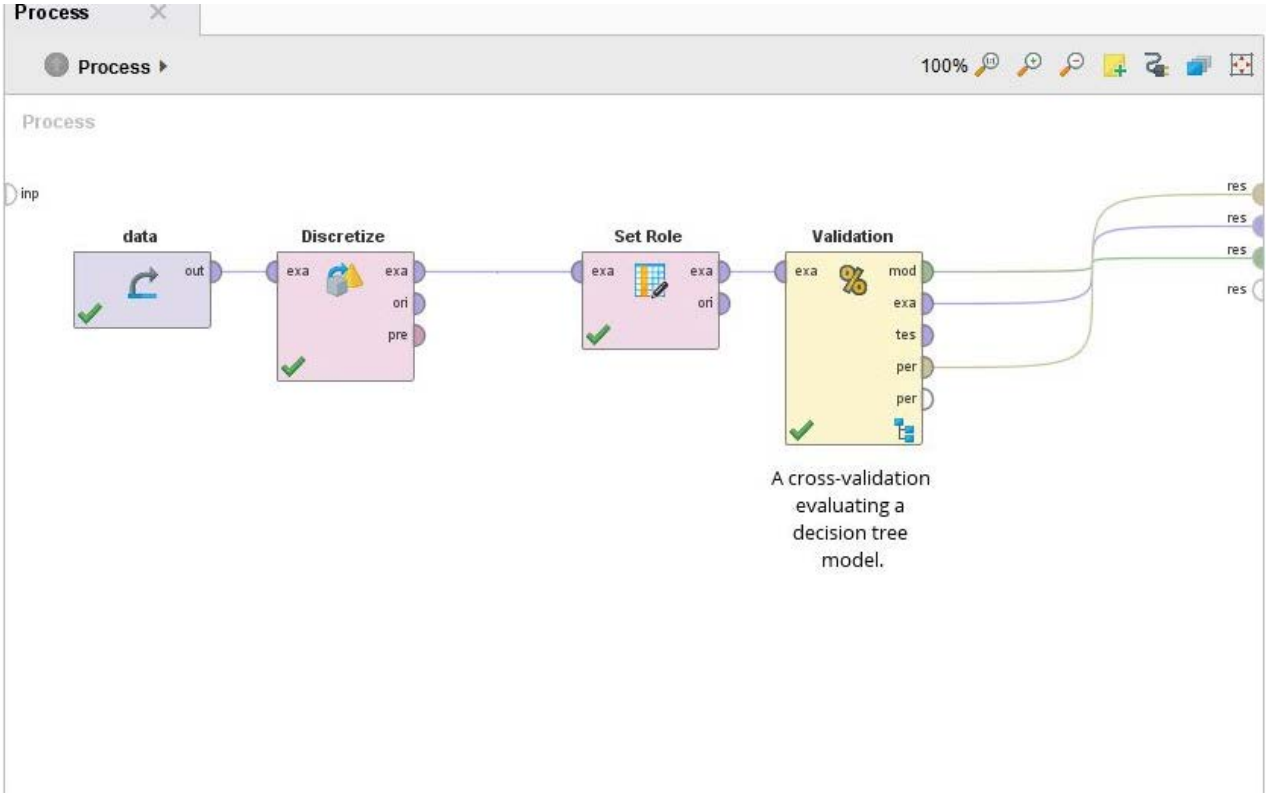
BAB IV
HASIL DAN PEBAHASAN

Jatuh adalah kejadian yang melibatkan seseorang mendadak terbarik, terduduk dilantai, berada ditempat yang lebih rendah, dengan tanpa sadar dan atau menimbulkan luka.Dari penjelasan diatas, mengartikan bahwa saat seseorang terjatuh bisa berada pada posisi yang berbeda-beda. Maka penelitian ini mengambil data contoh dari terjatuh yang memiliki posisi akhir yang berbeda-beda. Kurangnya akurasi prediksi menggunakan accelerometer akan posisi terjatuh menjadi alasan utama untuk penelitian ini, maka untuk meningkatkan akurasi prediksi dilakukan percobaan dan perhitungan dengan metode *decision tree* menggunakan RapidMiner.

3.1 Perhitungan *Decision tree*

RapidMiner merupakan perangkat lunak bersifat terbuka (*open source*) yang menjadi sebuah solusi untuk melakukan analisis terhadap data mining, *text mining* dan analisis prediksi. Data yang sudah kami bagi berdasarkan posisi terakhir objek pada saat pengujian. Hasil pengujian tersebut akan kami hitung ulang menggunakan metode *decision tree*, untuk membuktikan ketepatan prediksi atau hasil pengujian berdasarkan nilai titik koordinat setiap pengujian.



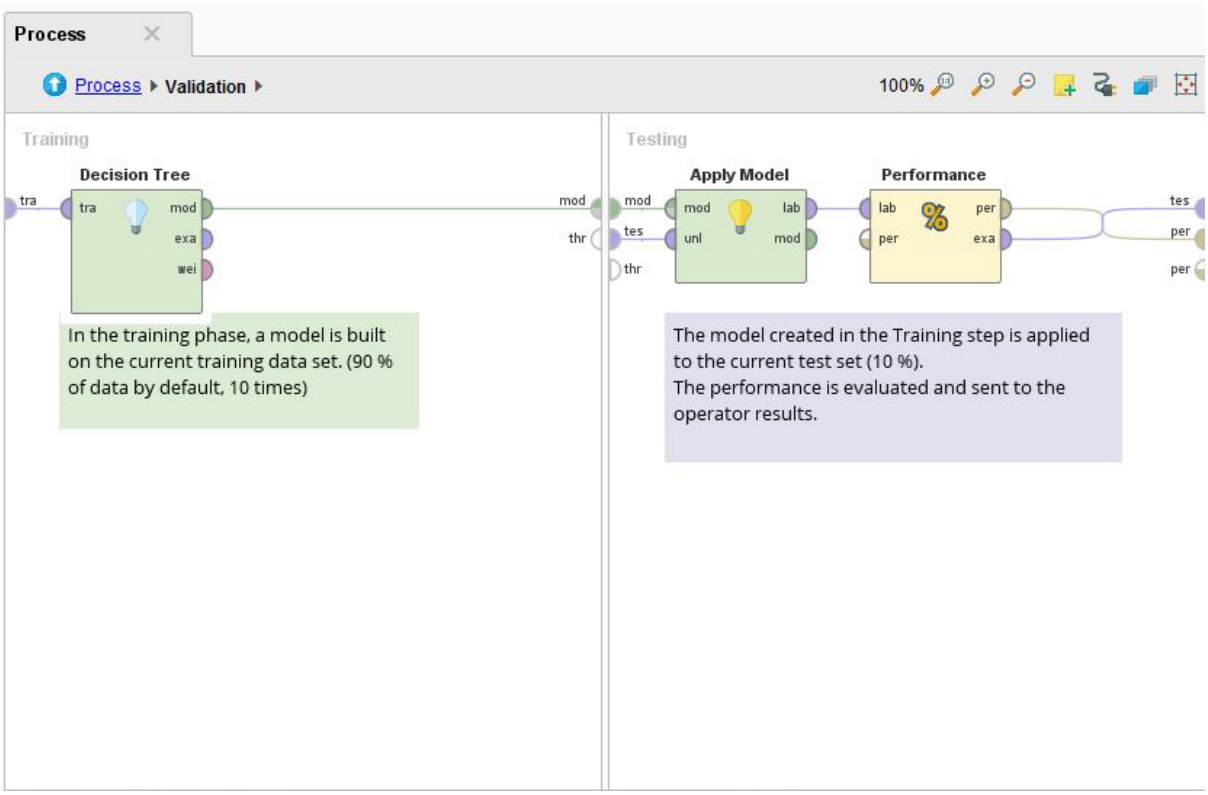
Gambar 4.3.1.1 *Process* 1

Berdasarkan Gambar 4.3.1.1 *Process* 1 ada beberapa operator yang digunakan. Berikut keterangan dan fungsi dari operator-operatortersebut :

a. *Retrieve* (data) : Terdapat 140 data sampel yang telah disiapkan untuk diklasifikasikan dengan metode decision tree yang hasilnya akan diolah lagi untuk menentukan keakurasian pengkategorian menggunakan *confusion matrix.*
b. *Discretize (by binning)* : data awal dibagi atau di diskrit sesuai dengan atribut *numeric* dari masing-masing data. Pembagian yang dilakukan berdasarkan jumlah parameter bins sehingga dapat menentukan jumlah bins yang diperlukan. Data ini akan di diskrit dengan mengacu pada rentang nilai masing – masing kelompok data dengan batasan min(nilai rentang terendah) dan max(nilai rentang tertinggi) sebgai batasan dari masing-masing bins.
c. *Set Role* : setiap data yang ada terbagi menjadi beberapa baris data prediksi sesuai posisi pada saat diambilnya data tersebut, masing-masing data prediksi juga memiliki kolom atribut koordinat sesuai dengan kasus. *Set Role* berfungsi untuk membedakan baris penamaan atribut koordinat dan prediksi posisi yang akan di masukan kedalam kategori

'label'. Agar pada saat pengkategorian data 'label' tidak ikut serta terhitung dan merubah hasil.

d. *Validation (Cross Validation)* : setelah data melewati beberapa operator untuk mempersiapkan data, operator validasi memulai proses pengkategorian. Terbagi menjadi dua *subproses Training* dan *subproses Testing*. Pada *subproses Training* diletakan operator *decision tree*, data akan diuji atau diproses dengan menggunakan algoritma *decision tree*. Selesai diproses data hasil lalu masuk kedalam *subproses Testing* untuk diukur hasil kinerjanya.



Gambar 4.3.1.2 Process 2

Di dalam Validasi seperti pada Gambar 4.3.1.2 Process 2 terdapat beberapa operator pendukung sebagai berikut :

a. Modeling (*Decision tree*) : operator model ini menerapkan algoritma *decision tree* dengan atribut model yang telah disesuaikan, dipilih *decision tree* dengan metode information gain.

b. Apply Model : Operator ini akan menguji data hasil dari operator *decision tree* setelah itu, model ini mendapatkan prediksi pada data yang tidak terlihat atau untuk mengubah data dengan mmenerapkan model *preprocessing*.

a. Performance : pada hasil akhir proses ini performance akan menampilkan hasil. Data hasil pembagian masing-masing data sesuai prediksi atau tidak hingga presentasi keakurasian pengkategorian *decision tree*, dengan memberikan dua hasil pokok yaitu bagan *decision tree* dan tabel *performance*.

Hasil *performance vector* pada gambar dan gambar menunjukkan seberapa besar ketepatan ataupun kesamaan hasil data percobaan jatuh dengan perhitungan *decision tree*, yang dalam kasus ini kami menggunakan *decision tree information gain*.

accuracy: 81.43% +/- 38.89% (mikro: 81.43%)

	true 'walking'	true 'falling'	true 'Lying Do...	true 'Lying'	true 'Sitting'	true 'Standing ...	true 'Standing ...	true Lying'	class pr
pred. 'walking'	16	0	0	0	3	3	0	0	72.73%
pred. 'falling'	0	20	0	0	0	0	1	0	95.24%
pred. 'Lying Do...	1	0	18	0	0	2	2	0	78.26%
pred. 'Lying'	1	0	0	16	0	3	0	1	76.19%
pred. 'Sitting'	1	0	0	0	16	0	1	0	88.89%
pred. 'Standin...	1	0	1	3	1	12	0	0	66.67%
pred. 'Standin...	0	0	1	0	0	0	16	0	94.12%
pred. Lying'	0	0	0	0	0	0	0	0	0.00%
class recall	80.00%	100.00%	90.00%	84.21%	80.00%	60.00%	80.00%	0.00%	

Gambar 4.3.1.3 *Performance*

Tampak pada Gambar 4.3.1.3 *Performance* hasil diatas bahwa ketepatan akurasi *decision tree* dengan hasil uji coba jatuh mencapai 81.43% . performance vector juga memberitahukan berapa dan pada kategori mana saja data yang sesuai atau tidak sesuai antara hasil ujicoba dengan perhitungan *decision tree*.

Dalam hal ini hasil perhitungan yang di dapat dikategorikan sesuai dengan ketepatan prediksinya menjadi *true positive, true negative, false positive,* dan *false negative* .

Tabel 3.1.1 *Performance vector*

Kategori	*True Positive*	*False Positive*	*False Negative*	*True Negative*
Falling	20	1	0	119
Lying	16	5	3	116
Lying Down	18	5	2	115
Sitting	16	2	4	118
Standing up from Lying	12	6	8	114
Standing up from Sitting	16	1	4	119
walking	16	6	4	114

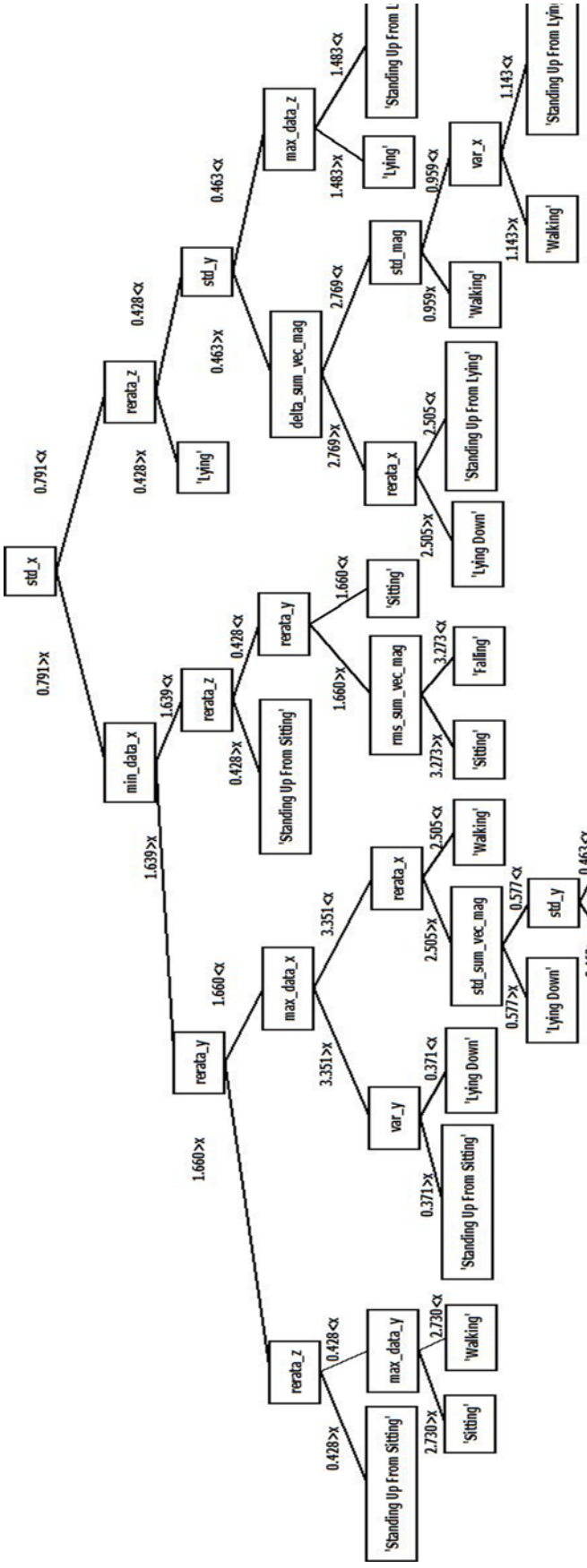
Hasil perhitungan Tabel 3.1.1 *Performance vector* yang dihitung dengan teori *decision tree* informasi gain bila dikelompokkan *true positive, true negative, false positive*, dan *false negative* dari setiap kategori maka akan menjadi seperti pada tabel. *Falling* menjadi kelompok kategori yang paling akurat.

3.2 Perhitungan Manual

Pada bagian ini kita akan mencoba mencoba membuktikan perhitungan dari hasil uji coba menggunakan RapidMiner, akan ada dua pembuktian yang pertama pembuktian penentuan *decision tree* dan yang kedua menghitung hasil validasi dengan menggunakan metode *confusion matrix*.

3.2.1 ***Decision tree***

Hasil yang didapat ada dua yaitu bagan *decision tree* dan *performance*. Seperti yang dapat dilihat pada Gambar 4.3.2.1 Hasil *decision tree*.



Gambar 4.3.2.1 Hasil *decision tree*

hasil ini menjelaskan pemetaan pada Gambar 4.3.2.1 Hasil *decision tree* yang dilakukan oleh RapidMiner dengan data yang telah di masukkan. Dari gambar hasil *decision tree* di awali dengan root pertamanya std_x . perhitungan ini menggunakan metode *decision tree Informasi Gain*.

Untuk membuktikan perhitungan *decision tree* oleh RapidMiner kami akan mencoba menghitung ulang secara manual, dengan rumus *decision tree informasi gain*.

Agar dapat dihitung data-data yang tadi sudah terbagi menjadi tujuh kategori *falling, lying, lying down, sitting, standing up from lying, standing up fromm sitting,* dan *walking,* dibagi lagi menjadi dua jenis yaitu data MIN (nilai data di bawah nilai pengkategorian) dan data MAX (nilai data di atas nilai pengkategorian) . pengkategorian tersebut dimasukan pada setiap data dan nilai klasifikasi yang telah di ujicoba, seperti pada Tabel 3.2.1 Information gain. Dari data-data di atas lalu dihitung menggunakan rumus *decision tree* rapidminer sebagai berikut (Han, Kammber, & Pei, 2012):

$$\text{Entropy (S)} = \sum_{1=l}^{n} -pi\ x\ log_2\ pi \tag{4.1}$$

Keterangan :

S : Himpunan kasus

n : Jumlah partisi dalam S

Pi : Partisi dari Si terhadap S

$$\text{Informasi Gain (S)} = \text{Entropy(S)} - \sum_{i=1}^{n} \frac{|Si|}{|S|} \text{ x info}(S_i) \tag{4.2}$$

Keterangan :

S : Himpunan kasus

n : Jumlah partisi dalam S

Si : Partisi dari S

Perhitungan *decision tree* berdasarkan data dari Lampiran B dari keseluruhan data :

Entropy (Total)

$$\textit{Entropy} = \left(-\frac{20}{140}.log_2\frac{20}{140}\right)+\left(-\frac{20}{140}.log_2\frac{20}{140}\right)+\left(-\frac{20}{140}.log_2\frac{20}{140}\right)+\left(-\frac{20}{140}.log_2\frac{20}{140}\right)+$$

$$=\left(-\frac{20}{140}.log_2\frac{20}{140}\right)+\left(-\frac{20}{140}.log_2\frac{20}{140}\right)+\left(-\frac{20}{140}.log_2\frac{20}{140}\right) = 2.807344922$$

a. Perhitungan Informasi Gain dengan Entropy (min_data_x) berdasarkan Lampiran B =

MAX =

$$= \left(-\frac{20}{40}.log_2\frac{20}{40}\right)+\left(-\frac{0}{40}.log_2\frac{0}{40}\right)+\left(-\frac{0}{40}.log_2\frac{0}{40}\right)+\left(-\frac{15}{40}.log_2\frac{15}{40}\right)+$$

$$=\left(-\frac{0}{40}.log_2\frac{0}{40}\right)+\left(-\frac{5}{40}.log_2\frac{5}{40}\right)+\left(-\frac{0}{40}.log_2\frac{0}{40}\right) \quad = 1.405639062$$

MIN =

$$= \left(-\frac{0}{100}.log_2\frac{0}{100}\right)+\left(-\frac{20}{100}.log_2\frac{20}{100}\right)+\left(-\frac{20}{100}.log_2\frac{20}{100}\right)+\left(-\frac{5}{100}.log_2\frac{5}{100}\right)+$$

$$=\left(-\frac{20}{100}.log_2\frac{20}{100}\right)+\left(-\frac{15}{100}.log_2\frac{15}{100}\right)+\left(-\frac{20}{100}.log_2\frac{20}{100}\right) \quad = 2.48418372$$

InformasiGain = 2.807344922 – (($\frac{40}{140}$) x 1.405639062) – (($\frac{40}{140}$)x 2.48418372)

= 0.6313168186

b. Perhitungan Informasi Gain dengan Entropy (max_data_x) berdasarkan Lampiran B =

MAX =

$$= \left(-\frac{20}{120}.log_2\frac{20}{120}\right)+\left(-\frac{20}{120}.log_2\frac{20}{120}\right)+\left(-\frac{19}{120}.log_2\frac{19}{120}\right)+\left(-\frac{13}{120}.log_2\frac{13}{120}\right)+$$

$$=\left(-\frac{20}{120}.log_2\frac{20}{120}\right)+\left(-\frac{8}{120}.log_2\frac{8}{120}\right)+\left(-\frac{20}{120}.log_2\frac{20}{120}\right) \quad = 2.752135707$$

MIN =

$$= \left(-\frac{0}{20}.log_2\frac{0}{20}\right)+\left(-\frac{0}{20}.log_2\frac{0}{20}\right)+\left(-\frac{1}{20}.log_2\frac{1}{20}\right)+\left(-\frac{7}{20}.log_2\frac{7}{20}\right)+$$

$$=\left(-\frac{0}{20}.log_2\frac{0}{20}\right)+\left(-\frac{12}{20}.log_2\frac{12}{20}\right)+\left(-\frac{0}{20}.log_2\frac{0}{20}\right) \quad = 1.188376372$$

InformasiGain = 2.807344922 – (($\frac{40}{140}$) x 1.405639062) – (($\frac{40}{140}$ x 2.48418372)

= 0.27813

c. Perhitungan Informasi Gain dengan Entropy (rerata_x) berdasarkan Lampiran B =

MAX =

$$= \left(-\frac{20}{93}.log_2\frac{20}{93}\right)+\left(-\frac{20}{93}.log_2\frac{20}{93}\right)+\left(-\frac{0}{93}.log_2\frac{0}{93}\right)+\left(-\frac{15}{93}.log_2\frac{15}{93}\right)+$$

$$=\left(-\frac{19}{93}.log_2\frac{19}{93}\right)+\left(-\frac{4}{93}.log_2\frac{4}{93}\right)+\left(-\frac{15}{93}.log_2\frac{15}{93}\right) \quad = 2.466100054$$

MIN =

$$= \left(-\frac{0}{47}.log_2\frac{0}{47}\right)+\left(-\frac{0}{47}.log_2\frac{0}{47}\right)+\left(-\frac{20}{47}.log_2\frac{20}{47}\right)+\left(-\frac{5}{47}.log_2\frac{5}{47}\right) +$$

$$=\left(-\frac{1}{47}.log_2\frac{1}{47}\right)+\left(-\frac{16}{47}.log_2\frac{16}{47}\right)+\left(-\frac{5}{47}.log_2\frac{5}{47}\right) \quad = 1.859741132$$

InformasiGain = 2.807344922 – (($\frac{93}{140}$) x2.466100054) – (($\frac{47}{140}$)x 1.859741132)

= 0.544818

Perhitungan diatas dari fitur pertama min_data_x hingga fitur terakhir rms_sum_vec_mag, untuk perhitungan fitur std_x hingga fitur rms_sum_vec_mag terdapat pada Lampiran C. Dari perhitungan di atas didapatkan hasil seperti pada Tabel 3.2.1 Information gain.

Tabel 3.2.1 Information gain

ATRIBUT	VALUE	JUMLAH KASUS							Entropi	Information Gain
		falling	Lying	Lying D	Sitting	SUFL	SUFS	walking		
TOTAL	140	20	20	20	20	20	20	20	2.807355	
std_x										0.74144313
54	MAX	0	20	1	1	20	0	12	1.756801	
86	MIN	20	0	19	19	0	20	8	2.260005	
std_sum_vec_hor										0.722530623
51	MAX	0	20	1	1	20	0	9	1.723279	
89	MIN	20	0	19	19	0	20	11	2.292002	
min_data_x										0.631326819
40	MAX	20	0	0	15	0	5	0	1.405639	
100	MIN	0	20	20	5	20	15	20	2.484184	
std_mag										0.620599805
52	MAX	0	20	7	0	19	1	5	1.884857	
88	MIN	20	0	13	20	1	19	15	2.36515	
std_sum_vec_mag										0.601523714
63	MAX	0	20	4	2	18	2	17	2.12039	
77	MIN	20	0	16	18	2	18	3	2.275738	
rerata_x										0.544818221
93	MAX	20	20	0	15	19	4	15	2.4661	
47	MIN	0	0	20	5	1	16	5	1.859741	
var_x										0.54390958
33	MAX	0	17	0	0	15	0	1	1.162872	
107	MIN	20	3	20	20	5	20	19	2.602874	
rerata_z										0.532711469
97	MAX	20	2	15	20	17	3	20	2.536399	
43	MIN	0	18	5	0	3	17	0	1.684171	
rms_sum_vec_mag										0.530618503
51	MAX	0	20	0	10	1	5	15	1.949468	
89	MIN	20	0	20	10	19	15	5	2.464272	
max_data_z										0.40759743
62	MAX	0	7	15	1	19	6	14	2.280354	
78	MIN	20	13	5	19	1	14	6	2.494668	
rerata_y										0.361144666
86	MAX	0	17	20	10	16	9	14	2.531183	
54	MIN	20	3	0	10	4	11	6	2.310883	
max_data_y										0.334943515
63	MAX	0	9	10	4	17	5	18	2.391506	
77	MIN	20	11	10	16	3	15	2	2.538607	
delta_sum_vec_mag										0.308400678
39	MAX	0	6	3	0	11	3	16	2.027113	
101	MIN	20	14	17	20	9	17	4	2.681151	
max_data_x										0.278613406
120	MAX	20	20	19	13	20	8	20	2.752136	
20	MIN	0	0	1	7	0	12	0	1.188376	
std_y										0.212743221
48	MAX	1	14	9	5	13	5	1	2.394186	
92	MIN	19	6	11	15	7	15	19	2.699182	
std_z										0.150274128
22	MAX	1	3	10	0	4	4	0	2.00606	
118	MIN	19	17	10	20	16	16	20	2.778457	
min_data_z										0.129387268
118	MAX	19	13	17	20	16	13	20	2.786999	
22	MIN	1	7	3	0	4	7	0	2.093165	
min_data_y										0.12115668
81	MAX	16	9	10	16	4	13	13	2.710714	
59	MIN	4	11	10	4	16	7	7	2.652541	
var_y										0.097048723
9	MAX	0	4	4	0	0	1	0	1.392147	
131	MIN	20	16	16	20	20	19	20	2.800867	
var_z										0.092187037
8	MAX	0	0	5	0	1	2	0	1.298795	
132	MIN	20	20	15	20	19	18	20	2.801009	

Untuk menentukan root pertama kita cari data yang memiliki nilai information gain tertinggi, dan terlihat pada Tabel 3.2.1 Information gain hasil nilai information gain tertinggi

pada data std_x dengan nilai 0.74144313. hasil perhitungan manual dan perhitungan rapid miner dalam menentukan root pertama sesuai atau sama yaitu std_x.

3.2.2 ***Confusion Matrix***

Dilakukan perhitungan *confution matrix falling* berdasarkan data dari Tabel 3.1.1 *Performance vector* :

Falling	*True Positive*	*False Positive*	*False Negative*	*True Negative*
	20	1	0	119

Sensitifity : $\frac{20}{20}$ = 1 x $\frac{100}{100}$ = 100%

Specifity : $\frac{119}{120}$ = 0.99 x $\frac{100}{100}$ = 99%

Precision : $\frac{20}{20+1}$ = $\frac{20}{21}$ = 0.95 x $\frac{100}{100}$ = 95%

Accuracy : $\left(1\ x\ \frac{20}{20+120}\right)+\left(0.99\ x\ \frac{120}{20+120}\right)$ = $(1\ x\ 0.143)+(0.99\ x\ 0.857)$

= 0.99

= 0.99x $\frac{100}{100}$ = 99%

Hasil perhitungan diatas lalu kami ubah ke dalam bentuk tabel seperti Tabel 3.2.2 *Confusion matrix* sebagai berikut :

Confusion Matrix	*Falling*
Sensitifity	100 %
Specifity	99%
Precision	95%
Accuracy	99%

Tabel 3.2.2 *Confusion matrix*

Perhitungan *accuracy* deteksi jatuh dengan algoritma *decision tree* menggunakan RapidMiner memiliki nilai 81.43%. Berbeda dengan hasil perhitungan *confusion matrix* secara manual yang dapat mencapai hasil *accuracy* 99%. Hasil ini menggabarkan meningkatnya *accuracy falling* dengan menggunakan algoritma *decision tree.*