BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Dalam penelitiannya mengenai klasifikasi dokumen, Shekar (2009) menyimpulkan beberapa hal. Dengan melatih lebih banyak dokumen maka akan mengurangi *Mapping Time.* Tingkat akurasi dalam pengujian dokumen yang akan diklasifikasi dapat ditingkatkan apabila dokumen yang digunakan untuk pelatihan berjumlah besar. Jadi semakin banyak dokumen yang dilatihkan semakin akurat pula hasil klasifikasi.

Ariana (2011) dalam penelitiannya berpendapat bahwa metode *Self Organizing Map* dapat digunakan untuk mengetahui pola-pola cluster dalam suatu basis data. Menurutnya metode *Self Organizing Map* adalah metode *black box* yang berarti kita tidak tahu bagaimana metode tersebut bekerja untuk menghasilkan suatu *output*.

Westerlund (2005) berpendapat *Self Organizing Map* tidak bisa ditujukan untuk menghasilkan klasifikasi yang optimal. Metode *unsupervised learning* dapat digunakan untuk klastering, tetapi jika data masukan termasuk dalam kelas yang ditentukan maka metode *supervised learning* lebih efektif.

Dalam penelitiannya, Hermanto (2008) menjelaskan bahwa *Self Organizing Map* dapat digunakan untuk klasifikasi *aves*. Penelitian ini menggunakan *input* ciri-ciri dari masing-masing ordo *aves* yang telah dikodekan. Dengan pengkodean *input* dan bobot awal yang tepat, metode *Self Organizing Map* dapat digunakan untuk pengelompokan data *aves* dengan cepat dan tepat.

Dalam penelitian yang dilakukan oleh Madarum (2006) didapat beberapa kesimpulan. Klaster data tidak dipengaruhi ukuran *output SOM.* Nilai *Cluster Recall* (CR) berbanding terbalik dengan jumlah klaster yang terbentuk sedangkan

Learning Rate berbanding lurus dengan *Cluster Precision* (CP). Nilai CR dan CP yang ideal adalah mendekati satu. Hasil yang dicapai pada akhir penelitiannya adalah tidak semua spesimen yang ada dalam satu famili terklaster dalam klaster yang sama.

2.2 Landasan Teori

2.2.1 Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah sistem pengolah informasi yang bekerja seperti cara kerja dari otak manusia yang sesungguhnya. Otak manusia terdiri dari jutaan sel saraf (neuron). Setiap *neuron* bekerja seperti *processor* dan terhubung dengan neuron lainnya. Hal inilah yang menjadi acuan dibangunnya sistem jaringan saraf tiruan. Di dalam jaringan saraf tiruan, setiap neuron mempunyai prinsip kerja yang sama yaitu setiap neuron selalu menerima *input* dan selalu menghasilkan *output*.

Menurut Laurene Fausett (1994), jaringan saraf tiruan merupakan model matematis dari jaringan saraf manusia sesungguhnya berdasarkan dari beberapa pendapatnya. Pengolahan infomasi terjadi pada setiap elemen dasar yang disebut neuron. Setiap neuron dihubungkan oleh sinyal yang mempunyai bobot masing-masing. Setiap neuron menerapkan fungsi aktivasi untuk mendapatkan nilai *output*.

2.2.2 *Self Organizing Map*

Disebut juga sebagai jaringan Kohonen yang ditemukan pada tahun 1982 oleh Prof. Teuvo Kohonen. *Self Organizing Map* merupakan salah satu metode dalam jaringan syaraf tiruan yang menggunakan pembelajaran tanpa arahan

(*unsupervised learning*) yang bersifat kompetitif. Artinya hasil target untuk *input* data tidak disediakan, tetapi ditentukan berdasarkan *unit* pemenang .

Proses yang dilakukan metode SOM adalah proses penyesuaian bobot setiap neuron *output* yang ada. Bobot awal suatu neuron *output* (*cluster*) bisa ditentukan secara manual ataupun *random* (acak). Neuron *output* yang bobotnya paling mirip dengan neuron *input* dipilih sebagai pemenang. Neuron pemenang dan juga neuron tetangganya akan diperbaharui bobotnya. Sedangkan bobot neuron lainnya akan tetap.

Berikut adalah algoritma *Self Organizing Map* (SOM):

Step 0. Inisialisasi bobot (bisa *random*)
Tentukan topologi serta parameter
Tentukan Learning rate

Step 1. Bila kondisi terpenuhi lakukan step 2-8

Step 2. Untuk setiap input vector lakukan step 3-5

Step 3. Untuk setiap j hitung jarak dengan rumus

$$D(j) = \sum_i \left(w_{ij} - x_i\right)^2$$

Step 4. Pilih indeks J dengan D(J) paling kecil

Step 5. Update bobot j dan tetangganya dengan rumus

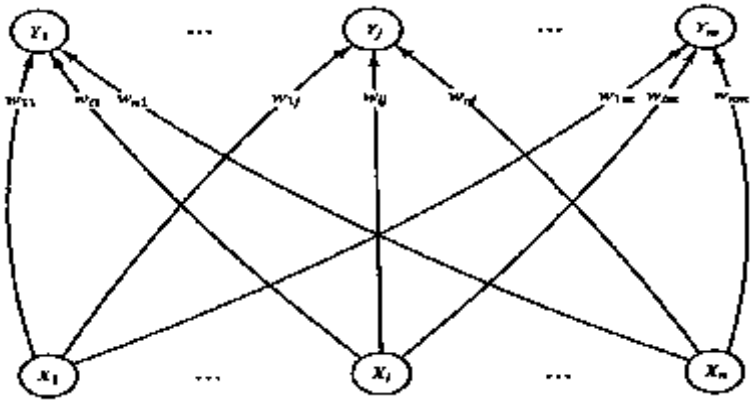
$$w_{ij}(new) = w_{ij}(old) + \alpha\left[x_i - w_{ij}(old)\right]$$

Step 6. Update *Learning Rate*

Step 7. Kurangi radius tetangga topologi

Step 8. Berhenti

Arsitektur SOM terdiri dari satu *layer input* dan satu *layer output*. Setiap *unit* pada *layer input* dihubungkan dengan semua *unit* pada *layer output* dengan bobot W_{ij} seperti pada Gambar 2.1. Jumlah *unit input* adalah sebanyak data yang akan dilakukan *clustering*. Sedangkan untuk jumlah *unit output* adalah berdasarkan banyaknya *cluster*.



Gambar 2.1 Arsitektur SOM

Dikutip dari : Fausett, Laurence (1994). *Fundamental of Neural Networks, Architectures, Algorithms, and Applications*, hlm 170.

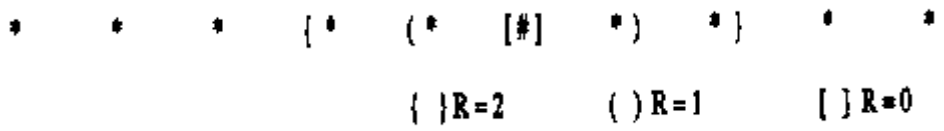
Keterangan Gambar 2.1 :

X_{1-n} : fitur neuron *input*

Y_{1-m} : neuron *output*

W_{ij} : bobot hubungan neuron *input* dan neuron *output*

Metode *Self Organizing Map* mempunyai beberapa topologi yang bisa digunakan untuk penyelesaian kasus antara lain topologi 1-dimensi, topologi 2-dimensi dan topologi *hexagonal*. Pemilihan topologi untuk setiap kasus harus tepat guna memaksimalkan hasil penelitian. Topologi yang digunakan untuk penelitian ini adalah topologi 1-dimensi seperti pada Gambar 2.2.



Gambar 2.2 Topologi 1-dimensi

Dikutip dari : Fausett, Laurence (1994). Fundamental of Neural Networks, Architectures, Algorithms, and Applications, hlm 170.

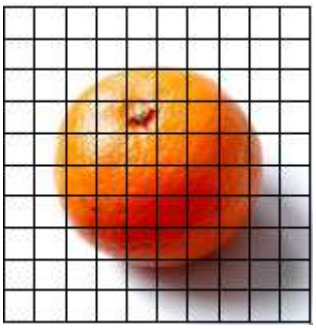
2.2.3 Citra Digital

Citra adalah gambar pada bidang dua dimensi. Citra didapatkan ketika cahaya mengenai objek lalu objek memantulkan sebagian cahaya itu ke sensor optik. Sebuah citra digital adalah citra dalam bentuk digital dan dapat disimpan pada memori komputer. Untuk mendapatkan citra digital diperlukan beberapa perangkat khusus antara lain kamera digital, *scanner*, atau sensor lainnya. Di dalam computer, citra digital disimpan dalam dalam format tertentu.

Format-format tersebut menunjukan cara penyimpanan sebuah citra digital. Contoh dari format citra digital antara lain .jpg, .bmp, .png, Dalam penelitian ini akan digunakan citra dengan format .bmp.

2.2.4 Implementasi *Self Organizing Map* untuk Klasifikasi Citra

Input dari sistem adalah berupa citra digital. Citra asli adalah milik penulis yang diperoleh dari hasil *capture* kamera digital. Citra asli mempunyai format JPEG dengan ukuran asli 4608 x 3456 piksel. Citra tersebut selanjutnya akan diproses dahulu dengan menggunakan Paint sebelum digunakan sebagai masukan ke dalam sistem. Dengan menggunakan software Paint, citra dirubah menjadi format BMP dengan ukuran 100 x 100 piksel.

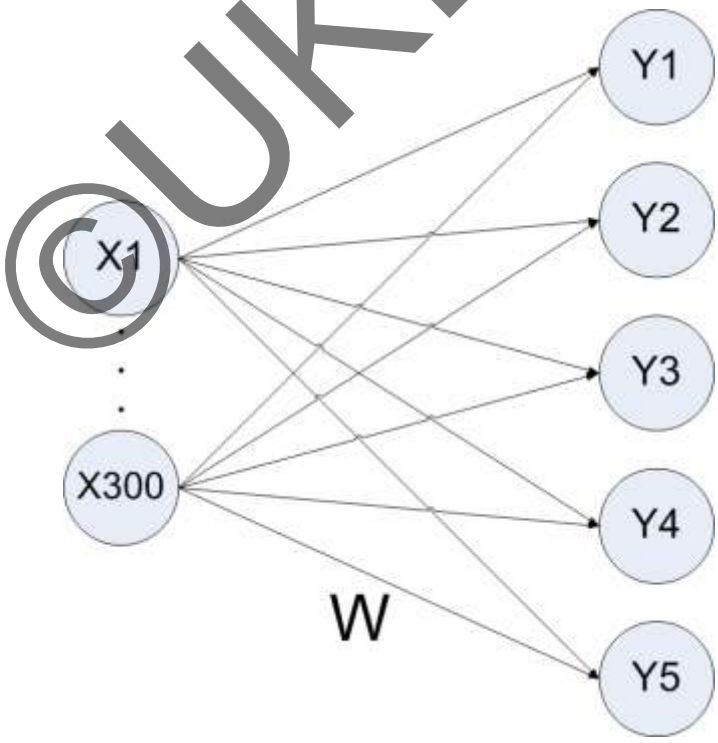


Gambar 2.3 Citra 100 Blok

Selanjutnya citra akan dibagi menjadi 100 blok kecil dengan ukuran 10 x 10 piksel untuk setiap blok seperti pada Gambar 2.3. Proses pembagian citra ke dalam 100 blok ini dilakukan di dalam sistem yang dibangun.

Pada Gambar 2.3 citra telah dibagi menjadi 100 blok berukuran 10 x 10 piksel. Setiap blok kecil dalam citra akan diwakili oleh sebuah nilai. Nilai yang mewakili setiap blok ini merupakan warna rata-rata dalam blok tersebut. Jadi dari sepuluh ribu piksel di dalam satu blok akan diambil nilai *red, green,* dan *blue* lalu dihitung rata - ratanya. Nilai - nilai tersebut lalu digabungkan menjadi satu dan digunakan untuk mewakili satu citra buah jeruk.

Untuk penelitian ini jumlah citra yang digunakan sebanyak 100 dengan jumlah setiap jenis jeruk sebanyak 20. Penelitian ini akan digunakan topologi 1-dimensi. Jumlah *cluster* (neuron *output*) sebanyak lima berdasarkan jumlah jenis jeruk yang digunakan. Sedangkan jumlah input yang digunakan bervariasi mulai dari 25, 50, 75, dan 100. Pada Gambar 2.4 banyaknya fitur yang digunakan untuk penelitian adalah 300.



Gambar 2.4 Arsitektur Penelitian

Gambar 2.4 adalah arsitektur yang digunakan dalam penelitian ini. X1-X300 adalah fitur dalam setiap neuron *input* yang digunakan dalam penelitian ini. Y1-Y5 adalah neuron *output* (*cluster*), jumlah *cluster* adalah lima yang merupakan jumlah jenis buah jeruk yang digunakan untuk penelitian ini. W adalah bobot hubungan antara neuron *input* dan neuron *output*. Bobot awal penelitian akan ditentukan oleh penulis dan akan selalu berubah selama metode *Self Organizing Map* (SOM) dilakukan.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Spesifikasi Sistem

3.1.1 Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun sistem ini adalah :

1. Sistem Operasi Windows 7
2. Borland Delphi 7
3. Paint
4. Microsoft Access 2007

3.1.2 Perangkat Keras

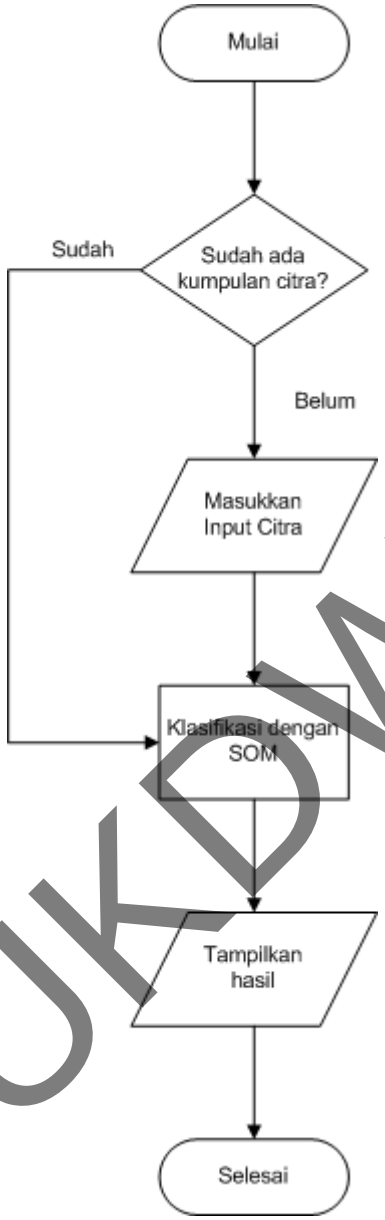
Perangkat keras yang digunakan untuk membangun sistem in adalah :

1. Intel (R) Core i3 2.27 GHz
2. Memory 3.00 GB
3. Harddisk 320 GB

3.2 *Flowchart*

3.2.1 *Flowchart* Program

Flowchart program merupakan gambaran umum pemakaian sistem program ini dalam penelitian ini. Di dalam *flowchart* dijelaskan bahwa bila belum ada kumpulan data citra maka harus dilakukan pendataan citra terlebih dahulu. Setelah semua citra yang akan diklasifikasikan terkumpul maka proses klasifikasi dapat langsung dilakukan. *Flowchart* program dapat dilihat pada Gambar 3.1

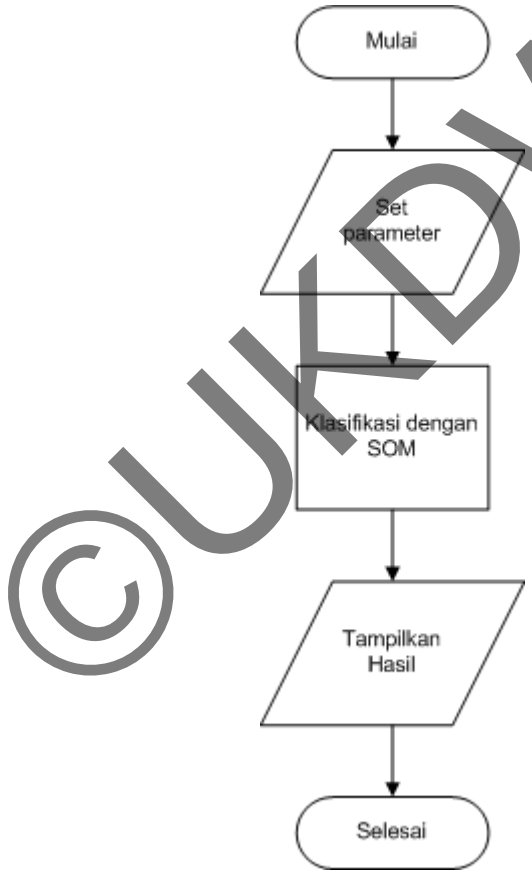


Gambar 3.1 *Flowchart* Program

Pada sistem ini peran pengguna (*user*) hanya sebatas memasukkan citra ke dalam database, mengatur parameter klasifikasi dan melihat hasil klasifikasi. Untuk proses klasifikasi, pengguna hanya dapat mengandalkan metode *Self Organizing Map* untuk melakukannya secara mandiri.

3.2.2 *Flowchart* Klasifikasi

Flowchart ini merupakan gambaran umum proses klasifikasi yang dilakukan dengan metode *Self Organizing Map* (SOM). Proses klasifikasi akan menggunakan data citra yang sebelumnya telah disimpan pada *database*. Sebelum melakukan klasifikasi pengguna wajib mengisi parameter berupa jumlah *epoch*, *learning rate* dan jumlah data. Setelah proses klasifikasi selesai dilakukan pengguna dapat melihat hasil klasifikasi tersebut. Flowchart klasifikasi dapat dilihat pada Gambar 3.2.

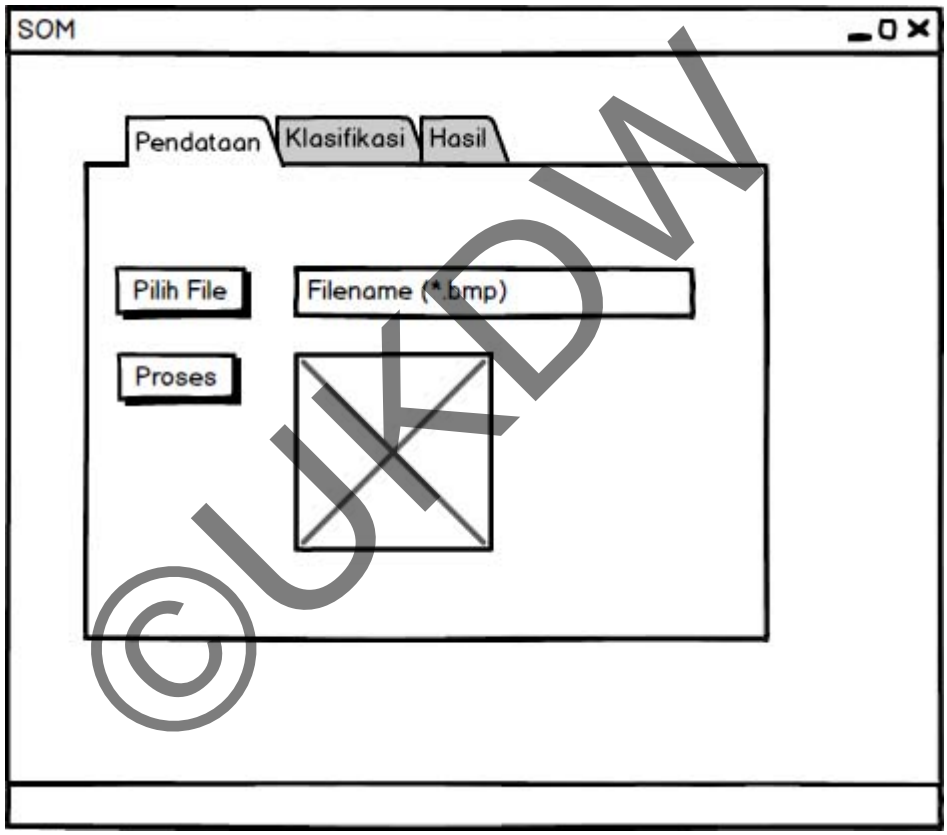


Gambar 3.2 *Flowchart* Klasifikasi

3.3 Rancangan Antarmuka

3.3.1 *Form* Pendataan

Form pendataan digunakan untuk mengumpulkan citra-citra yang akan dipakai untuk proses klasifikasi. *Form* ini khusus digunakan untuk memilih citra yang akan digunakan dan menyimpan citra tersebut ke dalam *database*. *Form* pendataan bias dilihat pada Gambar 3.3.

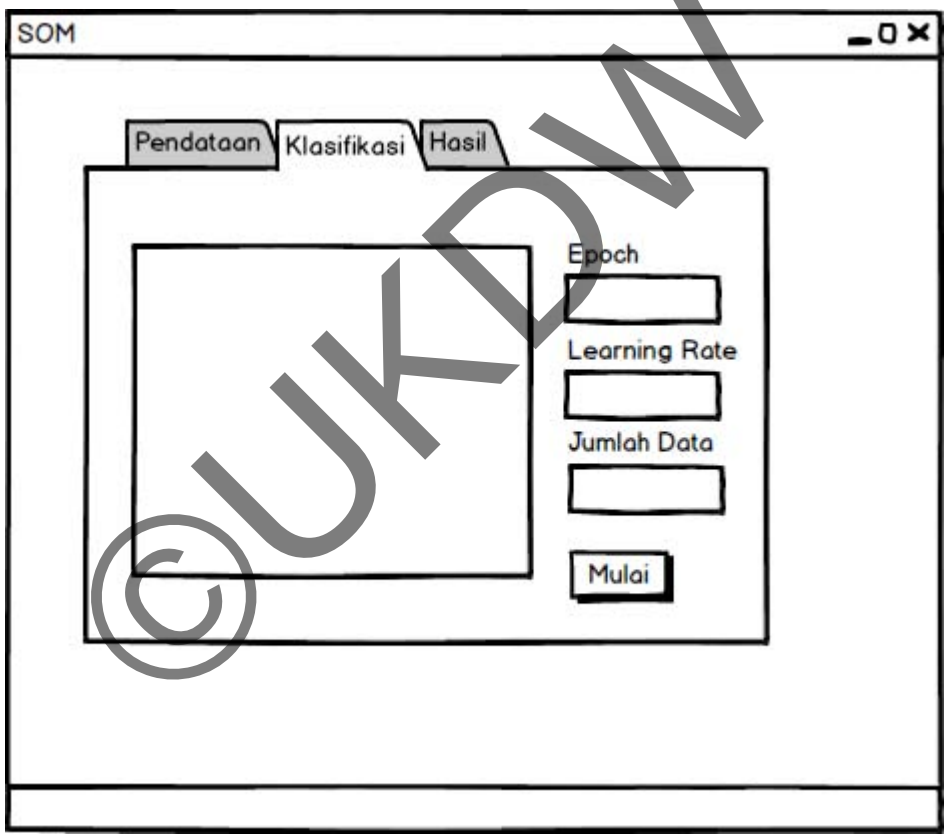


Gambar 3.3 Rancangan *Form* Pendataan

Form pendataan mempunyai 2 buah tombol, 1 buah *edit text* dan 1 buat kanvas seperti pada Gambar 3.3. Tombol pilih file digunakan untuk memilih citra yang akan diproses. Citra yang dipilih akan ditampilkan pada kanvas berukuran 100 x 100. Tombol proses adalah tombol untuk memproses citra yang telah dipilih menjadi *array* dan akan disimpan pada *database*.

3.3.2 *Form* Klasifikasi

Form klasifikasi digunakan untuk melakukan proses klasifikasi citra buah jeruk yang tersimpan di *database* dengan menggunakan metode *Self Organizing Map*. Data-data yang digunakan untuk proses klasifikasi sebelumnya telah dikumpulkan melaui form pendataan. Sebelum memulai proses pelatihan akan dilakukan inisialisasi parameter. Parameter yang wajib diisi adalah *epoch*, *learning rate* dan jumlah data. Proses klasifikasi dimulai dengan menekan tombol mulai. *Form* klasifikasi dapat dilihat pada Gambar 3.4.

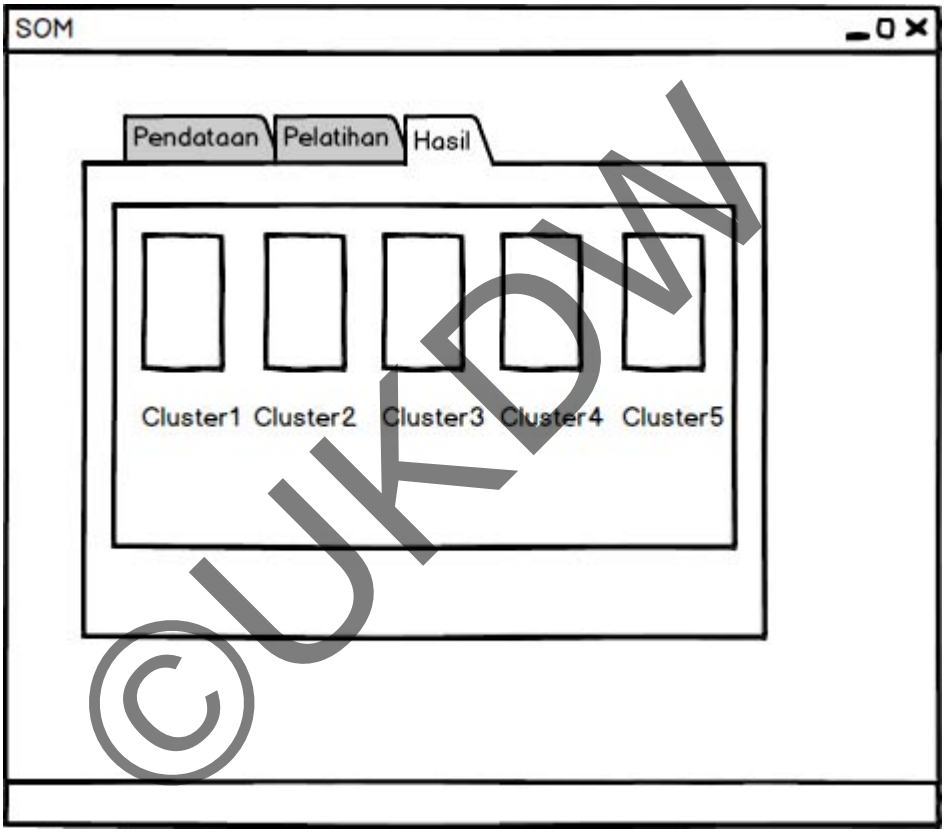


Gambar 3.4 Rancangan *Form* Klasifikasi

Pada Gambar 3.4 terdapat tempat kosong pada bagian kiri *form*. Bagian ini adalah sebuah *datagrid box* yang nantinya menampilkan sebuah tabel dari *database*. Tabel ini memuat informasi dari jeruk yang digunakan sebagai data klasifikasi.

3.3.3 Form Hasil

Form hasil digunakan untuk menampilkan hasil dari klasifikasi yang telah dilakukan sebelumnya. *Form* ini mempunyai lima buah *listbox* yang masing - masing mewakili sebuah *cluster*. Setiap *listbox* berisi nama - nama jeruk yang berkumpul menjadi satu. Selain itu *form* ini juga menampilkan informasi hasil klasifikasi dalam bentuk label. *Form* hasil dapat dilihat pada Gambar 3.5



Gambar 3.5 Rancangan *Form* Hasil

Tidak ada yang bisa dilakukan pengguna selain melihat hasil klasifikasi dalam *form* hasil. Untuk mengulangi proses klasifikasi, pengguna dapat memilih *form* klasifikasi lalu melakukan klasfikasi ulang.

3.4 Rancangan Database

Pada penelitian ini *database* yang digunakan adalah Microsoft Access. *Database* hanya terdiri dari sebuah tabel yang berisi informasi citra yang digunakan dalam klasifikasi. Berikut adalah rancangan untuk tabel jeruk.

Tabel 3.1

Tabel Jeruk

Field Name	Tipe Data
ID	*Autonumber*
Nama	*Text*
Cluster	*Text*
String_Warna	*Memo*

Tabel 3.1 adalah tabel pada *database* yang memuat informasi tentang citra buah jeruk yang digunakan dalam proses klasifikasi. Berikut adalah keterangan untuk setiap *field* tabel jeruk.

1. ID

   ID digunakan sebagai penanda unik setiap data pada tabel ini. Tipe data yang digunakan adalah *autonumber* sehingga pengguna tidak perlu mengisi ID secara manual. Tujuan lain digunakannya *autonumber* supaya tidak terjadi kesamaan ID.

2. Nama

   Nama merupakan nama dari citra yang dipilih oleh pengguna pada *form* pendataan. *Field* nama juga secara otomatis terisi pada saat proses pendataan. Ini dikarenakan sistem mengambil nama citra yang tercantum pada komponen *edit* di *form* pendataan. Tipe data yang digunakan adalah *text* dengan panjang maksimal 15.

3. Cluster

   Merupakan kelompok dimana citra tersebut berada ketika sudah diklasifikasikan. Pada awalnya *field cluster* tidak berisi apa - apa (kosong), kemudian setelah proses klasifikasi *field* ini akan terisi nilai antara 1 sampai dengan 5 sesuai dengan hasil klasifikasi.

4. String_Warna

   Field ini berisi *array* warna *RGB* untuk masing-masing citra. *Array* warna adalah susunan dari warna rata - rata citra seperti yang sudah dijelaskan pada Bab 2.

BAB 4

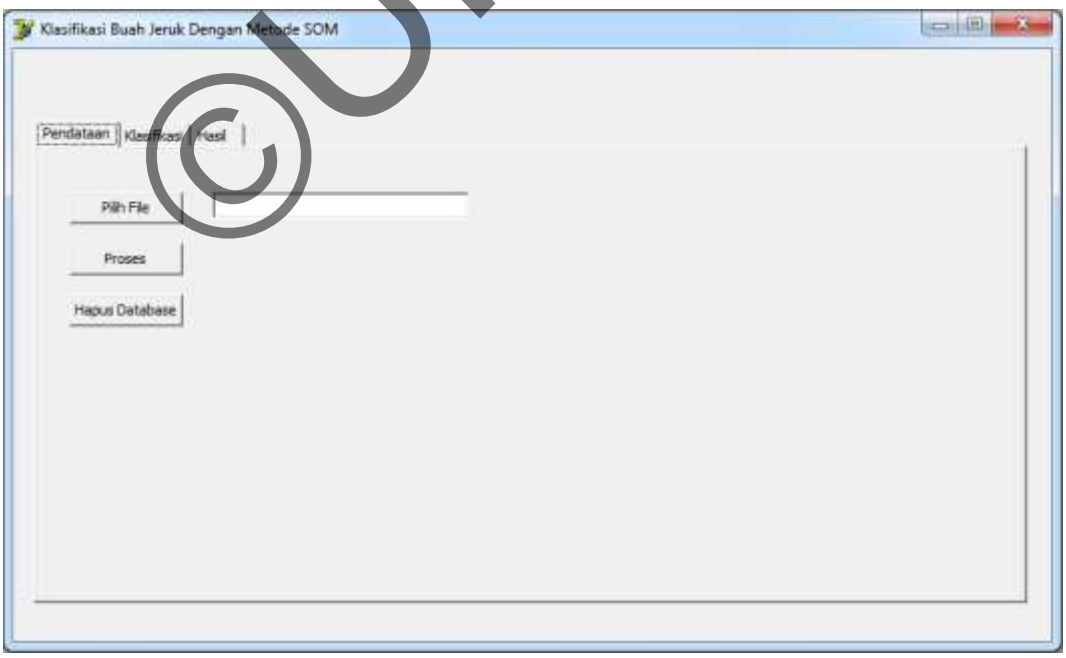
IMPLEMENTASI DAN ANALISIS SISTEM

4.1 Implementasi Sistem

Pada sub bab ini akan dijelaskan mengenai implementasi sistem yang telah dibuat berdasarkan pada rancangan pada Bab 3. Beberapa hal yang dijelaskan adalah penjelasan antar muka sistem, proses pemasukan (*input*) citra, proses klasifikasi dan hasil yang didapatkan dari proses klasifikasi.

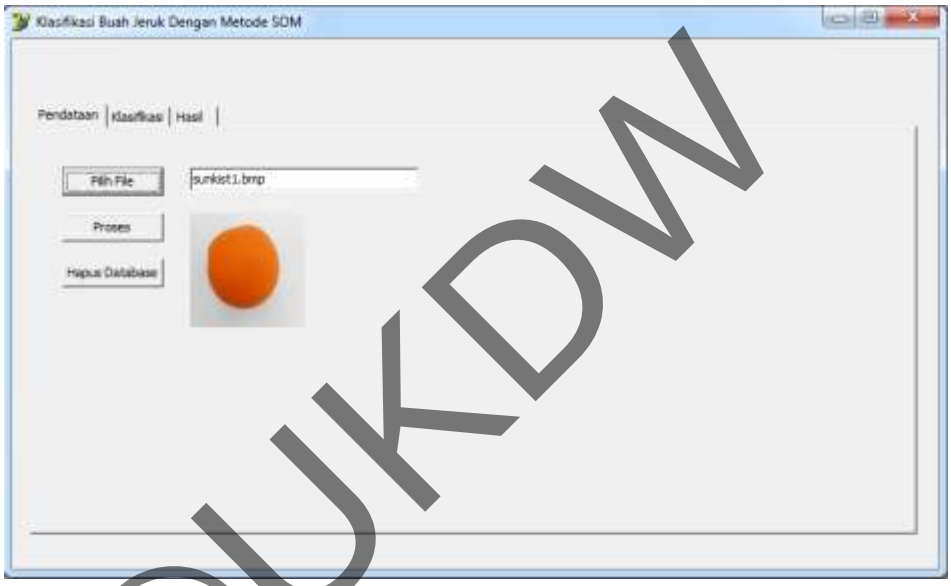
4.1.1 Pendataan Citra

Form pendataan adalah *form* awal pada sistem ini. Form pendataan dapat diliat pada Gambar 4.1.



Gambar 4.1 *Form* Pendataan

Form pendataan secara umum digunakan untuk memasukkan (*input*) citra buah jeruk ke dalam sistem. *Form* pendataaan seperti pada Gambar 4.1 mempunyai dua buah tombol yaitu Pilih *File* dan Proses. Tombol Pilih *File* digunakan untuk memilih citra yang diperlukan dengan format BMP. Bila tombol ini ditekan maka akan muncul sebuah jendela yang berfungsi untuk memilih citra. Setelah citra dipilih maka nama dari citra tersebut akan ditampilkan di *field edit* sedangkan citra itu sendiri akan ditampilkan pada kanvas seperti pada Gambar 4.2. Tombol Hapus *Database* digunakan untuk mengosongkan *database*.

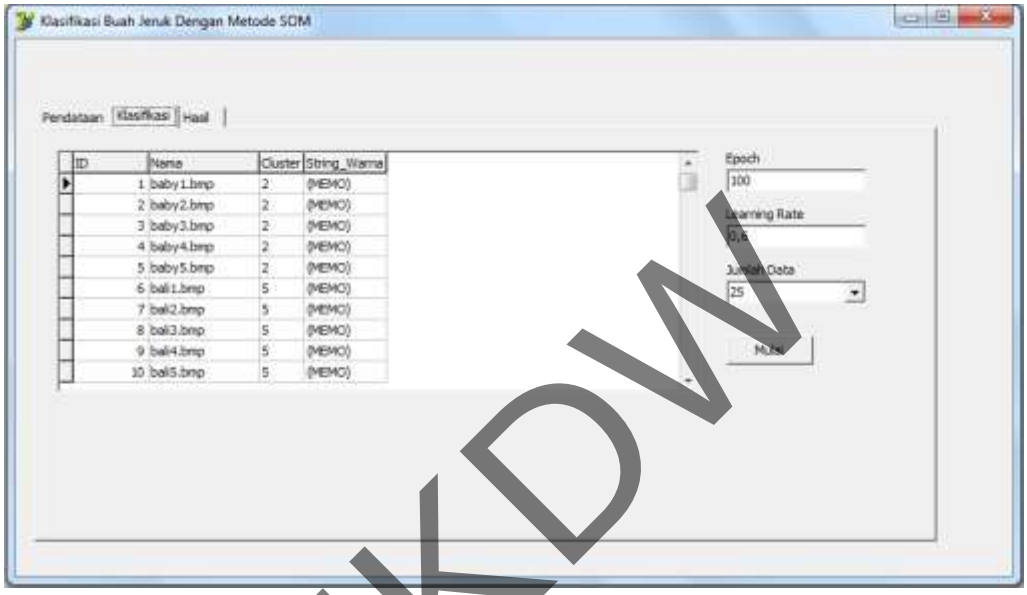


Gambar 4.2 *Form* Pendataan

Tombol proses dapat digunakan bila citra yang diperlukan sudah dipilih dan ditampilan di *form* pendataan. Tombol ini digunakan untuk memproses citra dan sekaligus menyimpan hasilnya ke dalam *database*. Proses yang dilakukan adalah membagi citra ke dalam 100 blok berukuran 10x10 piksel. Tiap blok akan diwakili 1 nilai yang merupakan warna rata-rata dalam blok tersebut. Nilai warna diambil dari nilai RGB sehingga tiap blok mempunyai 3 nilai. Kumpulan nilai dari setiap blok selanjutnya disusun menjadi satu dan disimpan ke dalam *database*. Jadi yang disimpan di dalam *database* adalah *string* warna dari citra bukan citra itu sendiri.

4.1.2 Proses Klasifikasi

Data yang ada di *database* merupakan data yang akan digunakan untuk proses klasifikasi. Proses klasifikasi dilakukan di *form* klasifikasi seperti pada Gambar 4.3.



Gambar 4.3 *Form* Klasifkasi

Form ini mempunyai beberapa komponen yaitu *datagrid*, tiga buah komponen *edit* untuk parameter klasifikasi, dan tombol mulai. Berikut penjelasan masing-masing komponen :

1. *Datagrid*

   Menampilkan tabel jeruk dari *database* yang berisi ID, nama, *cluster*, *string* warna. Nilai cluster yang ditampilkan merupakan hasil dari proses klasifikasi yang dilakukan sebelumnya.

2. Parameter *Epoch*

   Parameter penting dalam klasifikasi, merupakan jumlah iterasi yang diinginkan. Nilai *epoch* dapat diisi sembarang nilai, dengan pertimbangan semakin besar nilai maka semakin lama proses klasifikasi.

3. Parameter *Learning Rate*

   Disebut pula laju pembelajaran, merupakan parameter yang digunakan dalam klasifikasi. Nilai yang bisa digunakan 0,1 sampai 0,9. Setiap nilai akan menghasilkan hasil klasifikasi yang berbeda.

4. Jumlah Data

   Banyaknya data untuk proses klasifikasi. Jumlah yang disediakan adalah 25, 50, 75, dan 100. Secara *default* jumlah akan menunjukan nilai 25. Semakin banyak data maka semakin lama proses klasifikasi.

5. Tombol Mulai

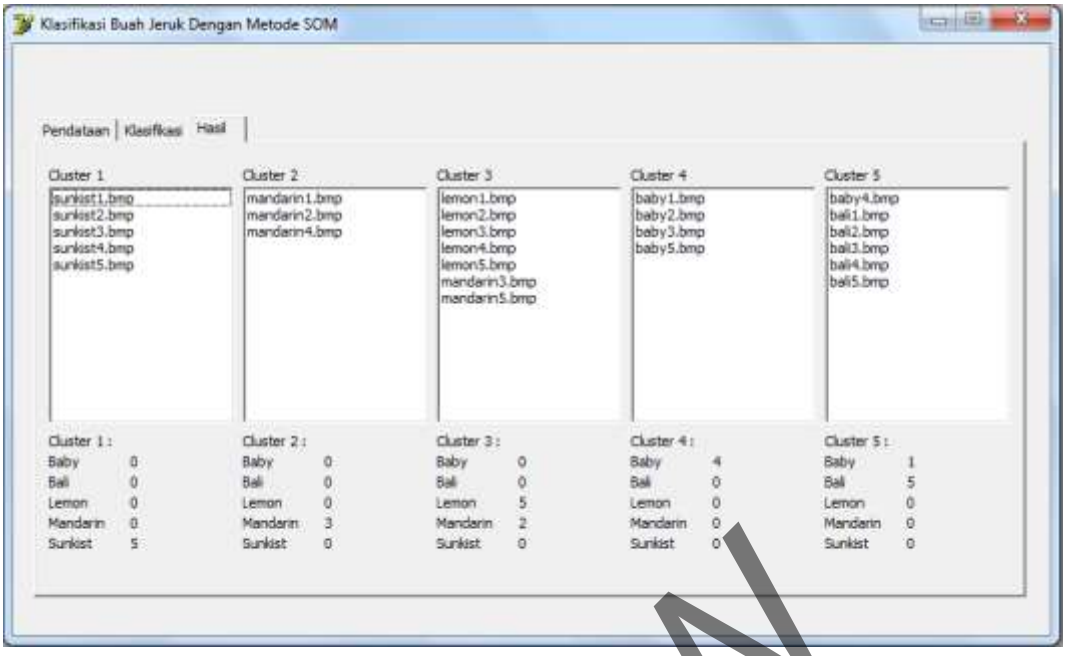
   Memulai proses klasifikasi setelah semua parameter diisi .

Proses klasifikasi dilakukan dengan algoritma *Self Organizing Map (SOM)* dengan menggunakan parameter-parameter yang telah dipilih. Lamanya proses klasifikasi dipengaruhi oleh banyaknya *epoch* dan jumlah data. Semakin banyak keduanya semakin lama proses klasifikasi dilakukan.

4.1.3 Hasil Klasifikasi

Hasil dari klasifikasi akan ditampilkan pada *form* hasil seperti pada Gambar 4.4. Hasil yang ditampilkan berdasarkan parameter-parameter sebelumnya. Hasil yang ditampilkan merupakan hasil dari proses klasifikasi yang terakhir dilakukan.

Form hasil mempunyai lima buah *listbox* yang masing-masing mewakili satu *cluster*. Isi dari setiap *cluster* akan selalu berbeda tergantung pada parameter klasifikasi. Jika klasifikasi dilakukan dengan parameter yang sama dengan sebelumnya, isi dari *cluster* juga akan sama. Hal ini dikarenakan nilai bobot awal tiap *cluster* bukanlah nilai *random* melainkan nilai yang sudah ditentukan.



Gambar 4.4 *Form* Hasil

Di bagian bawah *form* hasil terdapat beberapa label untuk menampilkan informasi tentang hasil klasifikasi. Informasi yang berada di bawah *listbox* setiap *cluster* adalah jumlah jeruk yang tergabung dalam *cluster* tersebut.

4.2 Analisis Sistem

Pada sub bab ini dijelaskan tentang analisis dari program yang dibuat. Analisis akan berdasarkan pada percobaan-percobaan yang telah dilakukan dengan menggunakan program ini.

4.2.1 Analisis Pengaruh Jumlah Epoch Terhadap Jumlah *Cluster*

Secara teori, *epoch* memberikan pengaruh besat terhadap hasil klasifikasi. Semakin besar *epoch* maka semakin lama proses dan semakin akurat hasil yang

didapat. Pada nilai *epoch* tertentu akan dicapai kondisi dimana hasil klasifikasi tetap dan tidak berubah lagi.

Percobaan yang akan dilakukan menggunakan *learning rate* tetap yaitu 0,1 dan jumlah data 25, 50, 75, dan 100. Jumlah iterasi (*epoch*) akan dimulai dari 1 sampai dengan 100. Berikut adalah hasil dari percobaan ini :

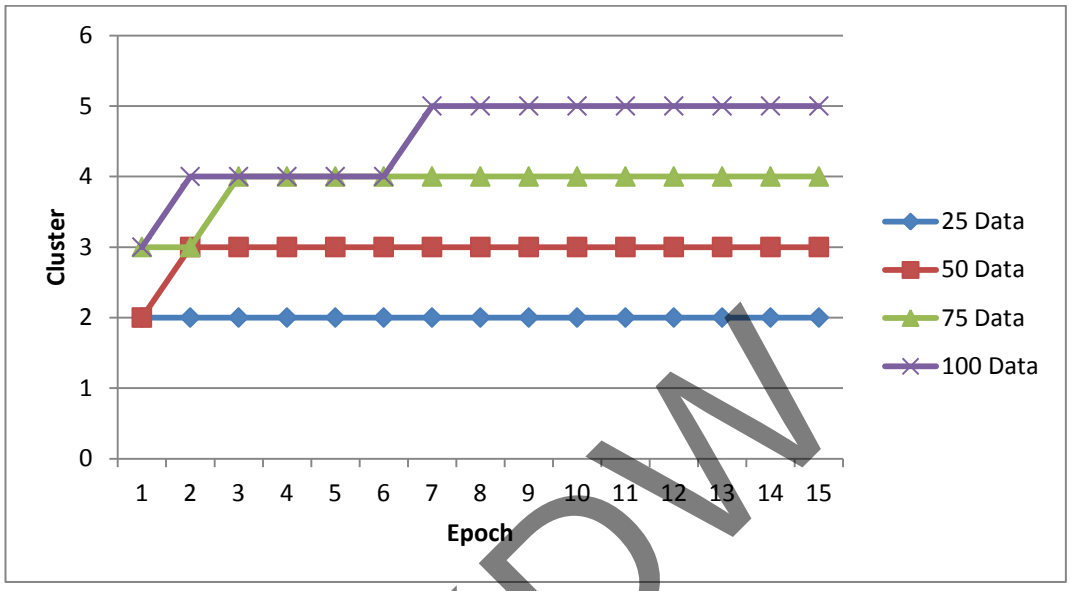
1. Dengan jumlah data 25 dan *epoch* dari 1 sampai dengan 100 selalu menghasilkan jumlah *cluster* yang sama yaitu 2 *cluster*.
2. Dengan jumlah data 50 dan *epoch* 1 menghasilkan jumlah *cluster* sebanyak 2. Sedangkan untuk *epoch* 2 sampai dengan 100 menghasilkan 3 *cluster*.
3. Dengan jumlah data 75 akan menghasilkan jumlah *cluster* sebanyak 4. Untuk *epoch* 1 dan 2 hanya menghasilkan 3 *cluster*.
4. Dengan menggunakan data sebanyak 100 buah, jumlah *cluster* yang dihasilkan bervariasi. Hasil percobaan ini dapat dilihat pada Tabel 4.1.

Tabel 4.1

Pengaruh Jumlah *Epoch* Terhadap Jumlah *Cluster*

No	*Learning Rate*	Jumlah *Epoch*	Jumlah *Cluster*
1	0,1	1	3
2	0,1	2	4
3	0,1	3	4
4	0,1	4	4
5	0,1	5	4
6	0,1	6	4
7	0,1	7	5
8	0,1	8	5
9	0,1	9	5
10	0,1	10	5
11	0,1	50	5
12	0,1	100	5

Gambar 4.5 adalah grafik lengkap hasil klasifikasi percobaan pertama. Dengan menggunakan grafik ini dapat diketahui titik dimana jumlah *epoch* tertentu akan menghasilkan hasil yang stabil.



Gambar 4.5 Grafik Pengaruh *Epoch* Terhadap Jumlah *Cluster*

Dari percobaan yang telah dilakukan membuktikan bahwa banyaknya iterasi (*epoch*) mempengaruhi hasil klasifikasi. Walaupun hal ini juga dipengaruhi oleh banyaknya data klasifikasi. Pada Gambar 4.5 dapat disimpulkan bahwa semakin banyak *epoch* akan meningkatkan peluang setiap *cluster* untuk terisi. Selain itu setiap set data yang dipakai mempunyai titik dimana pada *epoch* tertentu, jumlah *cluster* yang dihasilkan sudah stabil.

4.2.2 Analisis Pengaruh *Learning Rate* Terhadap Jumlah *Cluster*

Menurut teori, nilai *learning rate* (laju pembelajaran) sangat berpengaruh terhadap hasil klasifikasi. Tidak ada patokan berapa nilai *learning rate* yang optimal untuk sebuah penelitian. Jadi setiap penelitian mempunyai nilai *learning rate* optimal yang berbeda.

Percobaan ini bertujuan untuk mencari nilai *learning rate* yang mempunyai hasil terbaik. Dari percobaan sebelumnya sudah didapatkan nilai *epoch* yang stabil untuk percobaan ini. Jumlah iterasi yang digunakan untuk percobaan ini adalah 50. Hasil pengaruh *learning rate* terhadap jumlah *cluster* dapat dilihat pada Tabel 4.2.

Tabel 4.2

Pengaruh *Learning Rate* Terhadap Jumlah *Cluster*

No	*Learning Rate*	Jumlah *Cluster*			
		Data 25	Data 50	Data 75	Data 100
1	0,1	2	3	4	5
2	0,2	3	5	5	5
3	0,3	4	5	5	5
4	0,4	4	4	4	5
5	0,5	4	4	4	4
6	0,6	4	4	4	5
7	0,7	4	4	4	5
8	0,8	5	5	5	5
9	0,9	4	4	4	5

Dari tabel 4.2 dapat disimpulkan beberapa hal yaitu :

1. *Learning rate* 0,8 merupakan nilai yang paling baik karena berapapun jumlah data klasifikasinya, *cluster* yang dihasilkan selalu 5 buah.
2. *Learning rate* 0,5 merupakan nilai tidak dapat digunakan di percobaan berikutnya karena sama sekali tidak menghasilkan 5 *cluster*.
3. Semakin banyak data klasifikasi maka semakin besar peluang setiap *cluster* akan terisi. Hal ini bila dilihat pada kolom Data 100 hampir semua menghasilkan 5 *cluster*. Pernyataan ini juga berlaku pada jumlah data 50 dan 75. Kedua set data tersebut mempunyai hasil yang lebih baik daripada jumlah data 25.

4.2.3 Analisis Pasangan *Learning Rate* dan Banyak Data Terbaik

Pada percobaan ini akan menggunakan nilai *learning rate* dan jumlah data yang telah didapatkan pada percobaan sebelumnya. Pasangan *learning rate* dan jumlah data yang dipakai hanya yang menghasilkan 5 *cluster* di percobaan sebelumnya.

Tabel 4.3

Persentase Ketepatan Klasifikasi

No	*Learning Rate*	Jumlah Data	*Cluster* 1 (%)	*Cluster* 2 (%)	*Cluster* 3 (%)	*Cluster* 4 (%)	*Cluster* 5 (%)	Rata-Rata
1	0,1	100	100	62,96	54,16	62,5	52,63	66,45
2	0,2	50	100	57,14	69,23	40	52,94	63,86
3	0,2	75	48,38	78,5	53,84	100	100	77,94
4	0,2	100	45,23	80	65,21	100	100	78,09
5	0,3	50	58,82	75	100	100	66,66	80,09
6	0,3	75	83,33	88,88	53,84	100	100	85,21
7	0,3	100	45,23	80	65,21	100	100	78,09
8	0,4	100	83,33	83,33	51,42	100	100	83,61
9	0,6	100	45,23	80	65,21	100	100	78,09
10	0,7	100	100	100	65,21	80	45,23	78,09
11	0,8	25	100	100	71,42	100	83,33	90,95
12	0,8	50	76,92	66,66	50	100	83,33	75,38
13	0,8	75	100	100	53,84	88,88	83,33	85,21
14	0,8	100	100	100	65,21	80	45,23	78,09
15	0,9	100	45,23	80	65,21	100	100	78,09

Berikut adalah penjelasan mengenai Tabel 4.3:

1. Persentase nilai pada kolom *cluster* pada tabel di atas didapatkan dari perhitungan jumlah jenis jeruk terbanyak dibagi dengan total jeruk dan dikalikan 100 persen.
2. Beberapa hasil klasifikasi mempunyai 2 *cluster* atau lebih yang mempunyai mayoritas jenis buah yang sama.
3. Beberapa cluster juga mempunyai 2 jenis buah jeruk yang berjumlah sama.
4. Persentase di bawah 50% artinya dalam *cluster* tersebut terdiri lebih dari 2 jenis buah jeruk.
5. Untuk cluster yang mempunyai 2 jenis buah jeruk berjumlah sama dan juga menghasilkan persentase di bawah 50% diberi *highlight* warna hijau.
6. Untuk hasil klasifikasi yang setiap jenis buah jeruk hanya diwakili oleh 1 *cluster* diberi *highlight* warna kuning.

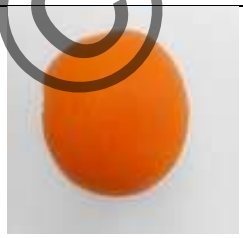
Harapan awal dari percobaan ini adalah setiap jenis buah jeruk akan diwakili oleh 1 *cluster* saja. Namun tidak semua pasangan *learning rate* dan jumlah data yang diujikan menghasilkan hasil sesuai harapan. Hanya ada 6 pasangan yang memenuhi harapan dalam percobaan kali ini. Persentase keenam pasangan tersebut ketepatannya di atas 80% dengan persentase tertinggi adalah 90,95% untuk *learning rate* 0,8 dan jumlah data 25.

4.2.4 Analisis Kecenderungan Pengelompokan Buah Jeruk

Pada sub ini akan dibahas mengenai kecenderungan pengelompokan antar jenis buah jeruk Data yang digunakan merupakan hasil dari percobaan sebelumnya, jadi pada pembahasan ini tidak akan dilakukan percobaan lain. Percobaan yang dilakukan sebelumnya sebanyak 15 kali. Jadi jumlah *cluster* total yang digunakan dalam pembahasan sebanyak 75.

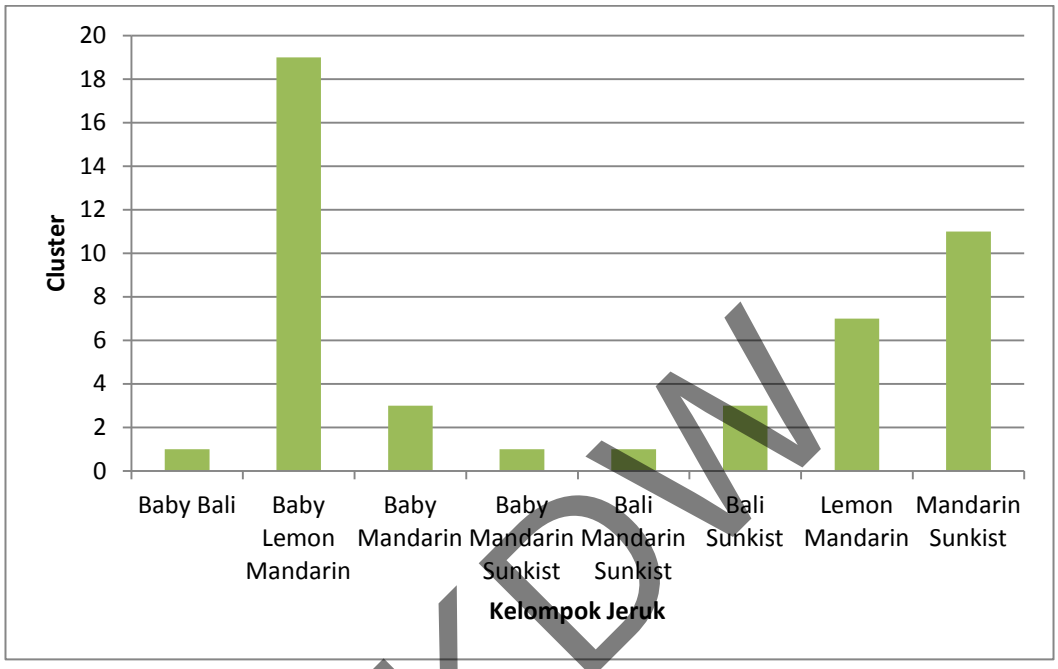
Tabel 4.4

Jumlah Cluster Yang Berisikan Satu Jenis Buah Jeruk

No	Contoh Gambar	Jenis Buah	Jumlah
1		Baby	6
2		Bali	12
3		Lemon	2
4		Mandarin	1
5		Sunkist	8

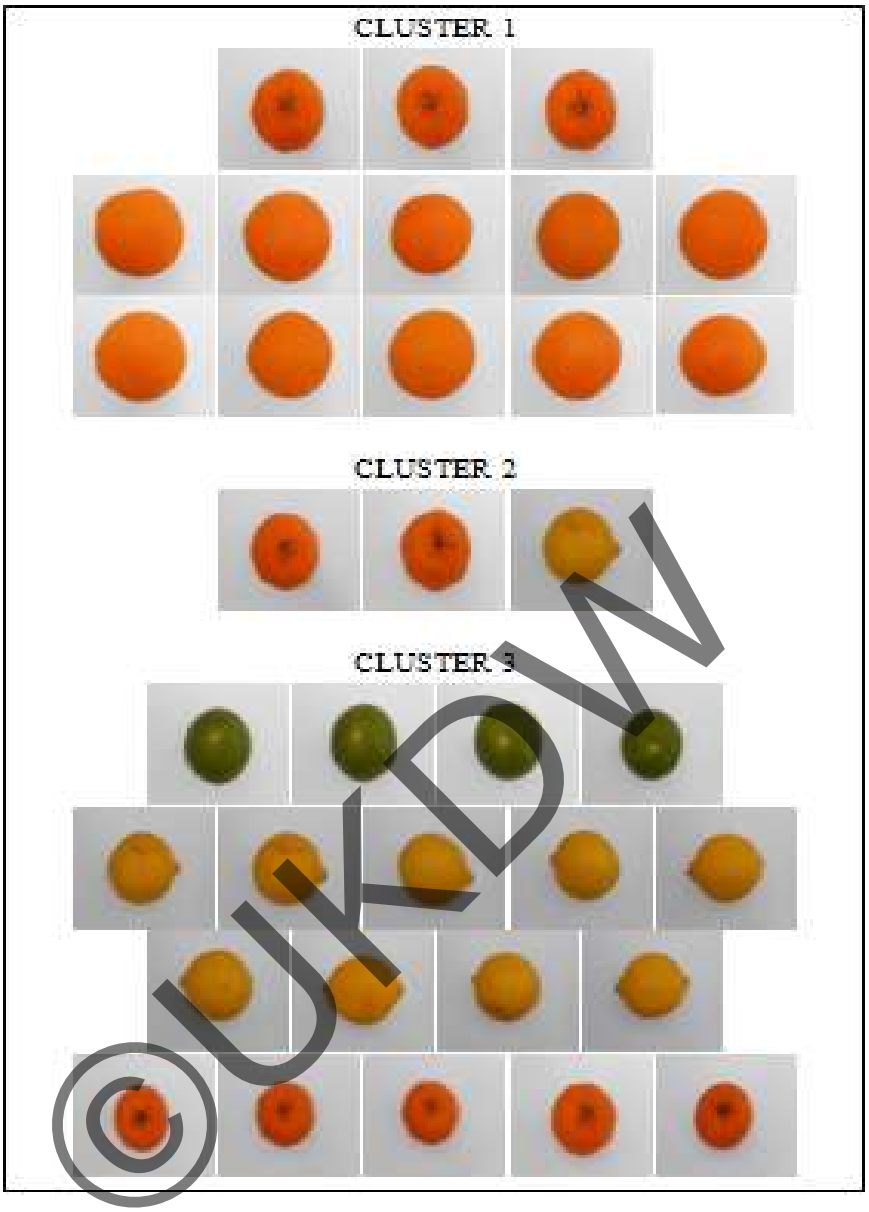
Tabel 4.4 memuat tentang banyaknya cluster yang hanya berisi satu jenis buah jeruk. Jeruk Bali merupakan jenis jeruk yang sering berkelompok dengan jenisnya sendiri. Posisi berikutnya adalah jeruk Baby dan jeruk Sunkist. Alasan jeruk Bali mampu mengelompokkan diri dengan baik kemungkinan karena memiliki warna dan ukuran yang sangat berbeda dengan jeruk lain.

Selanjutnya adalah tentang kecenderugan jenis jeruk tertentu untuk mengelompokkan diri dengan jenis jeruk yang berbeda. Hal ini dapat dilihat pada Gambar 4.6.



Gambar 4.6 Kelompok Jenis Buah Jeruk

Dari Gambar 4.6 diketahui bahwa kumpulan jeruk yang sering tergabung dalam satu *cluster* adalah jeruk Baby, jeruk Lemon dan jeruk Mandarin. Bahkan menurut data yang dimiliki penulis, ketiga jeruk tersebut juga pernah berkelompok dalam dua *cluster* yang berbeda untuk parameter yang sama. Analisis penyebab beberapa jenis jeruk sering berkumpul dengan jenis yang berbeda akan menggunakan hasil klasifikasi dengan *learning rate* 0,8 dan jumlah data 50. Parameter ini dipilih karena hasil klasifikasinya menghasilkan *cluster* jenis buah jeruk yang sering berkumpul jadi satu.



Gambar 4.7 Contoh Hasil Klasifikasi

Gambar 4.7 merupakan hasil klasifikasi dengan menggunakan *learning rate* 0,8 dan jumlah data 50. Hasil yang ditampilkan hanya *cluster* yang memiliki jenis buah jeruk bervariasi. *Cluster* 1 terdiri dari jeruk Mandarin dan Sunkist. Menurut penulis penyebab kedua jenis buah jeruk ini berkumpul karena mempunyai warna kulit yang mirip.

Cluster 2 terdiri dari jeruk Mandarin dan jeruk Lemon sedangkan pada cluster 3 terdiri dari jeruk Baby, jeruk Lemon dan jeruk Mandarin. Pada kasus ini

kemiripan warna kulit bukan menjadi faktor yang membuat jenis-jenis jeruk tersebut berkumpul. Jadi kemungkinan lain yang menyebabkan jenis-jenis jeruk ini berkumpul adalah kesamaan warna latar belakang, ukuran buah jeruk di dalam citra atau letak buah jeruk di dalam citra.

4.3 Kelebihan dan Kekurangan Sistem

4.3.1 Kelebihan Sistem

1. Sistem membebaskan pengguna menentukan parameter *epoch*, *learning rate* dan jumlah data.
2. Sistem menampilkan hasil klasifikasi dengan baik.

4.3.2 Kekurangan Sistem

1. Sistem hanya dapat menggunakan data *input* yang berasal dari *database*.
2. Karena sistem hanya mengambil warna sebagai ciri untuk proses klasifikasi, maka pengambilan citra harus konsisten. Bila tidak konsisten akan menyebabkan hasil tidak sesuai dengan harapan.

LAMPIRAN

Proses Pendataan

```
procedure TForm1.btnPilihClick(Sender: TObject);
begin
  OpenPictureDialog1.Title :='Pilih Gambar'; //memberi judul pada open
dialog

  OpenPictureDialog1.Execute;
  Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);

  editPendataan.Text:=ExtractFileName(OpenPictureDialog1.FileName);
end;


procedure TForm1.btnProsesClick(Sender: TObject);
var
  Color : TColor;
  R, G, B : integer;        //nilai dari suatu pixel
  X, Y : integer;           //untuk penanda koordinat di kanvas
  ZX, ZY  : integer;        //ZX = 10pixel blok mendatar, ZY = 10 pixel
blok menurun
  I, J : integer;           //untuk perulangan
  index : integer;          //untuk index array
  tampR, tampG, tampB : integer;      //tampungan sementara sebelum di
rata2
  rataR, rataG, rataB : integer;
  arrayWarna : array[1..300] of integer;
  stringWarna : string;

begin
  ZX:=0;
  ZY:=0;
  tampR:=0;
  tampG:=0;
  tampB:=0;
  index:=1;
  stringWarna:='';
```

```
if Assigned(Image1.Picture.Graphic) then
begin
if length(editPendataan.Text)<=15 then //cek panjang nama
begin

////proses pembagian blok////
  for J:=0 to 9 do  //ke bawah
  begin
    for I:=0 to 9 do  //ke kanan
    begin
      for Y:=ZY to ZY+9 do
      begin
        for X:=ZX to ZX+9 do
        begin
          Color := Image1.Canvas.Pixels[X, Y];
          R := GetRValue(Color);
          G := GetGValue(Color);
          B := GetBValue(Color);
          tampR:=tampR+R;
          tampG:=tampG+G;
          tampB:=tampB+B;
        end;
      end;
      if ZX<90 then
        ZX:=ZX+10
      else
        ZX:=0;
      rataR:=round(tampR/100);
      rataG:=round(tampG/100);
      rataB:=round(tampB/100);
      arrayWarna[index]:=rataR;
      index:=index+1;
      arrayWarna[index]:=rataG;
      index:=index+1;
      arrayWarna[index]:=rataB;
      index:=index+1;               //index terakhir jadi 301
      tampR:=0;
      tampG:=0;
      tampB:=0;
    end;
    ZY:=ZY+10;
```

```
  end;

////proses membuat string Warna////
  for i:=1 to 300 do
  begin
    if length(IntToStr(arrayWarna[i]))=1 then
      stringWarna:=stringWarna+IntToStr(arrayWarna[i])+'  '
    else
    if length(IntToStr(arrayWarna[i]))=2 then
      stringWarna:=stringWarna+IntToStr(arrayWarna[i])+' '
    else
      stringWarna:=stringWarna+IntToStr(arrayWarna[i]);
  end;

////proses simpan ke database////
  ADOTable1.Append;
  ADOTable1.FieldByName('nama').AsString:=editPendataan.Text;
  ADOTable1.FieldByName('string_warna').AsString:=stringWarna;
  ADOTable1.Post;

////aktivasi query supaya tabel juga berubah////
  ADOQuery1.SQL.Text:='select * from TableJeruk';
  ADOQuery1.Active:=true;
  ShowMessage('Berhasil');

end
else
  ShowMessage('Panjang Maksimal Nama 15')
end
else
  ShowMessage('Pilih Citra Dahulu');

end;
```

Proses Klasifikasi

```
procedure TForm1.btnMulaiClick(Sender: TObject);
var
  epoch : integer;
  learningRate : double;
  r,i,j,k,l : integer;
  bobot : array[1..5, 1..300] of double;
```

```
  input : array[1..100, 1..300] of integer;    //100 data panjangnya 300
  stringInput : array[1..100] of string;       //string warna dari db
  tampungString : string;   //tampungan copy string
  x : integer;    //penanda copy string
  tampungBobot : double;    //tampungan bobot
  jarak : array[1..5] of double;   //jarak / D[j]
  tampungJarak : double;    //tampungan jarak
  indekTerkecil : integer;
  cluster : array[1..100] of integer;
  region : integer;
  combo, jumlahData : integer;
  tampungNama : string;   //tampungan nama jeruk cluster listbox
  tampungMandarin, tampungLemon, tampungBali, tampungBaby, tampungSunkist
: integer;
  jmlMandarin :  array[1..5] of integer;
  jmlLemon : array[1..5] of integer;
  jmlBali : array[1..5] of integer;
  jmlBaby : array[1..5] of integer;
  jmlSunkist : array[1..5] of integer;

begin
  x:=1;
  region:=1;
  epoch:=StrToInt(editEpoch.Text);
  learningRate:=StrToFloat(editLearning.Text);
  indekTerkecil:=1;
  combo:=ComboBox1.ItemIndex;
  jumlahData:=25;
  if combo=0 then
    jumlahData:=25
  else
  if combo=1 then
    jumlahData:=50
  else
  if combo=2 then
    jumlahData:=75
  else
  if combo=3 then
    jumlahData:=100;


if (learningRate>=0.1) and (learningRate<1) then //cek learning rate
```

```
begin

////proses tampung stringWarnaInput////
  for i:=1 to jumlahData do
  begin
    ADOQuery1.SQL.Text:='select * from tablejeruk where id='+IntToStr(i);
    ADOQuery1.Active:=true;
    stringInput[i]:=ADOQuery1.FieldValues['string_warna'];
  end;

////proses konver string_warna ke array////
  for i:=1 to jumlahData do     //jumlah data 25-50-75-100
  begin
    for j:=1 to 300 do
    begin
      tampungString:=Copy(stringInput[i],x,3);
      input[i][j]:=StrToInt(Trim(tampungString));
      x:=x+3;
    end;
    x:=1;
  end;

////inisialisasi bobot////
  for i:=1 to 5 do
  begin
    for j:=1 to 300 do
    begin
      bobot[i][j]:=i*0.15;
    end;
  end;

////inisialisasi jarak////
  for k:=1 to 5 do
  begin
    jarak[k]:=0;
  end;

////proses klasifikasi////
  for r:=1 to 2 do
  begin
    for i:=1 to epoch do
    begin
```

```
      for j:=1 to jumlahData do     //setiap vektor input lakukan
      begin
        //cari jarak tiap cluster
        for k:=1 to 5 do    //banyak cluster = 5
        begin
          for l:=1 to 300 do    //panjang arraywarna
          begin
            tampungBobot:=sqr(bobot[k][l]-input[j][l]);
            jarak[k]:=jarak[k]+tampungBobot;
          end;
        end;
        //cari jarak terpendek
        tampungJarak:=jarak[1];
        for k:=2 to 5 do
        begin
          if jarak[k] < tampungJarak then
          begin
            tampungJarak:=jarak[k];
            indekTerkecil:=k;
          end
          else if jarak[1]=tampungJarak then
            indekTerkecil:=1;
        end;
        //reset jarak ke 0
        for k:=1 to 5 do
        begin
          jarak[k]:=0;
        end;
        //update cluster untuk input J
        cluster[j]:=indekTerkecil;
        //update bobot baru cluster terdekat
        if region=0 then
        begin
          for k:=1 to 300 do
          begin
            bobot[indekTerkecil][k]:=((1-
learningRate)*bobot[indekTerkecil][k])+(learningRate*input[j][k]);
          end;
        end
        else
        if region=1 then
        begin
```

```
          if indekTerkecil=1 then
          begin
            for k:=1 to 300 do
            begin
              bobot[indekTerkecil][k]:=((1-
learningRate)*bobot[indekTerkecil][k])+(learningRate*input[j][k]);
              bobot[indekTerkecil+1][k]:=((1-
learningRate)*bobot[indekTerkecil+1][k])+(learningRate*input[j][k]);
            end;
          end
          else
          if indekTerkecil=5 then
          begin
            for k:=1 to 300 do
            begin
              bobot[indekTerkecil][k]:=((1-
learningRate)*bobot[indekTerkecil][k])+(learningRate*input[j][k]);
              bobot[indekTerkecil-1][k]:=((1-
learningRate)*bobot[indekTerkecil-1][k])+(learningRate*input[j][k]);
            end;
          end
          else
          begin
            for k:=1 to 300 do
            begin
              bobot[indekTerkecil][k]:=((1-
learningRate)*bobot[indekTerkecil][k])+(learningRate*input[j][k]);
              bobot[indekTerkecil+1][k]:=((1-
learningRate)*bobot[indekTerkecil+1][k])+(learningRate*input[j][k]);
              bobot[indekTerkecil-1][k]:=((1-
learningRate)*bobot[indekTerkecil-1][k])+(learningRate*input[j][k]);
            end;
          end;
        end;
      end;
      //update learning rate
      learningRate:=learningRate*0.5;
    end;
    region:=region-1;
  end;

//update cluster
```

```
  for i:=1 to jumlahData do
  begin
    ADOQuery1.SQL.Text:='select * from TableJeruk where ID='+IntToStr(i);
    ADOQuery1.Active:=true;
    ADOQuery1.Edit;
    ADOQuery1.Fields.FieldByName('cluster').Text:=IntToStr(cluster[i]);
    ADOQuery1.Post;
  end;

//inisialiasi jumlah jeruk
  for i:=1 to 5 do
  begin
    jmlMandarin[i]:=0;
    jmlLemon[i]:=0;
    jmlBali[i]:=0;
    jmlBaby[i]:=0;
    jmlSunkist[i]:=0;
  end;

////hasil klasifikasi ke memo////
  ListBox1.Clear;
  ListBox2.Clear;
  ListBox3.Clear;
  ListBox4.Clear;
  ListBox5.Clear;
  for i:=1 to jumlahData do
  begin
    //inisialisasi
    tampungMandarin:=0;
    tampungLemon:=0;
    tampungBali:=0;
    tampungBaby:=0;
    tampungSunkist:=0;
    ADOQuery1.SQL.Text:='select    nama    from    tablejeruk    where
id='+IntToStr(i);
    ADOQuery1.Active:=true;
    tampungNama:=ADOQuery1.FieldValues['nama'];
    //tampung sementara untuk penjumlahan
    if AnsiContainsText(tampungNama, 'mandarin') then
    begin
      tampungMandarin:=1;
    end
```

```
else
if AnsiContainsText(tampungNama, 'lemon') then
begin
  tampungLemon:=1;
end
else
if AnsiContainsText(tampungNama, 'bali') then
begin
  tampungBali:=1;
end
else
if AnsiContainsText(tampungNama, 'baby') then
begin
  tampungBaby:=1;
end
else
if AnsiContainsText(tampungNama, 'sunkist') then
begin
  tampungSunkist:=1;
end;
//isi listbox dan jumlah jeruk
if cluster[i]=1 then
begin
  ListBox1.Items.Add(tampungNama);
  jmlMandarin[1]:=jmlMandarin[1]+tampungMandarin;
  jmlLemon[1]:=jmlLemon[1]+tampungLemon;
  jmlBali[1]:=jmlBali[1]+tampungBali;
  jmlBaby[1]:=jmlBaby[1]+tampungBaby;
  jmlSunkist[1]:=jmlSunkist[1]+tampungSunkist;
end
else
if cluster[i]=2 then
begin
  ListBox2.Items.Add(tampungNama);
  jmlMandarin[2]:=jmlMandarin[2]+tampungMandarin;
  jmlLemon[2]:=jmlLemon[2]+tampungLemon;
  jmlBali[2]:=jmlBali[2]+tampungBali;
  jmlBaby[2]:=jmlBaby[2]+tampungBaby;
  jmlSunkist[2]:=jmlSunkist[2]+tampungSunkist;
end
else
if cluster[i]=3 then
```

```
    begin
      ListBox3.Items.Add(tampungNama);
      jmlMandarin[3]:=jmlMandarin[3]+tampungMandarin;
      jmlLemon[3]:=jmlLemon[3]+tampungLemon;
      jmlBali[3]:=jmlBali[3]+tampungBali;
      jmlBaby[3]:=jmlBaby[3]+tampungBaby;
      jmlSunkist[3]:=jmlSunkist[3]+tampungSunkist;
    end
    else
    if cluster[i]=4 then
    begin
      ListBox4.Items.Add(tampungNama);
      jmlMandarin[4]:=jmlMandarin[4]+tampungMandarin;
      jmlLemon[4]:=jmlLemon[4]+tampungLemon;
      jmlBali[4]:=jmlBali[4]+tampungBali;
      jmlBaby[4]:=jmlBaby[4]+tampungBaby;
      jmlSunkist[4]:=jmlSunkist[4]+tampungSunkist;
    end
    else
    if cluster[i]=5 then
    begin
      ListBox5.Items.Add(tampungNama);
      jmlMandarin[5]:=jmlMandarin[5]+tampungMandarin;
      jmlLemon[5]:=jmlLemon[5]+tampungLemon;
      jmlBali[5]:=jmlBali[5]+tampungBali;
      jmlBaby[5]:=jmlBaby[5]+tampungBaby;
      jmlSunkist[5]:=jmlSunkist[5]+tampungSunkist;
    end;
  end;

//caption label
  lblMandarin1.Caption:=IntToStr(jmlMandarin[1]);
  lblMandarin2.Caption:=IntToStr(jmlMandarin[2]);
  lblMandarin3.Caption:=IntToStr(jmlMandarin[3]);
  lblMandarin4.Caption:=IntToStr(jmlMandarin[4]);
  lblMandarin5.Caption:=IntToStr(jmlMandarin[5]);
  lblLemon1.Caption:=IntToStr(jmlLemon[1]);
  lblLemon2.Caption:=IntToStr(jmlLemon[2]);
  lblLemon3.Caption:=IntToStr(jmlLemon[3]);
  lblLemon4.Caption:=IntToStr(jmlLemon[4]);
  lblLemon5.Caption:=IntToStr(jmlLemon[5]);
  lblBali1.Caption:=IntToStr(jmlBali[1]);
```

```
  lblBali2.Caption:=IntToStr(jmlBali[2]);
  lblBali3.Caption:=IntToStr(jmlBali[3]);
  lblBali4.Caption:=IntToStr(jmlBali[4]);
  lblBali5.Caption:=IntToStr(jmlBali[5]);
  lblBaby1.Caption:=IntToStr(jmlBaby[1]);
  lblBaby2.Caption:=IntToStr(jmlBaby[2]);
  lblBaby3.Caption:=IntToStr(jmlBaby[3]);
  lblBaby4.Caption:=IntToStr(jmlBaby[4]);
  lblBaby5.Caption:=IntToStr(jmlBaby[5]);
  lblSunkist1.Caption:=IntToStr(jmlSunkist[1]);
  lblSunkist2.Caption:=IntToStr(jmlSunkist[2]);
  lblSunkist3.Caption:=IntToStr(jmlSunkist[3]);
  lblSunkist4.Caption:=IntToStr(jmlSunkist[4]);
  lblSunkist5.Caption:=IntToStr(jmlSunkist[5]);

//dipakai untuk mereset sql ADOquery1, karena diatas sudah dirubah
  ADOQuery1.SQL.Text:='select * from TableJeruk';
  ADOQuery1.Active:=true;

end
else
  ShowMessage('Learning Rate Harus 0.1 - 0.9');

end;
```

Proses Hapus Database

```
procedure TForm1.btnHapusClick(Sender: TObject);
var
  buttonselected : integer;

begin
  buttonselected:=MessageDlg('Hapus
Database',mtConfirmation,mbOKCancel,0);
  if buttonselected=mrOk then
  begin
    ADOConnection1.Execute('DELETE * FROM tablejeruk');
    ADOConnection1.Execute('ALTER  TABLE  tablejeruk  ALTER  COLUMN  ID
Autoincrement(1,1)');
    ADOQuery1.SQL.Text:='select * from tablejeruk';
    ADOQuery1.Active:=true;
```

```
    ShowMessage('Database Dihapus');
  end
  else
  if buttonselected=mrCancel then
    ShowMessage('Batal Dihapus');
end;
```