ADABOOST DENGAN RANDOM FOREST

Ahmad Rofiqul Muslikh[(1)], Heru Agus Santoso[(2)], Aris Marjuni[(3)]

[1] Dosen Universitas Merdeka Malang
[2,3] Dosen Universitas Dian Nuswantoro Semarang

[1] rofickachmad@unmer.ac.id, [2] heru.agus.santoso@dsn.dinus.ac.id,
[3]aris.marjuni@dsn.dinus.ac.id

Tersedia Online di
http://www.jurnal.unublitar.ac.id/index.php/briliant

Sejarah Artikel
Diterima pada 30 Januari 2019
Disetuji pada 18 Februari 2019
Dipublikasikan pada 20 Februari 2019 Hal. 78-88

Kata Kunci:
Traffic flow , Prediction , Adaboost, Random Forest

DOI:
http://dx.doi.org/10.28926/briliant.v3i3.272

Abstrak: Data arus lalu lintas di Indonesia di gunakan untuk manajemen kontrol arus lalu lintas, padahal data tersebut di dapatkan dari hasil survei yang di lakukan langsung kelokasi, survei yang di lakukan masih kurang efektif, dan data yang di dapat dari hasil survei tersebut di gunakan sebagai acuan dalam kontrol arus lalu lintas, maka dari itu untuk mendapatkan data arus lalu lintas yang lebih efektif di perlukan sebuah pendekatan baru yang bisa mengklasifikasikan data yang di dapat dengan akurasi yang lebih tinggi, agar kepadatan dan kemacetan dapat di prediksi lebih dini.Pada penelitian ini digunakan pendekatan menggunakan algoritma *Adaboost* dan Random forest untuk mengklasifikasikan dan memprediksi data hasil survei yang bersifat time series, hasil pengujian untuk prediksi menggunakan *Adaboost* dengan *Random Forest* Dengan *Confusion Matrix* sebagai pengukur tingkat akurasinya sebesar 99,08%, dan tingkat error di dapatkan sebesar 0,0629. Pada hasilnya menggunakan pendekatan *Adaboost* dengan *Random Forest* terbukti lebih efisien dalam mengklasifikasikan dan memprediksi data hasil survei daripada hanya mengandalkan data asli dalam memprediksi arus lalu lintas

PENDAHULUAN

Transportasi merupakan sektor yang penting dalam kehidupan masyarakat, karena dengan adanya transportasi membantu masyarakat untuk melakukan kegiatan ekonomi. Kegiatan ekonomi tidak akan berjalan jika tidak di seimbangkan dengan kebutuhan transportasi yang memadai. Pada penelitian [1] bahwa trend perkembangan transportasi pada saat ini masih belum memadai, atau dalam istilah lain masih belum seimbang antar pertumbuhan penduduk dengan pembangunan infrastruktur transportasi.

Fungsi Jalan pada kota malang dibagi menjadi 3 jalan, Arteri Primer, Arteri Sekunder, Lokal Primer, dan Lokal Sekunder [4]. Kota Malang memiliki panjang total jalan sebesar 663,34 KM, yang terbagi menjadi jalan Arteri Primer sepanjang 11,82 km, Arteri Sekunder sepanjang 15,94 km, Kolektor Primer sebesar 8,16 km, Kolektor sekunder sebesar 27,09 km, Lokal Primer sepanjang 9,66 km, dan Lokal Sekunder sepanjang 590,67 km [4]. Titik kemacetan paling parah pada pintu masuk kota malang berada di pertigaan Karanglo yang termasuk jalur Arteri Primer. Sedangkan pertigaan karanglo yang menjadi objek penelitian ini berada pada jalan malang kabupaten. Yang mana pertumbuhan jalan kabupaten

malang yang diantaranya jalan Negara sepanjang 115,63 km (1%), jalan provinsi 114,93 km (1%), jalan kabupaten 1.668,76 km (19%), dan jalan desa 6.907,90 km (79%), dan rata – rata pertumbuhan jalan di kabupaten Malang hanya tumbuh 1% tiap tahun, berbanding dengan pertumbuhan transportasinya yang tumbuh 3,6% pertahun [bappekab.] Di pertigaan karanglo ini arus semua kendaraan jadi satu, bayangkan saja, arus kendaraan yang ingin ke luar dari kota malang dan dari luar kota malang ke kota malang dan dari luar kota malang ingin ke kota batu, arusnya bertemu jadi 1 di pertigaan karang lo ini. Betapa macetnya jalan tersebut jika semua ingin berbagi jalan dalam 1 jalan yang sudah melebihi kapasitasnya. Pemerintah Malang sudah merencanakan solusi jangka panjang untuk kemacetan yang terjadi di pintu kota malang ini, yaitu membangun jalan alternatif lawang – batu, dengan membangun jalur alternative lawang – batu kedepannya arus jalan sudah tidak lagi menjadi satu bertemu di pertigaan karanglo, tetapi wisatawan yang dari luar kota malang ingin ke kota batu langsung bisa menuju kota batu tanpa harus melewati pertigaan karanglo. Arus kendaraan yang berkurang menjadi dampak positif dari kemacetan yang terjadi.

Sudah banyak penelitian sebelumnya yang membahas solusi arus lalu lintas dalam jangka pendek di antaranya adalah penelitian yang di lakukan oleh *Tao Li dan Liu Seng* [5] mereka meneliti arus lalu lintas dalam jangka pendek menggunakan pendekatan algoritma PSO (Particle Swarm Optimization) based cloud theory dan di optimasikan menggunakan Neural Network Backgpropagation, pada penelitian tersebut di peneliti menggunakan 2 metode yang mana 2 metode yang digunakan tersebut bergantung pada karakteristik data yang di kirim oleh sensor pada lokasi dan konfigurasi system sensor yang di gunakan. Karena pada metode tersebut menggunakan dua asumsi waktu yang bervariasi dengan mengintegrasikan mekanisme cloud pada batas tertentu menjadikan model wavelet neural network menjadi lemah dari segi waktu tetapi hasil prediksi menjadi lebih menigkat.

METODE

Data Mining

Data mining merupakan salah satu disiplin ilmu yang mempelajari metode dan mengekstrak pengetahuan atau menemukan pola dari suatu data. Data sendiri merupakan suatu fakta yang terekam dan tidak mempunyai arti. Dan pengetahuan merupakan suatu pola, aturan atau model yang muncul dari data, sehingga data mining disebut dengan Knowledge Discovery in Database (KDD)[14]Transformasi data dalam Data Mining bisa di gambarkan sebagai berikut : Data -Informasi - Pengetahuan.

Dalam proses KDD bisa dibagi menjadi berikut[15]:

a. Seleksi
b. Pre – processing
c. Transformasi
d. Data Mining
e. Interpretasi / evaluasi

Adaboost

Adaboost atau secara kata lain disebut dengan *Adaptive Boosting* merupakan meta algoritma dalam machine learning, di usulkan oleh Yoav Freund

dan Robert Schapire[21]. Algortima adaboost ini bisa digunakan dengan jenis algoritma lain untuk menigkatkan kinerja dari algoritma yang digabungkan tersebut. Output dari algoritma yang lain yang digabungkan dengan adaboost adalah "weak Learners" dikombinasikan dengan jumlah yang ditimbang berdasarkan bobot untuk mewakili hasil akhir dari classifier yang boosted. Algoritma adaboost merupakan algoritma yang adaptive artinya setiap "weak Learners" yang akan ditingkatkan berdasarkan kesalahan klasifikasi dari pengklasifikasi sebelumnya. Dalam beberapa kasus adaboost merupakan algoritma yang sensitif terhadap noise data, namun bisa berkurang kerentanan terhadap noise data bila digabungkan dengan algoritma machine learning lainnya. Individual Learners (pemebelajar individu) dapat menjadi weak "lemah" tetapi selama kinerja dari masing – masing sedikit lebih baik daripada menebak secara acak,(tingkat kesalahan lebih kecil dari 0,5 untuk klasifikasi secara biner). Pemodelan akhir dapat dibuktikan dengan mengumpulkan mana sebagai learner atau pembelajar.

a. Training

Algoritma AdaBoost mengacu pada metode tertentu untuk melatih classifier, sebuah boosting classifier bisa dirumuskan sebagai berikut :

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \quad \text{(1.1)}$$

Yang mana setiap f_t sebagai "weak learner" yang mengambil objek dari x sebagai inputan dan mengmbalikan hasil berupa class dari objek. Tanda dari hasil "weak learner" yaitu mengidentifikasi objek yang diprediksi dan nilai absolut memeberikan confidence dalam pengklasifikasian. Demikian pula *T*-layer sebagai classifier akan menjadi positif jika sampel diyakini berasal dari class yang memang positif dan negative jika sebaliknya. Setiap "weak learner" menghasilkan sebuah output berupa hipotesis $h(x_i)$ untuk setiap sampel pada data training set, dan pada setiap iterasi *t*. "weak learner" yang lemah dipilih dan ditetapkan secara koefisien α_t sehingga kesalahan pada saat training penjumlahan E_t dari yang dihasilkan dengan *t*- dapat meningkatkan dalam tahap classifier dapat diminimalkan.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)] \quad \text{(1.2)}$$

Yang mana $F_{t-1}(x)$ adalah sebagai classifier yang di boosting dan di bangun sampai sebelum tahap training. *E(F)* merupakan beberapa fungsi dari kesalahan $f_t(x) = \alpha_t h(x)$ merupakan "weak learner" yang akan dipertimbangkan selain classifier yang terakhir.

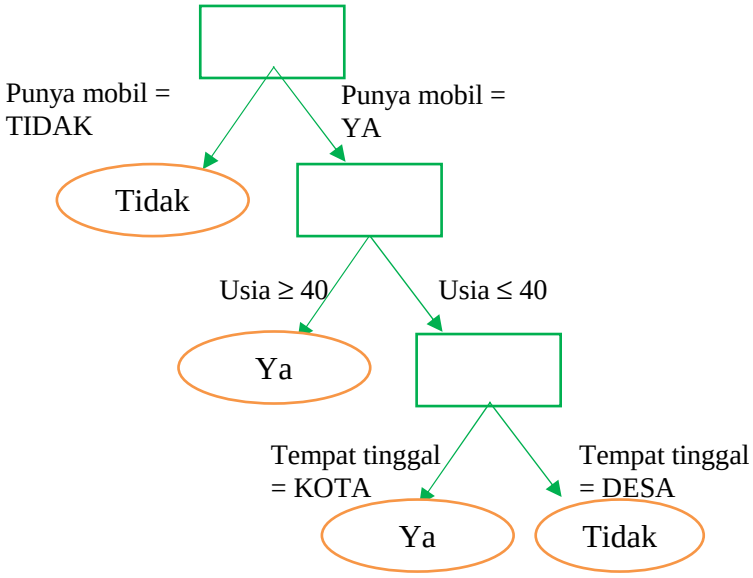
a. Weighting

Pada setiap iterasi dari proses training, pembobotan ditugaskan pada masing – masing sampel ditetapkan sama dengan kesalahan saat $E(F_{t-1}(x_i)$ bobot ini dapat digunakan untuk menginformasikan training dari "weak learner" decision tree dapat tumbuh dan mendukung pemisahan set dan sampel dengan bobot yang tinggi.

RANDOM FOREST

Random forest merupakan salah satu metode pohon gabungan berdasarkan klasifikasi dan regresi. Cara kerja dari random forest yaitu membangun banyak pohon keputusan pada saat training data dan output class yang merupakan modus dari class (Klasifikasi) atau prediksi (regresi) dari sebuah pohon tunggal.

Dari sebuah gugus data yang dimiliki kita bisa mengambil sebagian data untuk digunakan sebagai pengamatan secara acak (resampling) yang selanjutnya digunakan untuk menyusun pohon. Pengulangan proses ini akan menghasilkan sebagian data yang berbeda dan di ikuti dengan pohon yang berbeda pula. Nama *Bagging* merupakan singkatan dari *Bootstrap aggregating*. Berikut contoh penggabungan dari dugaan pohon[20].



Gambar 2.2.5.1 : Ilustrasi Pohon 1[20]

TEKNIK EVALUASI

Confusion Matrix merupakan table klasifikasi yang mengandung informasi hasil perhitungan system secara keseluruhan[22]. Pengukuran data di evaluasi melalui akurasi, presisi, recall. Hasil pengukuran direpresentasikan ke dalam sebuah table klasifikasi untuk memudahkan pembacaan. Akurasi adalah jumlah presentase dari prediksi system yang tepat. Presisi merupakan ukuran dari akurasi suatu *class* yang telah di prediksi oleh system. Sedangkan *recall* adalah presentase data dengan nilai positif dari hasil prediksi yang nilainya juga positif. Adapun perhitungannya yaitu :

Akurasi = $(TP+TN) / (TP+FP+TN+FN)$
Presisi = $TN / (FP+TN)$
Recall = $TP / (TP+FN)$

(4)

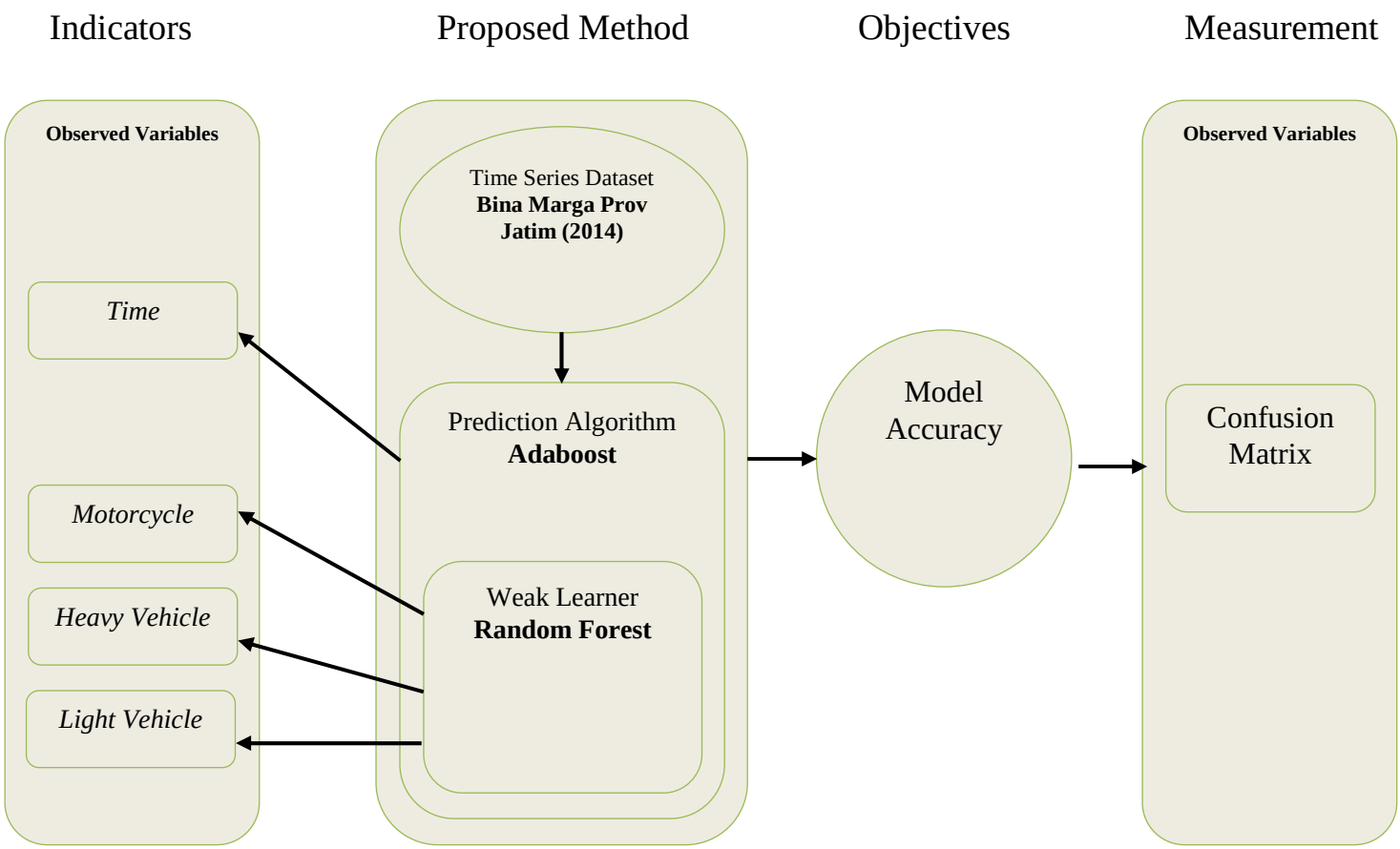
Keterangan:

TP=True Positif *TN*=True Negative
FP=False Positif *FN*=False Negative

		Predicted	
		Negative	Positive
Actual	Negative	TN	TP
	Positive	FN	FP

Kerangka Pemikiran

Ilustrasi Kerangka Pemikiran



Hasil Perhitungan Trafik

Dalam perhitungan trafik, dalam dataset terdapat beberapa kategori yaitu LV, HV, dan MC. Maka kategori tersebut dijadikan sebuah atribut dalam perhitungan klasifikasi dalam *Adaboost* dan *Random Forest* nanti, pengkategorian atribut bisa di lihat di bawah ini :

Tabel 4.1.1.2.1 : Tabel hasil kategori / atribut

Hari	Pukul	Jumlah Kendaraan (Kend.)		
		Kend. Ringan (LV)	Kend. Berat (HV)	Sepeda Motor (MC)
6	06.00-06.15	166	63	546
6	06.15-06.30	146	54	578
6	06.30-06.45	190	68	762
6	06.45-07.00	210	80	702
6	07.00-07.15	194	90	789
6	07.15-07.30	202	83	700
6	07.30-07.45	198	65	711
6	07.45-08.00	186	69	671

Dalam kategori atribut di atas representasi hari

Tabel 4.1.1.1 : Tabel hasil normalisasi

Hari	Pukul	(LV)	(HV)	(MC)	Y
6	06.00.00	166	63	546	1
6	06.15.00	146	54	578	1
6	06.30.00	190	68	762	1
6	06.45.00	210	80	702	1

Tabel di atas bertipe numerik semua dan yang menjadi atribut x adalah Hari, pukul, LV, HV, dan MC. Pada atribut hari di lambangkan dengan 1 – 6 yang mana artinya hari 1 adalah senin, 2 adalah selasa, 3 adalah rabu, 4 adalah kamis, 5 adalah jum'at, 6 adalah sabtu, dan 7 adalah minggu. Adapun class atau label yang di prediksi adalah Y, dalam class / label Y bias di representasikan menjadi 1 dan -1, yang mana 1 di representasikan terjadinya kemacetan dan -1 sebaliknya atau di representasikan tidak terjadi kemacetan. Atribut LV merupakan kepanjangan dari *Light Vehicle* yang berartt kendaraan ringan atau bisa di representasikan sebagai mobil. Atribut HV kepanjangan dari Height Vehicle atau kendaraan berat dan bias di representasikan sebagai Truck, Bus. Dan MC adalah *MotorCycle* yang artinya kendaraan roda dua bisa di representasikan dengan Motor.

PEMBAHASAN

Menentukan Train dan Test Data

1. Import dataset dari table matrix yang sudah di buat pada saat konversi data.
2. Menentukan sample training dataset $S=((x_i,y_i),i=(1,2,\ldots,m))$ dengan label *Y*

S = sample

x_i = atribut domain

y_i = *class*

i = *iteration*

m = *label training*

Eksperimen 4.1 : sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)$

x1	x2	x3	x4	x5	*yi*
0.0060	*0.0003*	*0.1660*	*0.0630*	*0.5460*	*0.0010*
0.0060	*0.0003*	*0.1460*	*0.0540*	*0.5780*	*0.0010*
0.0060	*0.0003*	*0.1900*	*0.0680*	*0.7620*	*0.0010*
0.0060	*0.0003*	*0.2100*	*0.0800*	*0.7020*	*0.0010*
0.0060	*0.0003*	*0.1940*	*0.0900*	*0.7890*	*0.0010*
0.0060	*0.0003*	*0.2020*	*0.0830*	*0.7000*	*0.0010*
0.0060	*0.0003*	*0.1980*	*0.0650*	*0.7110*	*0.0010*
0.0060	*0.0003*	*0.1860*	*0.0690*	*0.6710*	*0.0010*
0.0060	*0.0003*	*0.2270*	*0.0640*	*0.6120*	*0.0010*
……	……	……	……	……	……

Eksperiment 4.1 terdapat x1 = hari, x2 = pukul, x3= lv, x4 = hv, x5 = mc, dan *yi = y* class / label di representasikan 1 dan -1.

3. Mengakses label y_i sebagai class.

Eksperiment 4.2 : akses label / class y

```
Columns 1 through 14

     1     1     1     1     1     1     1     1     1     1     1    -1    -1    -1

Columns 15 through 28

    -1     1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1     1     1

Columns 29 through 42

     1     1     1    -1     1     1     1     1     1     1     1     1     1     1

Columns 43 through 56

     1     1     1     1     1     1     1     1    -1     1     1     1     1     1
```

Eksperiment 4.2 di gambarkan label class yang di akses mulai dari kolom 1 sampai 14, dan seterusnya.

4. Menentukan iterasi

i = 50

Eksperiment 4.3 : Max Iteration

```
MaxIter = 100

MaxIter =
   100
```

Dalam eksperiment 4.2 di atas di tentukan banyaknya iterasi sebanyak 100.

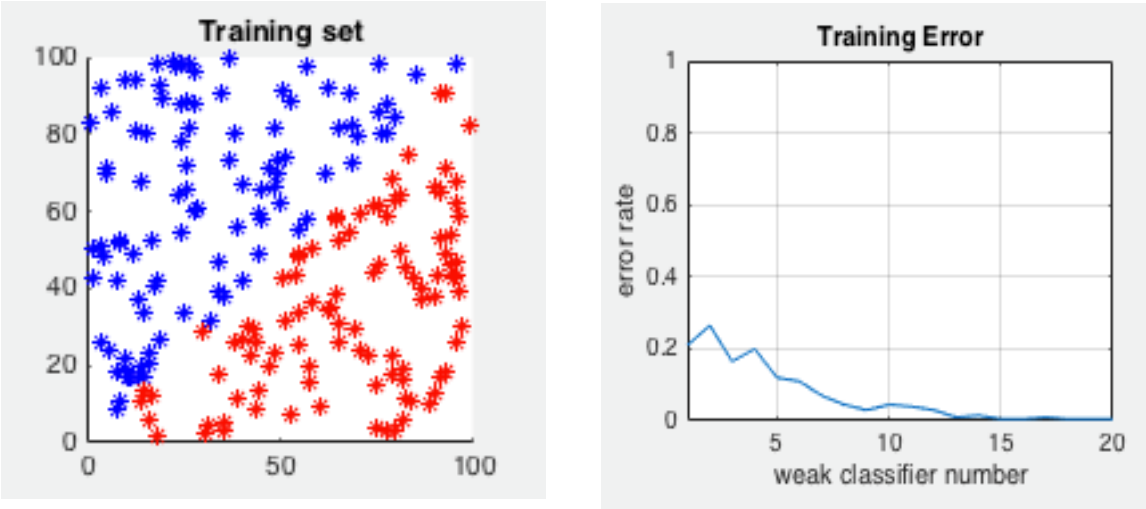
5. Split Dataset menjadi train data dan test data.
   a. Train data

Eksperimen 4.4 : Train data

```
TrainData   = Data(:,1:2:end)
TrainData =
  1.0e+03 *

Columns 1 through 8

    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060
    0.0003    0.0003    0.0003    0.0003    0.0003    0.0004    0.0004    0.0004
    0.1660    0.1900    0.1940    0.1980    0.2270    0.2580    0.1830    1.9870
    0.0630    0.0680    0.0900    0.0650    0.0640    0.0890    0.0910    0.0720
    0.5460    0.7620    0.7890    0.7110    0.6120    0.5380    0.4140    0.4580

Columns 9 through 16

    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060
    0.0004    0.0004    0.0005    0.0005    0.0005    0.0005    0.0005    0.0006
    0.1910    0.2400    0.2060    0.2120    0.2450    0.2480    0.1930    0.1820
    0.0620    0.1040    0.0880    0.0810    0.1030    0.0900    0.0480    0.0740
    0.4320    0.4690    0.4380    0.4160    0.4920    0.5610    0.5760    0.5270

Columns 17 through 24

    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060
    0.0006    0.0006    0.0006    0.0006    0.0007    0.0007    0.0007    0.0007
    0.0350    0.2440    0.1830    0.2640    0.2160    0.1750    0.2290    0.3110
    0.0420    0.0760    0.0460    0.0560    0.0400    0.0340    0.0310    0.0470
    0.5120    0.7210    0.5420    0.9770    0.7970    0.6800    0.7470    1.0610

Columns 25 through 32

    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060    0.0060
    0.0008    0.0008    0.0008    0.0008    0.0008    0.0009    0.0009    0.0009
    0.2950    0.1980    0.1560    0.1190    0.1980    0.1820    0.2320    0.1730
    0.9320    0.1610    0.6300    0.5710    0.5950    0.4900    0.4220    0.4390
   ……        ……        ……        ……        ……        ……
……        ……
```

Train data di gunakan sebanyak 90 % dari dataset, dan di mulai dari kolom 1 sampai 8, sampai kolom terakhir dari 90% dataset.

Ilustrasi 4.2 : Result Train Data



Eksperimen 4.5 : Test Data

ControlData =

1.0e+03 *

Columns 1 through 9

0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060
0.0003	0.0003	0.0003	0.0003	0.0003	0.0004	0.0004	0.0004	0.0004
0.1460	0.2100	0.2020	0.1860	0.1960	0.2410	0.2190	0.2230	0.1360
0.0540	0.0800	0.0830	0.0690	0.0710	0.0990	0.1100	0.0680	0.0590
0.5780	0.7020	0.7000	0.6710	0.5230	0.4640	0.4390	0.5020	0.3930

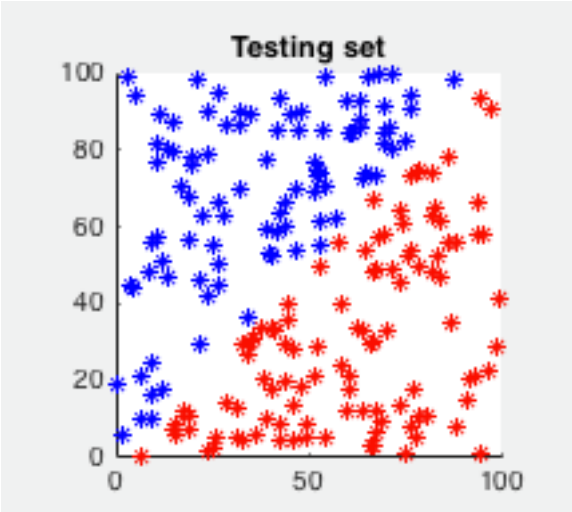
Columns 10 through 18

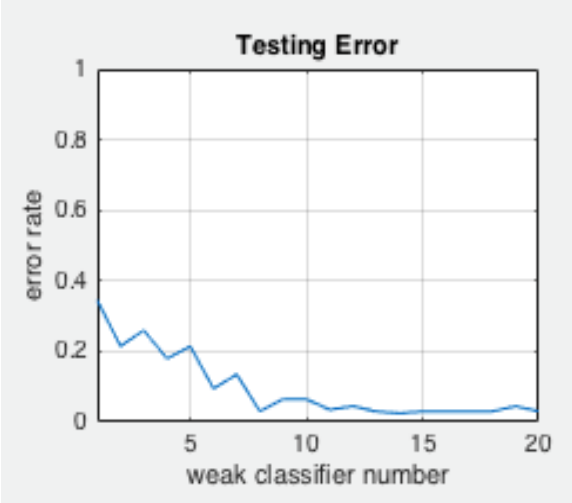
0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060
0.0004	0.0005	0.0005	0.0005	0.0005	0.0006	0.0006	0.0006	0.0006
0.1970	0.2520	0.2350	0.1880	0.2920	0.2290	0.2110	0.2180	0.3280
0.0720	0.0960	0.0850	0.0920	0.0730	0.0790	0.0620	0.0610	0.0810
0.4140	0.4600	0.4710	0.4160	0.5470	0.6340	0.4880	0.6220	0.7280

Columns 19 through 27

0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060	0.0060
0.0006	0.0007	0.0007	0.0007	0.0007	0.0007	0.0008	0.0008	0.0008
0.2870	0.1630	0.2190	0.2130	0.2120	0.2930	0.2310	0.1900	0.1930
0.0660	0.0530	0.0380	0.0340	0.0320	0.0490	0.0450	0.0370	0.0460
0.7390	0.7480	0.8480	0.8370	0.7570	1.2190	1.2440	0.6340	0.7260
0.7390	0.7480	0.8480	0.8370	0.7570	1.2190	1.2440	0.6340	0.7260
.....	.....	.....	.....	.....	.....	.....	.....	
.....	.....	.....						

Test data di gunakan 10% dari keseluruhan dataset.





Membangun Weak Learner Random Forest

a. Menentukan nilai pohon *k* untuk di generate

Eksperiment 4.6 : menentukan K

nTrees = 50

nTrees =50

Pada eksperiment 4.1 di atas ditentukan nilai k sebesar 50 dimana nTrees merupakan sebuah variable.

b. Menentukan prediktor

Eksperiment 4.7 : menentukan prediktor

Hari	Pukul	KendaraanringanLV	KendaraanBeratHV	SepedaMotorMC
6	0.25	166	63	546
6	0.26042	146	54	578
6	0.27083	190	68	762
6	0.28125	210	80	702
6	0.29167	194	90	789
6	0.30208	202	83	700
6	0.3125	198	65	711
6	0.32292	186	69	671
6	0.33333	227	64	612
6	0.34375	196	71	523
6	0.35417	258	89	538
6	0.36458	241	99	464
6	0.375	183	91	414
6	0.38542	219	110	439
6	0.39583	1987	72	458
6	0.40625	223	68	502
..	……	…..	…..	…..

Pada eksperiment 4.2 di atas ditentukan prediktor yang digunakan yaitu hari, pukul, KendaraanLV, KendaraatBeratHV, SepeedaMotorMC. Pada prediktor pukul terdapat tipe data numerik, tetapi yang asli tipe datanya adalah time, agar mempermudah pembacaan maka di buatlah tabel konversi dari tipe data date atau time menjadi numerik seperti di bawah ini:

Tabel 9 : Tabel konversi time ke numerik

Time	Konversi	Numerik
06.00.00	->	0.25
06.15.00	->	0.26042
06.30.00	->	0.27083
06.45.00	->	0.28125
07.00.00	->	0.29167
07.15.00	->	0.30208
07.30.00	->	0.3125
07.45.00	->	0.32292
08.00.00	->	0.33333
08.15.00	->	0.34375
08.30.00	->	0.35417
08.45.00	->	0.36458
09.00.00	->	0.375
09.15.00	->	0.38542
09.30.00	->	0.39583
09.45.00	->	0.40625
10.00.00	->	0.41667
10.15.00	->	0.42708
10.30.00	->	0.4375
10.45.00	->	0.44792
……..	->	……..

c. Menentukan class atau label

Eksperiment 4.8 : Menentukan class / label

Y
—
1
1
1
-1
1
-1
-1
-1

Pada eksperiment 4.3 di atas ditentukan class atau label sebagai supervised learning, dan dalam class atau label bernilai 1 dan -1.

d. Membuat tree $h = (x, \theta_k)$ menggunakan algoritma Ensemble Decision Tree

h = tree

x = predictor

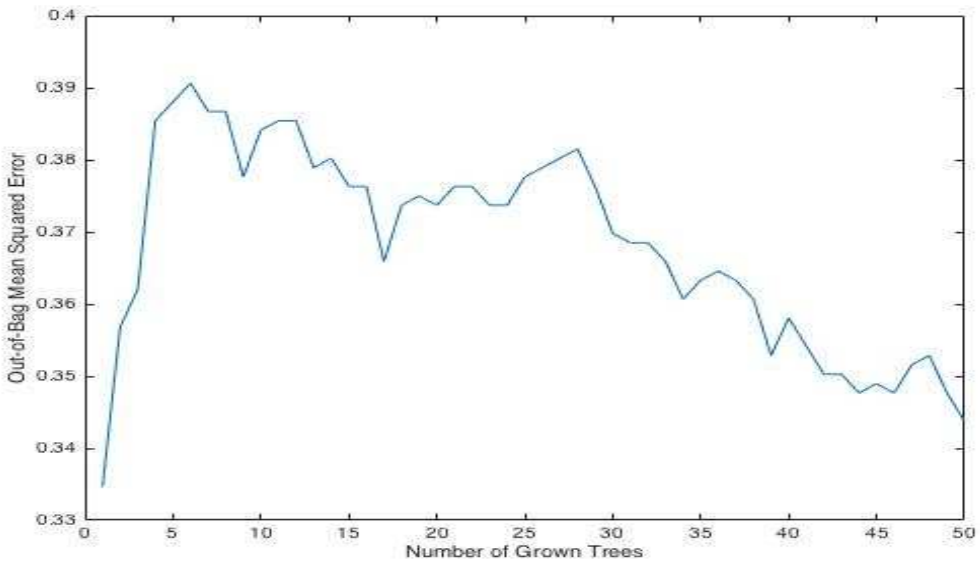
θ_k = banyaknya nilai tree

Eksperiment 4.9: *ensemble decision tree bagger*

```
Tree Bagger
Ensemble with 50 bagged decision trees:
                Training X:            [768x2]
                Training Y:            [768x1]
                    Method:     classification
           NumPredictors:                   2
  NumPredictorsToSample:    2
              MinLeafSize:                  1
           InBagFraction:                   1
  SampleWithReplacement:1
  ComputeOOBPrediction:       1
ComputeOOBPredictorImportance:   0
                Proximity:                 []
               ClassNames:          '-1'          '1'
```

Jika di ilustrasikan hasil dari algoritma ensemble decision tree bisa di lihat pada gambar 4.10 di bawah ini :

Gambar 4.10 : Ilustrasi Tree Bagger



Di jelaskan untuk membangun sebuah random forest bisa menggunakan algoritma decision tree, dan pada eksperiment ini menggunakan ensemble tree bagger, karena dengan menggunakan tree bagger bisa meminimalisir kesalahan dalam klasifikasi.

e. Eksperiment prediksi hanya menggunakan random forest

Setelah membangun sebuah klasifikasi dengan tree bagger, di eksperimenkan untuk memprediksi sehingga terlihat pengklasifikasian yang di eksperimenkan berjalan dengan baik atau tidak. Sebagai contoh pada eksperimen ini di ambil sedikit train data dari dataset berupa hari dan pukul tanpa menyertakan label / class yang akan di prediksi, class atau label bernilai 1 dan -1 tetapi tidak di ikut sertakan karena class atau label tersebut merupakan yang akan di prediksi.

Eksperimen 4.11 : try prediction random forest

```
>> newData1 = [0.0060 0.0003 0.1660 0.0630];
>> predictedClass = C.predict(newData1)

predictedClass =

'-1'
```

Dalam eksperiment 4.11 bisa di ketahui newData1 merupakan variable dalam menampung data yang akan di prediksi dan mempunyai variabel numerik, dan predictClass merupakan variable untuk memprediksi class, kemudian C merupakan variabel penampung dan di dalamnya terdapat algoritma random forest, lebih jelas di dalam variabel C berada pada eksperien 4.4. dan hasil dari prediksi bisa di ketahui bernilai -1.

Menentukan Weight

Setelah melakukan training weak classifier pada 4.1.2.2 maka selanjutnya menentukan bobot / weight dari weak learner yang di hasilkan oleh random fores1, adapun untuk menentukan weight untuk algoritma adaboost sebagai berikut :

$$\text{i: } w_i(i) = \frac{1}{m}$$

i = atribut
w = bobot
m = sample

Eksperimen 4.12 : hasil weight

Columns 1 through 8

17.4190 15.0465 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Columns 9 through 16

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Columns 17 through 21

0.0000 0.0000 0.0000 0.0000 0.0000

GWeights =

Columns 1 through 14

1 1 1 1 1 1 1 1 1 1 1 1 1 1

Columns 15 through 28

1 1 1 1 1 1 1 1 1 1 1 1 1 1

Column 29

1

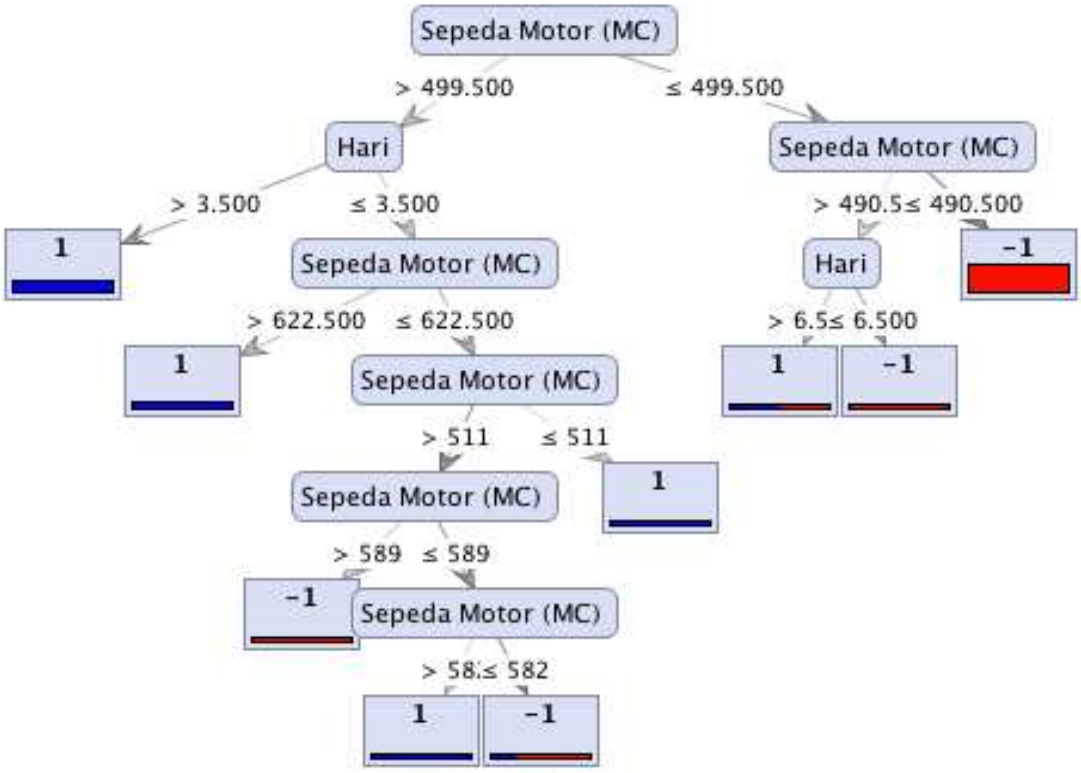
Dari eksperiment 4.12 di atas di hasilkan weight dari proses random forest. Di dalam proses random forest di vote atribut dan class mana yang paling kuat. Pada atribut numerik di hasilkan atribut yang paling banyak di vote begitu juga dengan class atau label yang digunakan.

Training dan Testing menggunakan Algoritma Adaboost

$$\varepsilon_t = \sum_i p_t(i)\,[h_t(x_i) \neq y_i)]$$

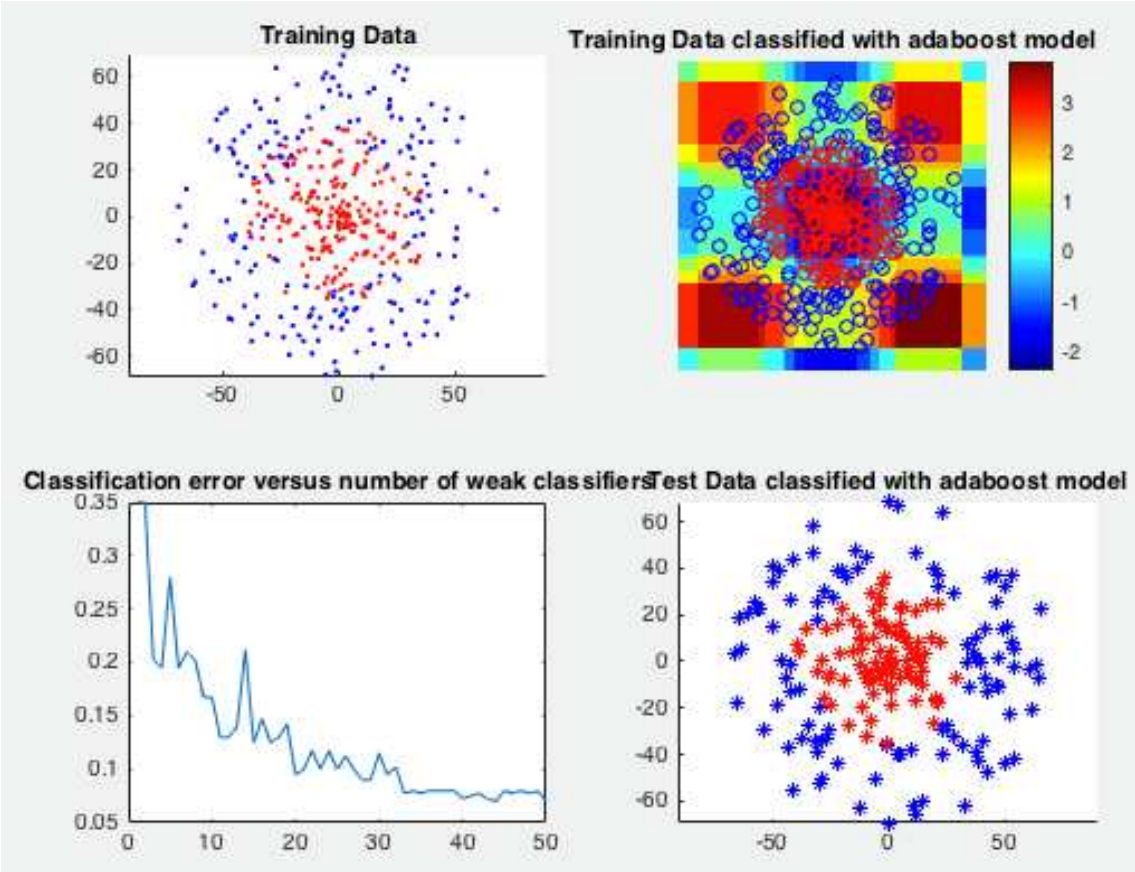
Dari hasil pengolahan algoritma Adaboost untuk training dataset dengan menggunakan hasil weight yang terbaik di dapat hasil sebagai berikut :

Eksperimen 4.13 : Adaboost model Weight = 3.916



Dari eksperimen 4.13 di ketahui bahwa jika jumlah sepeda motor (MC) lebih dari 499.500 maka terjadi kemacetan. Apabila jumlah sepeda motor kurang dari 499.500 maka tidak akan terjadi kemacetan kecuali pada hari sabtu dengan di representasikan pada data dengan angka maka terjadi kemacetan.

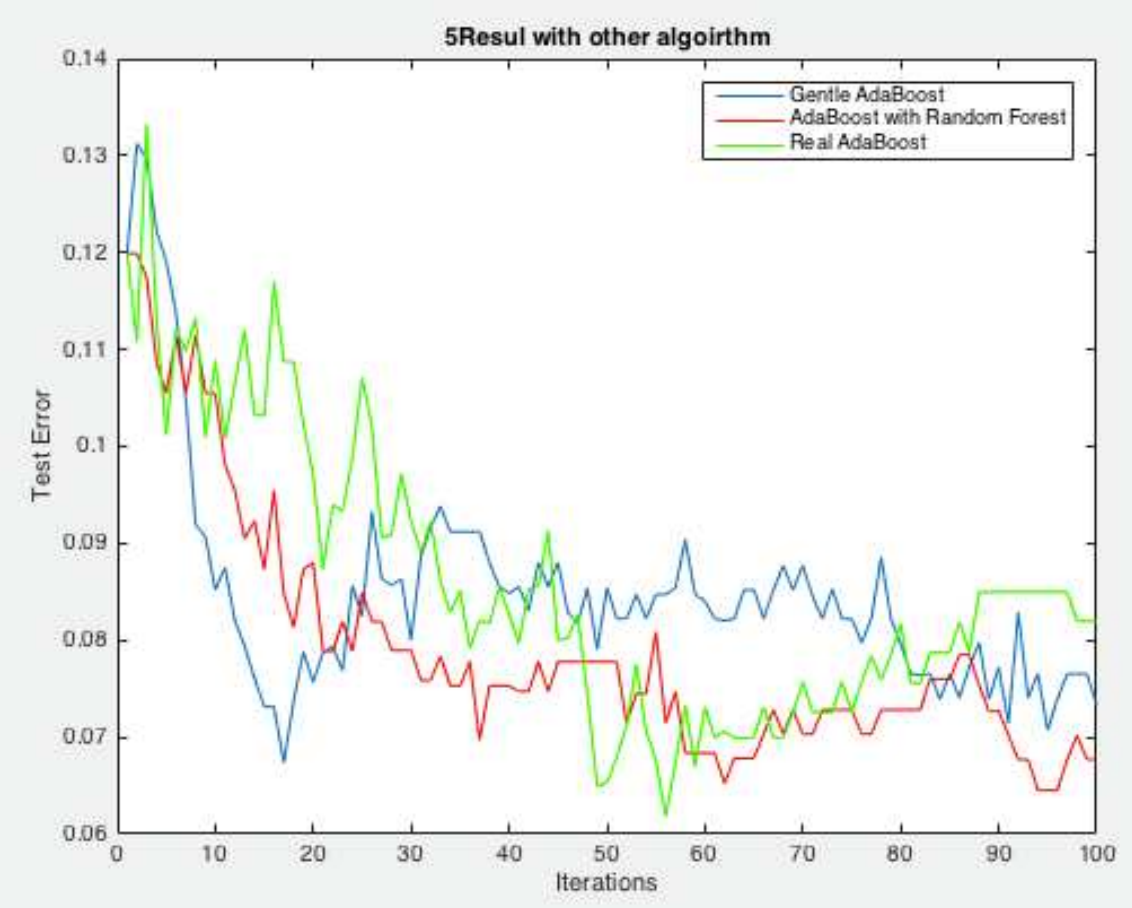
Eksperimen 4.14.



Hasil training dan test menggunakan *Adaboost dan Random forest* bisa di ketahui dari ekperiman 4.14, yang hasilnya menunjukkan tingkat errornya lebih sedikit dari pada menggunakan *Adabosst* versi asli. Untuk lebih detail hasil perbandingan dengan algoritma lain bisa di liha di eksperimen 4.15

Sebagai algoritma pembanding digunakan algoritma adaboost asli yang di sebut dengan *Real Adaboost* dan Algoritma pembanding kedua yang lebih baik dari *Real Adaboost* yaitu *Gentle Adaboost* yang sama – sama hanya menggunakan weak classifier asli dari algoritma tersebut, kedua algoritma tersebut akan di jadikan pembanding dengan algoritma yang digunakan dalam penelitian ini yaitu *Adaboost* dengan *Random Forest,* hasilnya bisa di lihat di bawah ini :

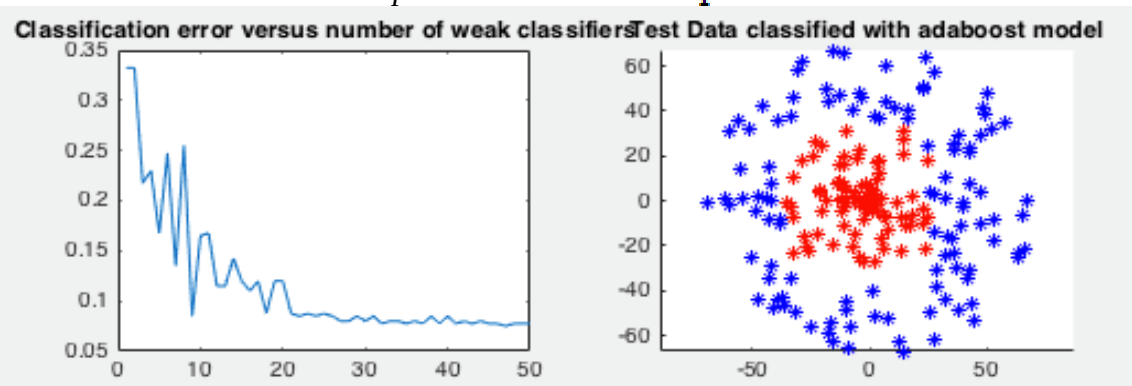
Eksperimen 4.15 : Perbandingan hasil Algoritma



Kalkulasi error dari h_1.

Untuk mengkalkulasi tingkat error dari learning training dari adaboost di gunakan rumus $if\ \varepsilon_t > 1/2$

Eksperiment 4.16 : Error h_1

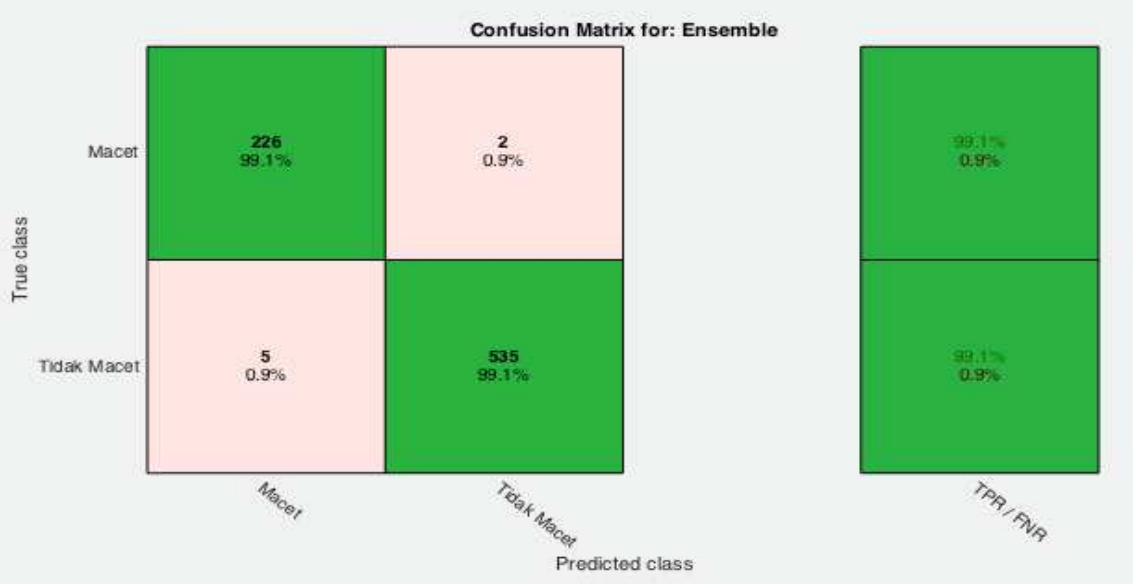


ErrorR=0.0629

Eksperimen 4.16 di hasilkan tingkat error sebesar 0,0629 artinya semakin mendekati 0 semakin kecil pula tingkat errornya.

Evaluasi menggunakan confusion matrix

Eksperimen 4.17 : Confusion Matrix



a. Nilai *True Positive* untuk macet sebesar 226 sedangkan yang tidak macet sebesar 535. Selain nilai True Positive di sebut false negative. Untuk nilai false negative untuk macet sebesar 5 dan untuk tidak macet sebesar 2.
b. Presisi untuk macet adalah 226 di bagi 226 + 2 hasilnya sebesar 0,9912 atau 99,12
c. Class recall untuk macet adalah 226 di bagi 226 + 5 dan hasilnya adalah 0,9783 atau 97,83%
d. Akurasi secara keseluruhan adalah 226 + 535 / 226 + 535 + 2 + 5 = 0,9908 atau 99,08%

KESIMPULAN

Dari hasil analisa metode Adaboost dengan weak learner Random Forest untuk memprediksi arus lalu lintas jangka pendek maka di dapat kesempulan sebagai berikut :

a. Metode Adaboost dan Random Forest sebagai weak learner untuk memprediksi arus lalu lintas jangka pendek mempunyai akurasi sebesar 99,1 %.
b. Metode Adaboost dengan weak learner Random Forest terbukti bisa meningkatkan akurasi dalam pengklasifikasi di dalam weak learner Adaboost.
c. Metdode Adaboost dengan weak learner Random Forest terbukti bisa sebagai alternatif dalam memprediksi data time series untuk arus lalu lintas jangka pendek.

SARAN

Dari hasil penelitian ini masih banyak kekurangan dalam cara bereksperimen dengan metode Adaboost dan Random Forest diantaranya yaitu :

a. Penelitian ini masih belum sempurna dan peneliti hanya mengimplementasikan penelitian hanya pada dataset yang bersifat private, harapan ke depan bisa di implementasikan pada dataset yang memang benar bersifat public.
b. Membandingkan hasil penelitian dengan metode lain yang tingkat akurasi lebih baik.
c. Pada penelitian selanjutnya bisa di harapkan penerapan Adaboost dengan Random Forest bisa di tingkatkan lagi akurasinya dengan beberapa eksperimen dan parameter maupun dataset yang lebih banyak lagi.

DAFTAR PUSTAKA

[1] X. Company, "Transportation Analytics," 2014.
[2] B. P. Statistik, "Statistik Transportasi Darat," 2014.
[3] B. P. Statistik, "Produk Domestik Regional Bruto Kabupaten / Kota di Indonesia 2010 - 2014," 2010 - 2014.
[4] B. Jatim, "Eksum Rencana Induk Jalan," 2012.
[5] T. Li and L. Seng, *Prediction for Short-Term Traffic Flow Based on Optimized Wavelet Neural Network Model*, Vols. 7, No 2.
[6] V. TOPUZ, "Hourly Traffic Flow Prediction by Different ANN Models," Istanbul, 2010.
[7] Y. Wang and C. Yanyan, "Short-Term Traffic Flow Prediction by a Sugeno Fuzzy System Based on Gaussian Mixture Models," Beijing, 2012.
[8] L. Breiman, "Random Forest," Berkeley, 2001.
[9] R. S. Ghadati and S. Santosa, "Prediksi Data Arus Lalu Lintas Jangka Pendek Menggunakan Optimasi Neural Network Berbasis Genetik Algorithm," *Jurnal Tekhnologi Infomasi,* vol. 9 Nomor 2, Oktober 2013.
[10] P. K. Hoong, I. K. T. Tan, O. K. Chien and C.-Y. Ting, "Road Traffic Prediction Using Bayesian Networks".
[11] D. M. S. A. SIDDIQUEE and Hamid-Uz-ZAMAN, "Prediction of Hourly Traffic from Short Count Using Artificial Neural Network and Support Vector Machine," *Journal of Society for Transportation and Traffic Studies (JSTS),* vol. 4 No 2, 2014.
[12] Y. HUANG, "Traffic Flow Forecasting Based on Wavelet Neural Network and Support Vector Machine," *Journal of Computational Information System,* vol. 8 No 8, 2012.
[13] C. M. Bishop, Pattern Recognition and Machine Learning, M. Jordan, P. J. Kleinberg and B. S. lkopf, Eds., New York: Springer, 2006.
[14] U. Fayyad, G. Piatetsky-Shapiro and a. P. Smith, "From Data Mining to Knowledge Discovery in Databases," New York, 1997.
[15] Ó. Marbán, G. Mariscal and J. Segovia, A Data Mining & Knowledge Discovery Process Model, I-Tech Education and Publishin, 2009.
[16] J. Lin, E. Keogh, S. Lonardi and B. Chiu, "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," Riverside, 2003.
[17] L. Rokach and O. Maimon, "Data Mining with Decision Tree Theory and Application," Danvers, Scientific Publishing Co. Pte. Ltd, 2008.

[18] L. Breiman, "Bagging Predictors," Berkeley, 2004.
[19] L. I. Kuncheva and J. J. Rodrıguez, "An Experimental Study on Rotation Forest Ensembles," Bangor, 2007.
[20] B. Sartono and U. D. Syafitri, "METODE POHON GABUNGAN: SOLUSI PILIHAN UNTUK MENGATASI KELEMAHAN POHON REGRESI DAN KLASIFIKASI TUNGGAL," vol. 14 No 1, pp. 1 - 7, April 2010.
[21] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 15 No 5, pp. 771 - 780, September 1999.
[22] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, pp. 427-237, 2009.