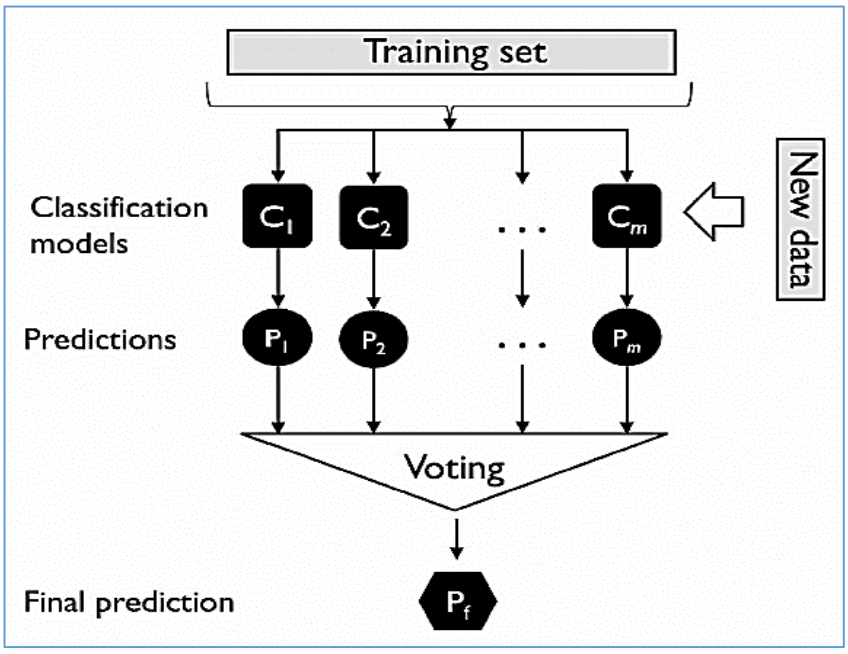
1. *Random Forest*

Ensemble learning merupakan gabungan dari beberapa model untuk menghasilkan hasil yang lebih baik daripada model tunggal (Frank 2017). Salah satu tipe *ensemle learning* adalah *bagging / bootstrap agrregating* dimana dari data asli akan diambil secara acak (*bootstraping*) sebanyak N, selanjutnya dilakukan pelatihan model dari data acak tersebut, proses bootstraping dan pelatihan model akan diulang sebanyak m kali, dan diakhir biarkan semua model memberikan suara (*majority vote*) pada hasil akhir (Frank 2017). Teknik *bagging* digunakan oleh algoritma *Random Forest* yang terdiri dari beberapa Independent Decision Tree dengan varian tinggi untuk membangun model klasifikasi dengan kinerja generalisasi yang lebih baik dan tidak rentan terhadap overfitting (Raschka and Mirjalili 2017). Ilustrasi ensemble learning bisa dilihat pada Gambar 1.



Gambar 1. Ilustrasi *Ensemble Learning* (Raschka and Mirjalili 2017)

Algoritma *Random Forest* menjawab kekurangan yang ada pada algoritma *Decision Tree*. Algoritma *Decison Tree* pada dasarnya menghasilkan diagram alur (flowchart) atau pohon keputusan untuk membuat sebuah keputusan, namun terdapat satu masalah yakni rentan terhadap overfitting (Frank 2017). Penanganan overfitting bisa dilakukan dengan menggunakan algoritma *Random Forest* yang merupakan ensemble dari algoritma *Decision Tree* (Frank 2017). Hal ini dibuktikan pada

penelitian Leo Breiman (2001) bahwa tidak terjadi overfitting pada algoritma *Random Forest* untuk jumlah data yang besar.

1.1 Algoritma *Random Forest*

Random Forest termasuk kedalam ensemble learning dengan mengguanakan teknik *bagging*, berikut ini algoritma *Random Forest* yang terbagi kedalam 4 tahap mudah (Raschka and Mirjalili 2017):

1. Secara acak pilih n sampel data dari data asli pelatihan (*bootstrap sample*) dengan penggantian *(with replacement).*
2. Lakukan pelatihan model pohon keputusan (*Decision Tree*) dari *bootstrap sample*, untuk setiap *node*:
   a. Pilih fitur sebanyak d secara acak tanpa penggantian *(without replacement).*
   b. *Split node* fitur yang memberikan nilai terbaik berdasrkan fungsi obyektik, misalnya : nilai maksimal dari *information gain* fitur d
3. Ulangi langkah 1-2 sebanyak k kali, sesuai dengan model *Decision Tree* yang ingin dibuat pada *Random Forest*
4. Prediksi untuk menetapkan label kelas diambil dari suara terbanyak (*majority vote*) masing-masing pohon.

Catatan : maksud dari *with replacement* (dengan penggatian) adalah data yang sudah terpilih dalam hal ini baris/row boleh muncul kembali, sedangkan maksud dari *without replacement* (tanpa penggantian) adalah data yang sudah terpilih dalam hal ini kolom/fitur data tidak boleh muncul kembali.

Implementasi algoritma *Random Forest* yang digunakan pada penelitian ini, menggunakan library Machine Learning Scikit Learn. Berikut ini persamaan-persamaan yang digunakan (Klikauer 2016) :

1. Jumlah n sample yang digunakan pada *bootstrap sample* adalah sebanyak jumlah baris/*row* data asli

$$n_{bootstrap\ sample} = n_{baris\ data\ asli} \quad \text{..... (1)}$$

2. pengisian d fitur bisa dilakukan dengan berbagai tipe isian (input) :

$$\boldsymbol{d_{fitur}} = func[\text{"sqrt", "log2", "auto"}]\,(n_{fitur\ data\ asli}), input = string$$
$$[1, \dots, n_{fitur\ data\ asli}], input = integer$$
$$[0, \dots, 0.99] * n_{fitur\ data\ asli}, input = float \quad \text{..... (2)}$$

Keterangan :

a. *string* berisi ["sqrt", "log2","auto"], dimana nilai d adalah hasil fungsi dari isian *string* dengan input jumlah n fitur/kolom data asli.
   misal : d = sqrt(jumlah n fitur/kolom data asli)
b. *integer* berisi [1 hingga jumlah n fitur/kolom data asli] , dimana nilai d adalah nilai isian *integer*.
c. *float* berisi [0 hingga 0.99] , dimana nilai d akan dihitung dengan mengalikan nilai isian *float* dan jumlah n fitur/kolom data asli

3. *Decision Tree* yang digunakan pada *Random Forest Scikit Learn* menggunakan algoritma *Cart Tree* (*Classification and Regression Trees*) menghasilkan pohon biner berdasarkan fitur dan ambang batas (*threshold*) dari perolehan *information gain* terbesar di setiap node.

a. Vektor :
   pelatihan $x_i = R^n, i = 1, .., \ell$ ..... (3)
   label $y = R^\ell$ ..... (4)
b. *Split Data* :

   Misal pada node *m* yang direpresentasikan dengan *Q* maka untuk setiap kandidat *split data* dengan fitur j dan *threshold* t_m partisi data menjadi subset $Q_{left}(\theta)$ dan subset $Q_{right}(\theta)$. Nilai *threshold* pada *scikit learn* dihitung dengan mengurutkan semua nilai *threshold* dari kecil ke besar, kemudian menjumlahkan nilai *threshold sekarang* dengan nilai *threshold* yang lebih besar satu tingkat kemudian dibagi dengan 2

   $split\ \theta = (j, t_m)$ ..... (5)
   $Q_{left}(\theta) = (x, y) \mid x_j \leq t_m$ ..... (6)
   $Q_{right}(\theta) = (x, y) \mid x_j > t_m$ ..... (7)

   Dengan :

   $split\ \theta$: split data saat pembuatan model *tree*

j : fitur

t_m : *threshold* pada node *m*

$Q_{left}(\theta)$: partisi data *child left*, dimana nilai *x pada* fitur *j* kurang dari atau sama dengan *threshold* pada node *m*

$Q_{right}(\theta)$: partisi data *child right*, dimana nilai *x pada* fitur *j* lebih besar dari *threshold* pada node *m*

x : data pelatihan

y : label

Impurity child node m dihitung dengan fungsi *H( )*, fungsi *H( )* disini menggunakan gini indeks dengan kasus klasifikasi

$$G_{child}(Q,\theta) = \frac{n_{left}}{N_m} H\left(Q_{left}(\theta)\right) + \frac{n_{right}}{N_m} H\left(Q_{right}(\theta)\right) \quad \text{.....(8)}$$

$$H(X_m) = \sum_k P_{mk}\,(1 - P_{mk}) \quad \text{..... (9)}$$

$$P_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\,(y_i = k) \quad \text{..... (10)}$$

Dengan :

$G_{child}(Q,\theta)$: *information gain child node m*

n_{left} : banyaknya partisi data *child left* (baris/*row*)

n_{right} : banyaknya partisi data *child right* (baris/*row*)

N_m : banyaknya data pada *node m* (baris/*row*)

$Q_{left}(\theta)$: partisi data *child left*, dimana nilai *x pada* fitur *j* kurang dari atau sama dengan *threshold* pada *node m*

$Q_{right}(\theta)$: partisi data *child right*, dimana nilai *x pada* fitur *j* lebih besar dari *threshold* pada *node m*

$H(X_m)$: fungsi untuk mencari *impurity*

k : nilai label klasifikasi [0,1,2 ,.. , k-1]

P_{mk} : proporsi kelas *k* pada *node m*

x : data pelatihan

y : label

I : banyak data yang nilai label y sama dengan kelas *k*

i : region baris pada *node m* [0,1,2, ..., N_m-1]

Split data dilakukan dengan cara memilih parameter fitur dan *threshold* node *m* yang menghasilkan nilai *Informatin Gain Child* terkecil (*min Impurity Decrease*).

$\theta^* = argmin_\theta G_{child}(Q, \theta)$ ..... (11)

Dengan :

θ^* : nilai *Informatin Gain Child* terkecil *node m*

$G_{child}(Q, \theta)$: *Impurity child node m*

c. Information Gain dihitung setelah minimum impurity child ditemukan, sehingga akan mempersingkat waktu komputasi. Selanjutnya IG ini akan digunakan untuk mencari *feature importance* model RF

$IG(Q, \theta) = \frac{N_m}{N} \left(H(Q(\theta)) - G_{child}(Q, \theta)\right)$..... (12)

Dengan :

$IG(Q, \theta)$: *information gain node m*

N_m : banyak data pada *node m* (baris/*row*)

N : banyak data (baris/*row*)

$H(Q(\theta))$: *impurity parent* pada *node m*

$G_{child}(Q, \theta)$: *information gain child node m*

d. Secara rekursif lakukan *split data* hingga kondisi stop terpenuhi, yaitu : maksimal kedalaman pohon terpenuhi, minimal sampel daun terpenuhi, minimal sampel split data terpenuhi.

e. Ulangi langkah a – d sebanyak model *Decision Tree* yang ingin dibuat pada *Random Forest*

1.2 Fungsi fitur terpenting pada *Random Forest*

Random Forest juga bisa digunakan untuk menghitung fitur terpenting (*feature importance*) dengan menghitung rata-rata penurunan *impurity* dari semua pohon keputusan dalam *Random Forest* tanpa memikirkan asumsi apakah data bisa dipisahkan secara linear atau tidak (Raschka and Mirjalili 2017). Berikut persamaan untuk mencari fitur terpenting / *feature importance* (Ronaghan 2018):

$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$ .....(13)

Dengan :

ni_j : fitur terpenting pada *node j*

w_j : bobot sampel pada *node j*

C_j : nilai impurity pada *node j*

$left(j)$: *child node left* pada *node j*

$right(j)$: *child node right* pada *node j*

Sebenarnya persamaan 2.12 dengan 2.11 sama saja, disini penggunaan persamaan 2.12 untuk menyederhanakan saja. Selanjutnya menghitung *importance* untuk setiap fitur pada *Decision Tree* :

$$fi_i = \frac{\sum_j ni_j}{\sum_k ni_k} \quad \text{.....(14)}$$

Dengan:

fi_i : importance untuk fitur *i* dalam *Decision Tree*

ni_j : fitur terpenting pada *node j*

j : *node* dalam *Decision Tree*

k : semua node pada *Decision Tree*

Selanjutnya menghitung *feture importance* pada *Random Forest* dengan menacari rata-rata fi_i pada semua Decision Tree yang ada pada RF.

$$RFfi_i = \frac{\sum_j fi_{ij}}{T} \quad \text{.....(15)}$$

Dengan :

$RFfi_i$: importance untuk fitur *i* dalam *Random Forest*

fi_{ij} : fitur terpenting *i* dalam *Decision Tree j*

j : *Decison Tree* pada *Random Forest*

T : Banyak *Decison Tree* pada *Random Forest*

1.3 Contoh Penggunaan Algoritma *Random Forest*

Model klasifikasi dengan algoritma *Random Forest* akan dibangun berdasarkan dokumentasi library Scikit Learn. Berikut ini penjabaran langkah-langkahnya :

1. Langkah 0 – model akan dibangun dengan 2 pohon, maksimal fitur ketika split data adalah akar kuadrat dari seluruh fitur dan agar mempermudah proses hitung, maka dataset yang digunakan hanya berisi 10 baris dan 6 profil fitur berdasarkan penelitian Gocze, et al. (2013), yaitu: miR-34a, miR-21, miR-27a, miR-155, miR-203 dan miR-221. Lebih jelasnya bisa dilihat pada Tabel 1.

Tabel 1. Tabel Dataset untuk Membangun Model Klasifikasi

No	**Fitur (x)**						**Label (y)**
	miR-21	**miR-27a**	**miR-155**	**miR-203**	**miR-221**	**miR-34a**	***Class***
1	468	240	4	2	12	17	0
2	20814	440	42	35	277	85	1
3	535	210	3	3	62	24	0
4	120815	329	312	486	1097	161	1
5	2658	640	13	2	151	30	0
6	26147	812	1289	31	69	106	1
7	11868	387	33	1	216	30	0
8	5423	241	14	143	125	60	1
9	9332	530	9	0	224	28	0
10	16574	896	742	20	53	185	1

Langkah 1 - Secara acak pilih *n* sampel data dari data asli pelatihan (*bootstrap sample*) dengan penggantian *(with replacement).* Jumlah *n* sampel yang digunakan pada *bootstrap sample* adalah sebanyak jumlah baris/*row* data asli. Hasil dari *bootstrap sample* bisa dilihat pada Tabel 2.

Tabel 2. Tabel *Bootstrap Sample* Pohon ke-1

No	Fitur (x)						Label (y)
	miR-21	**miR-27a**	**miR-155**	**miR-203**	**miR-221**	**miR-34a**	***Class***
1	535	210	3	3	62	24	0
2	9332	530	9	0	224	28	0
3	468	240	4	2	12	17	0
4	535	210	3	3	62	24	0
5	11868	387	33	1	216	30	0
6	16574	896	742	20	53	185	1
7	120815	329	312	486	1097	161	1
8	20814	440	42	35	277	85	1
9	120815	329	312	486	1097	161	1
10	26147	812	1289	31	69	106	1

2. Langkah 2 - Melakukan pelatihan model pohon keputusan (*Decision Tree*) dari *bootstrap sample*. *Decision Tree* yang digunakan pada *Random Forest Scikit Learn* menggunakan algoritma *Cart Tree* (*Classification and Regression Trees*) menghasilkan pohon biner berdasarkan fitur dan ambang batas (*threshold*) dari perolehan *information gain* terbesar di setiap node. Oleh karena itu, untuk setiap *node*:
   a. Pilih fitur sebanyak *d* secara acak tanpa penggantian *(with replacement),* sehinga diperoleh 2 fitur, yaitu: miR-203 dan miR-221. nilai *d* yang dimaksud disini dicari berdasarkan akar kuadrat dari 6 (total fitur). sehingga diperoleh nilai 2 setelah dibulatkan.

$$d = \sqrt{6}$$
$$= 2.449 \approx 2$$

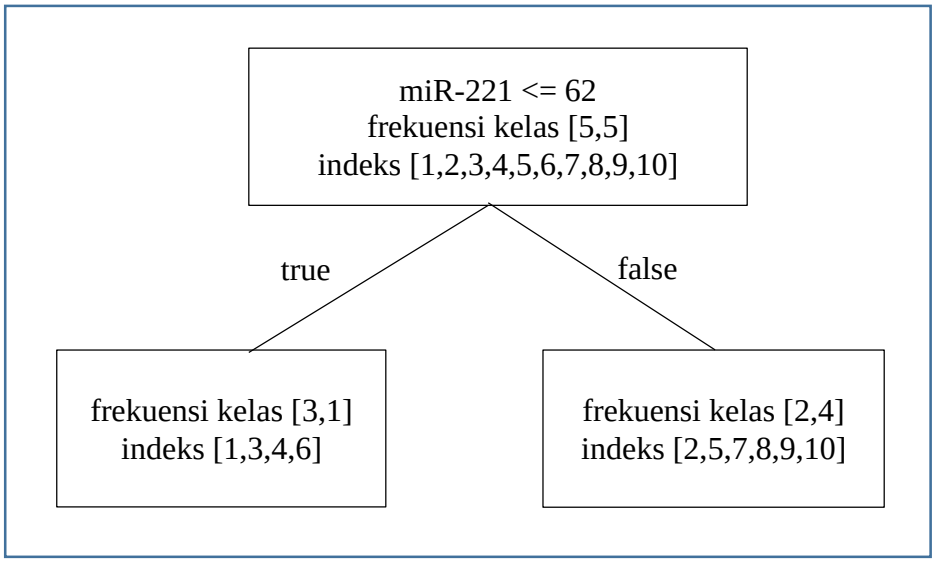
b. *Split node* fitur yang memberikan nilai terbaik berdasrkan fungsi obyektik, misalnya : nilai maksimal dari *information gain* fitur *d*. Berikut langkah yang harus ditempuh untuk melakukan *split node* :

1) Langkah 2.0 - Mencari nilai *information gain child* terkecil / *min impurity decrease* terhadap dua fitur terpilih dilangkah sebelumnya, yaitu: miR-203 dan miR-221. Pencarian nilai *information gain child* terkecil untuk mempercepat proses perhitungan, karena secara umum *information gain* merupakan hasil *information gain parent* dikurangi *information gain child.* Jadi bila *information gain child* semakin kecil maka nilai akhir *information gain* semakin besar.

Pencarian *information gain child* terkecil akan dilakukan satu per satu untuk setiap fitur dengan nilai ambang batasnya (*threshold*). Misalnya fitur miR-221 dan nilai *threshold* 62, maka akan membagi data *bootstrap sample* menjadi dua bagian seperti Gambar 2. Pada *library scikit learn* nilai *threshold* dihitung mengurutkan semua nilai *threshold* dari kecil ke besar, kemudian menjumlahkan nilai *threshold sekarang* dengan nilai *threshold* yang lebih besar satu tingkat kemudian dibagi dengan 2.

$$threshold = 62$$

$$threshold_{\ scikit-learn} = \frac{62 + 69}{2} = 96.5$$



Gambar 2. *Split Data* untuk Fitur miR-221 dan nilai *threshold* 62

2) Langkah 2.1 – menghitung *information gain child* untuk fitur miR-221 dan nilai *threshold* 62. Berdasarkan Gambar 3.6 diketahui bahwa jumlah anak sebelah kiri adalah 4 dengan komposisi kelas 0 sebanyak 3 dan kelas 1 sebanyak 1. Kemudian anak sebelah kanan adalah 6 dengan komposisi kelas 0 sebanyak 2 dan kelas 1 sebanyak 4. Berikut ini perhitungan *information gain child*-nya :

$$G_{child}(Q,\theta) = \frac{n_{left}}{N_m} H\left(Q_{left}(\theta)\right) + \frac{n_{right}}{N_m} H\left(Q_{right}(\theta)\right)$$
$$= \frac{4}{10}(0.375) + \frac{6}{10}(0.444)$$
$$= 0.15 + 0.2664$$
$$= 0.4164$$

$$H\left(Q_{left}(\theta)\right) = H(X_m)$$
$$= \sum_k P_{mk}(1 - P_{mk})$$
$$= P_{m0}(1 - P_{m0}) + P_{m1}(1 - P_{m1})$$
$$= \frac{3}{4}\left(1 - \frac{3}{4}\right) + \frac{1}{4}(1 - \frac{1}{4})$$
$$= \frac{3}{4}\left(\frac{1}{4}\right) + \frac{1}{4}(\frac{3}{4})$$
$$= \frac{3}{16} + \frac{3}{16}$$
$$= \frac{6}{16}$$
$$= 0.375$$

$$H\left(Q_{right}(\theta)\right) = H(X_m)$$
$$= \sum_k P_{mk}(1 - P_{mk})$$
$$= P_{m0}(1 - P_{m0}) + P_{m1}(1 - P_{m1})$$
$$= \frac{2}{6}\left(1 - \frac{2}{6}\right) + \frac{4}{6}(1 - \frac{4}{6})$$

$$= \frac{2}{6}\left(\frac{4}{6}\right) + \frac{4}{6}\left(\frac{2}{6}\right)$$
$$= \frac{8}{36} + \frac{8}{36}$$
$$= \frac{16}{36}$$
$$= 0.444$$

$$P_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = k\right)$$

P_{mk} anak sebelah kiri/$left$:

$$P_{m0} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 0\right)$$
$$= \frac{1}{4}(3)$$
$$= \frac{3}{4}$$
$$P_{m1} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 1\right)$$
$$= \frac{1}{4}(1)$$
$$= \frac{1}{4}$$

P_{mk} anak sebelah kanan/$right$:

$$P_{m0} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 0\right)$$
$$= \frac{1}{6}(2)$$
$$= \frac{2}{6}$$
$$P_{m1} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 1\right)$$
$$= \frac{1}{6}(4)$$
$$= \frac{4}{6}$$

Berdasarkan perhitungan di atas, diketahui bahwa *information gain child* untuk fitur miR-221 dan nilai *threshold* 62 adalah 0.4164.

Langkah 3.1 akan dilakukan terus menerus untuk setiap fitur miR-221 dan miR-203 dan nilai *threshold*-nya.

3) Langkah 2.2 – Mencari nilai terkecil untuk *information gain child* untuk setiap fitur miR-221 dan miR-203 dan nilai *threshold*-nya. Hasil lengkap perhitungan pada langkah 1 disajikan pada Tabel 3

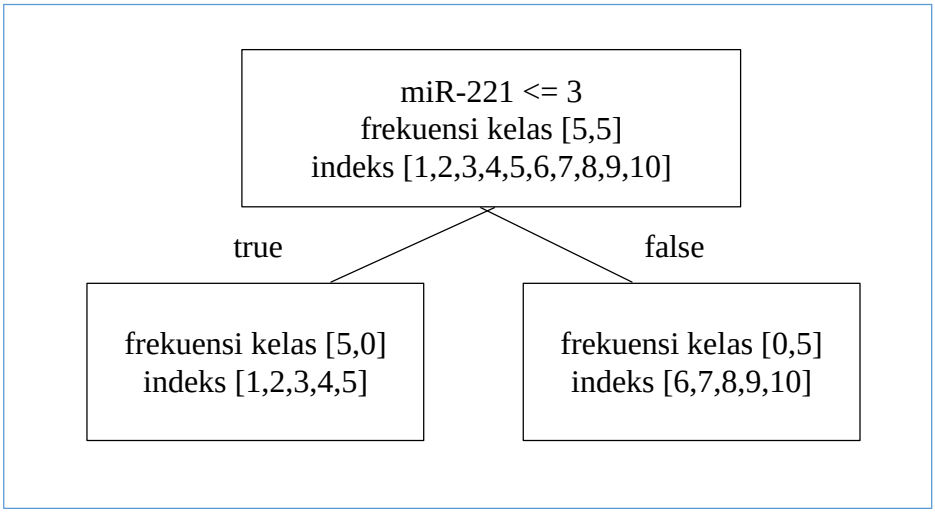
Tabel 3. Tabel *Information Gain Child*

No.	**Fitur**	***Threshold***	***Information Gain Child***
1	miR-221	224	0.2857
2	miR-221	69	0.48
3	miR-221	1097	-
4	miR-221	12	0.4444
5	miR-221	53	0.5
6	miR-221	227	0.375
7	miR-221	216	0.4164
8	miR-221	62	0.4164
9	miR-203	0	0.4444
10	miR-203	1	0.375
11	miR-203	2	0.2857
12	**miR-203**	**3**	**0**
13	miR-203	35	0.375
14	miR-203	486	-
15	miR-203	20	0.16

Berdasarkan Tabel 3.6 diketahui bahwa *information gain child* terkecil dimiliki oleh fitur miR-203 dengan *threshold 3.* Fitur miR-221 *threshold* 1097 dan fitur miR-203 *threshold* 486 tidak memiliki nilai *information gain child.* Dikarenkan kedua nilai *threshold* tersebut adalah yang terbesar di masing-masing fitur, sehingga tidak terjadi *split data.* (ingat rumus *split data* adalah fitur <= *threshold*)

4) Langkah 2.3 – Setelah diketahui *information gain child* terkecil dimiliki oleh fitur miR-203 dengan *threshold 3.* Barulah dicari *information gain* yang nantinya digunakan untuk mencari *feature importance,* berikut ini penjabarannya :

Split data fitur miR-203 dengan *threshold 3* menghasilkan banyaknya anak sebelah kiri adalah 5 dengan komposisi kelas 0 sebanyak 5 dan kelas 1 sebanyak 0. Kemudian anak sebelah kanan adalah 5 dengan komposisi kelas 0 sebanyak 0 dan kelas 1 sebanyak 5. Ilustrasi lengkap bisa dilihat pada Gambar 3.



Gambar 3. *Split Data* untuk Fitur miR-203 dan nilai *threshold* 3

Pada langkah sebelumnya diketahui bahwa *information gain child* fitur miR-203 dengan *threshold 3* adalah 0.

$$IG(Q,\theta) = \frac{N_m}{N}\left(H(Q(\theta)) - G_{child}(Q,\theta)\right)$$
$$= \frac{10}{10}(0.5 - 0)$$
$$= 1\ (0.5)$$
$$= 0.5$$

$$H(Q(\theta)) = H(X_m)$$
$$= \sum_k P_{mk}\ (1 - P_{mk})$$
$$= P_{m0}\ (1 - P_{m0}) + P_{m1}\ (1 - P_{m1})$$

$$= \frac{5}{10}\left(1 - \frac{5}{10}\right) + \frac{5}{10}\left(1 - \frac{5}{10}\right)$$
$$= \frac{5}{10}\left(\frac{5}{10}\right) + \frac{5}{10}\left(\frac{5}{10}\right)$$
$$= \frac{25}{100} + \frac{25}{100}$$
$$= \frac{50}{100}$$
$$= 0.5$$

P_{mk} orang tua/$parent$:

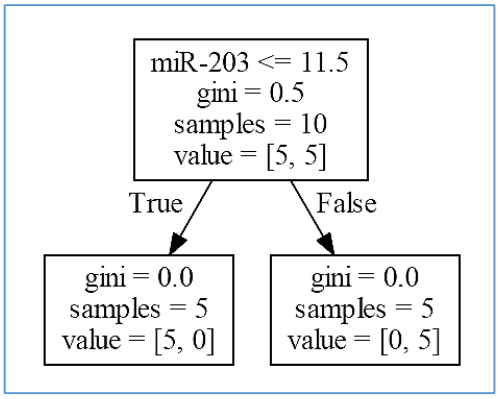
$$P_{m0} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 0\right)$$
$$= \frac{1}{10}(5)$$
$$= \frac{5}{10}$$
$$P_{m1} = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i = 1\right)$$
$$= \frac{1}{10}(5)$$
$$= \frac{5}{10}$$

5) Langkah 2.4 – ulangi langkah 3.3 dan 3.4 hingga kondisi berhenti untuk *split data* terpenuhi. misalnya : nilai maksimal kedalaman pohon terpenuhi, nilai minimal sampel pada *node* terpenuhi, nilai minimal sampel pada *node leaf* terpenuhi, data sudah tidak bisa di split lagi karena *purity* tercapai (*IG = 0*) dan nilai *IG child* lebih kecil atau sama dengan terhadap nilai minimal *IG child* yang telah didefinisikan *user* sebelumnya.

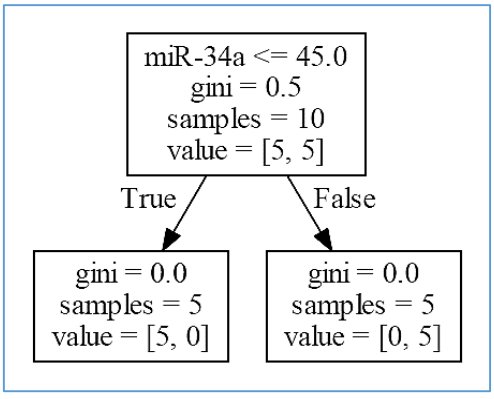
3. Langkah 3 - Ulangi langkah 1-2 sebanyak *k* kali, sesuai dengan model *Decision Tree* yang ingin dibuat pada *Random Forest*. Pada kasus ini nilai *k* adalah 2, sehinga akan diperoleh 2 buah pohon (*Decision Tree*). Ilustrasi pohon ke-1 bisa dilihat pada Gambar 4 dan ilustrasi pohon ke-2 bisa dilihat pada Gambar 5 Sementara itu untuk *bootstrap sample* pohon ke-2 bisa dilihat pada Tabel 4

Tabel 4. Tabel *Bootstrap Sample* Pohon ke-2

No	**Fitur (x)**						**Label (y)**
	miR-21	**miR-27a**	**miR-155**	**miR-203**	**miR-221**	**miR-34a**	***Class***
1	535	210	3	3	62	24	0
2	9332	530	9	0	224	28	0
3	468	240	4	2	12	17	0
4	535	210	3	3	62	24	0
5	11868	387	33	1	216	30	0
6	16574	896	742	20	53	185	1
7	120815	329	312	486	1097	161	1
8	20814	440	42	35	277	85	1
9	120815	329	312	486	1097	161	1
10	26147	812	1289	31	69	106	1



Gambar 4. Hasil Pohon ke-1



Gambar 5. Hasil Pohon ke-2

Setelah diketahui semua hasil pohon (*Decision Tree*) pada *Random Forest*, proses menghitung *feature importance* bisa dilakukan. Berikut penjabarannya :

1) Mencari *IG* untuk setiap node di setiap pohon (*Decision Tree*)

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} = IG(Q,\theta)$$

Pohon ke-1 :

$$ni_{miR-203} = \frac{N_m}{N}\left(H\big(Q(\theta)\big) - G_{child}(Q,\theta)\right)$$

$$= \frac{N_m}{N}\left(H\big(Q(\theta)\big) - \frac{n_{left}}{N_m} H\left(Q_{left}(\theta)\right) - \frac{n_{right}}{N_m} H\left(Q_{right}(\theta)\right)\right)$$

$$= \frac{10}{10}\left(0.5 - \frac{5}{10}(0) - \frac{5}{10}(0)\right)$$

$$= 1\,(0.5 - 0 - 0)$$

$$= 0.5$$

Pohon ke-2 :

$$ni_{miR-34a} = \frac{N_m}{N}\left(H\big(Q(\theta)\big) - G_{child}(Q,\theta)\right)$$

$$= \frac{N_m}{N}\left(H\big(Q(\theta)\big) - \frac{n_{left}}{N_m} H\left(Q_{left}(\theta)\right) - \frac{n_{right}}{N_m} H\left(Q_{right}(\theta)\right)\right)$$

$$= \frac{10}{10}\left(0.5 - \frac{5}{10}(0) - \frac{5}{10}(0)\right)$$

$$= 1\,(0.5 - 0 - 0)$$

$$= 0.5$$

2) Mencari *feature importance* untuk setiap pohon (*Decision Tree*)

$$fi_i = \frac{\sum_j ni_j}{\sum_k ni_k}$$

Pohon ke-1 :

$$fi_{miR-203} = \frac{ni_{miR-203}}{ni_{miR-203}}$$

$$= \frac{0.5}{0.5}$$

$$= 1$$

Pohon ke-2 :

$$fi_{miR-34a} = \frac{ni_{miR-34a}}{ni_{miR-34a}}$$
$$= \frac{0.5}{0.5}$$
$$= 1$$

3) Selanjutnya menghitung *feture importance* pada *Random Forest* dengan menacari rata-rata fi_i pada semua Decision Tree yang ada.

$$RFfi_i = \frac{\sum_j fi_{ij}}{T}$$

$$RFfi_{miR-203} = \frac{fi_{imiR-203}}{2}$$
$$= \frac{1}{2}$$
$$= 0.5$$

$$RFfi_{miR-34a} = \frac{fi_{imiR-34a}}{2}$$
$$= \frac{1}{2}$$
$$= 0.5$$

4) Hasil akhir perhitungan *feature importance* pada *Random Forest* diperoleh 2 fitur terpenting, yaitu : miR-203 dengan nilai 0.5 dan miR-34a dengan nilai 0.5

5. Langkah 4 - Prediksi untuk menetapkan label kelas diambil dari suara terbanyak (*majority vote*) masing-masing pohon.