TUGAS AKHIR – SM141501

IDENTIFIKASI JENIS KANKER DARAH (LEUKEMIA) TERHADAP PENGARUH PARAMETER KERNEL *SUPPORT VECTOR MACHINE* DAN EKSTRAKSI CIRI DENGAN RANTAI MARKOV ORDE 2

MOH. HAMIM ZAJULI AL FAROBY
NRP. 06111340000030

Dosen Pembimbing
Prof. Dr. Mohammad Isa Irawan, MT
NIP : 19631225 198903 1 001

DEPARTEMEN MATEMATIKA
Fakultas Matematika Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

FINAL PROJECT – SM141501

IDENTIFICATION OF BLOOD CANCER (LEUKEMIA) TYPES ON THE INFLUENCE OF KERNEL SUPPORT VECTOR MACHINE PARAMETERS AND LEAVING EXTRACTS WITH MARKOV CHAIN ORDE 2

MOH. HAMIM ZAJULI AL FAROBY
NRP. 06111340000030

Supervisor
Prof. Dr. Mohammad Isa Irawan, MT
NIP : 19631225 198903 1 001

DEPARTEMENT OF MATHEMATICS
Fakultas Matematika Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

IDENTIFIKASI JENIS KANKER DARAH (LEUKEMIA) TERHADAP PENGARUH PARAMETER KERNEL *SUPPORT VECTOR MACHINE* DAN EKSTRAKSI CIRI DENGAN RANTAI MARKOV ORDE 2

IDENTIFICATION OF BLOOD CANCER (LEUKEMIA) TYPES ON THE INFLUENCE OF KERNEL SUPPORT VECTOR MACHINE PARAMETERS AND LEAVING EXTRACTS WITH MARKOV CHAIN ORDE 2

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Sains
Pada bidang minat Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Matematika Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
MOH. HAMIM ZAJULI AL FAROBY

Menyetujui,
Dosen Pembimbing,

Prof. Dr. Mohammad Isa Irawan, MT
NIP. 19631225 198903 1 001

Mengetahui,
Kepala Departemen Matematika
FMKSD ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Halaman ini sengaja dikosongkan

IDENTIFIKASI JENIS KANKER DARAH (LEUKEMIA) TERHADAP PENGARUH PARAMETER KERNEL *SUPPORT VECTOR MACHINE* DAN EKSTRAKSI CIRI DENGAN RANTAI MARKOV ORDE 2

Nama Mahasiswa	: Moh. Hamim Zajuli Al Faroby
NRP	: 06111340000030
Departemen	: Matematika
Dosen Pembimbing	: Prof. Mohammad Isa Irawan, MT

ABSTRAK

Penelitian terhadap sekuens DNA/RNA leukemia dapat mempermudah dan mempercepat bagi tenaga medis dalam mendiagnosa jenis leukemia. Data leukemia diolah dengan mengekstrasi ciri data sekuens berupa tipe string ke bentuk numerik dengan menggunakan metode rantai Markov orde 2. Hasil dari ekstraksi ciri yaitu 64 ciri dari DNA/RNA leukemia dalam bentuk matriks Markovian. *Machine Learning* digunakan dalam proses klasifikasi dengan pendekatan metode *Support Vector Machine* (SVM). Pada saat mengklasifikasikan, penelitian ini menggunakan 40 data *training* sebagai pengklasifikasi data dan 25 data tes. SVM membagi kelas dari DNA/RNA kanker darah menjadi empat bagian yaitu *Acute Myeloid Leukemia, Acute Lymphocytic Leukemia, Chronic Myelogenous Leukemia* dan *Chronic Lymphocytic Leukemia*. Parameter-parameter pada SVM dilakukan uji coba untuk mendapaktan nilai optimal dimana SVM dapat bekerja dengan baik pada data leukemia. Hasil dari penelitian berupa gambar data latih pada SVM. Gambar berupa grafik 2D dimana 64 ciri dari DNA/RNA leukemia akan dikompres dengan menggunakan Principle Component Analysis (PCA). Hasil penelitian yang lain berupa prediksi yang divalidasi dengan percobaan terhadap data test. Serta dibandingkan nya hasil dari tiga kernel yaitu linear, gaussian dan polynomial.

Percobaan tersebut digunakan untuk mengetahui performa kernel mana yang paling bagus digunakan pada data sekuens DNA/RNA leukemia.

Kata Kunci : Rantai Markov, *Machine Learning, Support Vector Machine,* Kanker Darah.

IDENTIFICATION OF BLOOD CANCER (LEUKEMIA) TYPES ON THE INFLUENCE OF KERNEL SUPPORT VECTOR MACHINE PARAMETERS AND LEAVING EXTRACTS WITH MARKOV CHAIN ORDE 2

Name of Student	: Moh. Hamim Zajuli Al Faroby
NRP	: 06111340000030
Department	: Mathematics
Supervisor	: Prof. Mohammad Isa Irawan, MT

ABSTRACT

Research on DNA / RNA leukemia sequences can facilitate and accelerate for medical personnel in diagnosing leukemia types. Leukemia data were extracted by extracting sequence data characteristic of string type to numeric form by using Markov 2 chain method 2. The result of characteristic extraction is 64 characteristic of DNA / RNA leukemia in the form of Markovian matrix. Machine Learning is used in the classification process with the approach method of Support Vector Machine (SVM). At the time of classification, this research uses 40 training data as data classifier and 25 test data. SVM divides the class from DNA / RNA blood cancer into four parts namely Acute Myeloid Leukemia, Acute Lymphocytic Leukemia, Chronic Myelogenous Leukemia and Chronic Lymphocytic Leukemia. The parameters in the SVM were tested for obtaining optimal values in which SVM worked well on leukemia data. The result of the research is the image of trainer data on SVM. The image is a 2D graph where 64 features of DNA / RNA leukemia will be compressed using Principle Component Analysis (PCA). Other research results are validated predictions with experiments on the test data. And compared to the results of three kernels are linear, gaussian and polynomial. The experiment was used to determine which kernel performance is best used in DNA / RNA leukemia sequence data.

Keywords: Markov Chain, Machine Learning, Support Vector Machine, Blood Cancer.

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Alloh SWT, karena dengan ridlo-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

IDENTIFIKASI JENIS KANKER DARAH (LEUKEMIA) TERHADAP PENGARUH PARAMETER KERNEL *SUPPORT VECTOR MACHINE* DAN EKSTRAKSI CIRI DENGAN RANTAI MARKOV ORDE 2

yang merupakan salah satu prasyarat akademis dalam menyelesaikan Program Sarjana (S1) Departemen Matematika di Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dukungan dan doa dari banyak pihak. Sehubungan dengan hal tersebut, penulis ingin mengucapkan terimakasih kepada :

1. Dr. Imam Mukhlas, S.Si, MT selaku Kepala Departemen Matematika ITS dan selaku penguji juga dalam penelitian tugas akhir ini yang memberi arahan selama proses penelitian.
2. Kedua Orangtua penulis, yang selalu mendukung, mendoakan dan memotivasi penulis.
3. Prof. Dr. Mohammad Isa Irawan, MT selaku dosen pembimbing yang telah memberi bimbingan dan motivasi kepada penulis dalam mengerjakan penelitian Tugas Akhir sehingga dapat terselesaikan dengan baik.
4. Dr. Didik Khusnul Arif, S.Si , M.Si selaku Kepala Program Studi S1 Departemen Matematika ITS
5. Drs. Iis Herisman, M.Si selaku Sekretaris Program Studi S1 Departemen Matematika ITS yang selalu memberi arahan dan pelayanan administrasi selama mengerjakan tugas Akhir.
6. Dra. Wahyu Fistia Doctorina, M.Si selaku dosen wali penulis yang selalu memberi arahan terhadap penulis selama kuliah di Departemen Matematika ITS.
7. Seluruh jajaran dosen dan staf Departemen Matematika ITS yang telah memberikan ilmu pengetahuan kepada penulis selama diperkuliahan.

8. Teman-teman Angkatan 2013 yang selalu mendukung dan memotivasi. Terutama para pejuang 117 yang selalu berjuang bersama penulis.
9. Sahabat-sahabat dan teman dekat penulis, Fadlan, Chyntia, Dedi, Sandy, Amin, Maria, Brian, Humai dan lain-lain yang tidak bisa penulis sebutkan satu-satu. Terimakasih atas dukungan dan motivasi kalian.
10. Semua pihak yang belum bisa penulis sebutkan satu persatu. Terima kasih telah membantu sampai terselesaikannya Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik dan saran dari pembaca. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, 11 Januari 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL DAN DAFTAR DIAGRAM

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

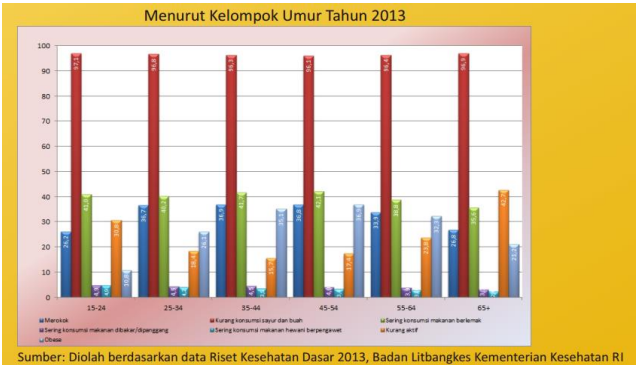
Pada bab ini dibahas mengenai latar belakang yang mendasari penulisan Tugas Akhir ini. Di dalamnya mencakup identifikasi permasalahan pada topik Tugas Akhir kemudian dirumuskan menjadi permasalahan yang diberikan batasan-batasan dalam pembahasan pada Tugas Akhir ini.

1.1 Latar Belakang Masalah

Masalah kesehatan merupakan sektor yang sangat penting diperhatikan disetiap negara, terutama penyakit kanker. Penyakit kanker merupakan salah satu penyebab kematian utama diseluruh dunia. Banyak peneliti berlomba-lomba mencari cara menangani penyakit tersebut. Kanker sendiri merupakan istilah umum untuk satu kelompok besar penyakit yang dapat mempengaruhi setiap bagian dari tubuh, istilah lain yang digunakan adalah tumor ganas dan neoplasma. Salah satu fitur mendefenisikan kanker adalah pertumbuhan sel-sel baru secara abnormal yang tumbuh melampaui batas normal dan kemudian dapat menyerang bagian sebelah tubuh dan menyebar ke organ yang lain, proses ini disebut metastasis [1].

Kanker berkembang apabila gen-gen pada sel normal mengalami gangguan genetik atau terjadi mutasi. Pertumbuhan yang tidak terkendali tersebut disebabkan kerusakan pengkodeaan pada rantai asam deoksirebonukleat atau biasa disebut DNA, menyebabkan mutasi di gen vital yang mengontrol pembelahan sel [2]. Mutasi pada DNA dapat terjadi karena berbagai sebab seperti faktor lingkungan, terutama paparan bahan kimiawi tertentu pada tempat kerja, polusi lingkungan, merokok, konsumsi alkohol berlebihan, infeksi virus atau bakteri, radiasi matahari dan radiasi ion, serta diet. Data dari Kementrian kesehatan Republik Indonesia tahun 2013 penderita kanker untuk segala umur paling tinggi dikarenakan kurangnya makan

sayur dan buah-buahan[2]. Seperti yang ditampilkan pada Gambar 1.1.

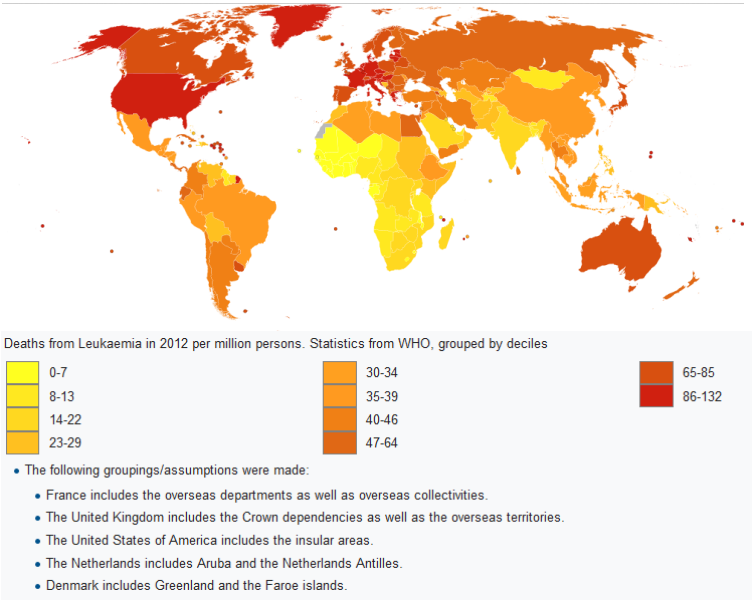


Gambar 1.1 Data penyebab kanker berdasarkan kelompok umur tahun 2013

Teknologi yang digunakan pada biomarker pada praktek klinik tidak cukup kuat untuk mendeteksi adanya bibit-bibit kanker, karena pada praktek klinik biomarker membutuhkan biaya yang mahal dan waktu yang terlalu lambat dalam mendeteksi, maka penelitian baru berupa *microarray* digunakan untuk mendeteksi secara dini [3]. Keberhasilan dalam menganalisis memerlukan identifikasi gen yang terkait dengan kelas tumor dan penghilangan variabel yang tidak diperlukan. Analisis yang buruk menyebabkan data *overfitting* dan hasil yang tidak dapat diidentifikasi [4]. Teknik analitik tradisional seperti pengelompokan hirarki, mengasumsikan linearitas biologis dan menggunakan pendekatan statistik untuk menyimpulkan hubungan kelas gen (*feature selection*), semua itu berkerja sangat buruk ketika menganalisis *dataset* yang terkontaminasi dengan variabel noice. Kecerdasan Buatan (*Artificial Inteligence*) merupakan metode pembelajaran mesin dengan pendekatan tanpa persaratan apapun, berbagai metode pembelajaran mesin yang ada seperti *Support Vector Machine, Artificial Neural Network, Bayesian*

Network, *Genetic Algorithms* dan lain-lain . Beberapa analisis microarray ekspresi gen yang telah sukses dengan menggunakan metode *Support Vector Machine* (SVM) dan *Artificial Neural Network* (ANN) [5].

Penelitian yang pernah mengklasifikasikan sekuens DNA yang dilakukan oleh Fajar adalah meneliti sekuens DNA bakteri dan menerapkan metode klasifikasi *Probabilistik Neural Network* (PNN) dengan membandingkan rantai Markov orde 1 dan orde 2, menghasilkan klasifikasi lebih akurat dengan menggunakan Markov orde 2 [9]. Pada penelitian yang dilakukan dalam tugas akhir ini menerapkan pada objek DNA manusia yang mengalami mutasi karena kanker dan menggunakan metode *Support Vector Machine* (SVM) sebagai metode klasifikasi. Pada penelitian ini obyek penelitian difokuskan pada jenis leukemia *Acute Lymphoblastic Leukemia* (ALL), *Acute Myeloid Leukemia* (AML), *Chronic Lymphocytic Leukemia* (CLL) dan *Chronic Myeloid Leukemia* (CML). Karena pada pada umumnya empat jenis kanker tersebutlah yang sering terjadi di dunia dan menjadi masalah krusial jika lamban dalam penanganannya. Data dari WHO menyebutkan bahwa penyakit leukemia merupakan penyakit yang sering terjadi di daerah Amerika Utara sekitar 86-132 perjuta orang. Pada Gambar 1.2 menunjukkan area diseluruh dunia yang sering mengalami leukemia. Afrika merupakan benua yang sangat rendah terjadinya penyakit leukemia ini.



Gambar 1.2 Data WHO penderita leukemia di dunia

Konsep dasar SVM sebenarnya merupakan kombinasi harmonis dari teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti *margin hyperplane*. Dengan mengklasifikasikan sekuens DNA/RNA dari Gen yang terkena kanker darah akan memudahkan diagnosa dan pengobatan terhadap jenis kanker tertentu. Pada pengklasifikasian dilakukan percobaan terhadap parameter-parameter SVM untuk mengetahui seberapa baik SVM bekerja dan pada parameter bagaimana SVM bekerja dengan baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, penulis merumuskan permasalahan dalam tugas akhir sebagai berikut:

1. Metode apa yng digunakan untuk mengolah data sekuens DNA/RNA leukemia?

2. Bagaimana mengolah data sekuens DNA/RNA leukemia agar bisa digunakan?
3. Bagaimana metode mengklasifikasikan data agar bisa dijadikan informasi yang bermanfaat ?
4. Paramater bagaimana yang menjadikan SVM bekerja dengan baik pada data leukemia ?

1.3 Batasan Masalah

Pada penelitian ini penulis membuat batasan:

1.Sample data menggunakan sekuens data dari GenBank DNA yaitu NCBI (*National Center of Biotechnology Information*) dan EMBL (*European Molecular Biology Laboratory*)

2.Data menggunakan 4 jenis leukemia, ALL, CLL, AML, CML.

1.4 Tujuan

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, tujuan penelitian Tugas Akhir ini adalah:

1.Membuat suatu aplikasi untuk mendeteksi jenis kanker darah dengan objek sekuens DNA/RNA dengan klasifikasi SVM dan ekstraksi ciri rantai markov .

2.Menganalisa data sekuens DNA/RNA berupa tipe data string ke dalam tipe data numerik dengan metode rantai markov.

3.Mengklasifikasikan data menjadi 4 golongan yakni ALL,CLL,AML dan CML.

4.Menguji parameter-parameter pada saat pengklasifikasian untuk mendapatkan performa terbaik SVM.

1.5 Manfaat

Setelah pengimplementasian pengembangan perangkat lunak untuk mengidentifikasi mutasi gen penyebab kanker darah, maka Tugas Akhir ini dapat memberikan manfaat sebagai berikut :

1.Mempermudah dalam bidang kesehatan dengan mengolah data sekuens DNA pasien untuk medeteksi mutasi gen dan mengetahui jenis kanker darah.
2.Membantu dalam mendiagnosa medis pasien jika terjadi kanker darah.
3.Sebagai referensi penelitian selanjutnya untuk identifikasi jenis kanker yang lain.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika dari penulisan Tugas Akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi tentang gambaran umum dari penulisan Tugas Akhir yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini berisi tentang teori-teori keilmuan untuk menunjang proses penelitian. Mencakup teori-teori keilmuan berupa keterangan data, pengertian istilah, eksraksi ciri, metode klasifikasi dan penelitian yang pernah dilakukan.

BAB III METODOLOGI PENELITIAN

Pada bab ini dibahas tentang langkah – langkah dan metode yang digunakan untuk menyelesaikan Tugas Akhir ini.

BAB IV PERANCANGAN DAN PROSES SISTEM

Pada bab ini akan menguraikan bagaimana tahapan tahapan dalam analisis dan perancangan sistem. Perancangan sistem dimulai dari deskripsi perangkat

lunak perangcangan Data Flow Diagram hingga pemodelan analisis sistem, sedangkan proses sistem dimulai dari proses pengolahan data mentah hingga menjadi informasi yang berguna.

BAB V HASIL DAN PEMBAHASAN PENGUJIAN

Bab ini menjelaskan mengenai pengujian terhadap sistem yang telah dibuat. Input program memuat data ekspresi gen, dan output program mengklasifikasikan data inputan tersebut ke class yang memiliki kemiripan lebih banyak.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari pembahasan masalah sebelumnya serta saran yang diberikan untuk pengembangan selanjutnya.

Halaman ini sengaja dikosongi

BAB II
TINJAUAN PUSTAKA

Pada bab ini menjelaskan teori-teori penunjang dalam melakukan penelitian tugas akhir ini. Berisi teori-teori yang menjelaskan data yang digunakan, metode dari ekstraksi ciri , metode-metode dalam mengklasifikasikan data dan juga bahasa pemrograman yang digunakan untuk mengolah data serta penelitian yang terkait sebelumnya.

2.1 Penelitian Yang Terkait

Penelitian tentang DNA/RNA dengan machine learning itu bermacam-macam. Pendekatan dengan machine learning memodelkan karateristik dari suatu DNA/RNA menjadi model matematika. Sehingga dengan bantuan perhitungan komputer, proses penyelesaian masalah menjadi lebih cepat. Perkembangan metode ekstraksi ciri dan juga metode machine learning bermacam-macam.

Pada tahun 2012, penelitian yang dilakukan oleh Kusuma WA adalah mencari ekstraksi ciri yang efektif dengan menggunakan metode markovian, penelitian yang dia lakukan menghasilkan pola yang memiliki akurasi tinggi yaitu pola dengan *weight of pattern* =3, yakni banyaknya basa nitrogen yang digunakan untuk membentuk pola. Selanjutnya pada tahun 2013, M. Lutfi Fajar mengidentifikasi DNA/RNA bakteri dengan rantai Markov orde 1 dan orde 2 dan juga menerapkan PNN(*Probabilistik Neural Network)* sebagai klasifikasinya. Model terbaik pada penelitianya menggunakan ekstraksi ciri dengan rantai markov orde 2, dimana semakin panjang sequence semakin besar pula nilai sensitivitas dan spesifikasinya.

Pada penelitian ini, objek penelitian yaitu DNA/RNA dari manusia yang terjangkit kanker darah atau leukemia. Metode yang

digunakan yaitu model markovian orde 2 dengan *weight of pattern* =3 dan juga klasifikasi yang digunakan yaitu *support vector machine* (SVM). *Support Vector Machine* memiliki kelebihan dibanding metode machine learning yang lain yaitu:

1.Efektif di ruang berdimensi tinggi
2.Masih efektif jika jumlah dimensi lebih besar dari jumlah sampelnya
3.Menggunakan subset dari titik latih pada fungsi keputusannya, sehingga memori yang digunakan lebih effisien.
4.Serbaguna, karena fungsi kernel yang berbeda dapat menentukan untuk pengambilan fungsi keputusan.

2.2 Kanker

Kanker adalah penyakit akibat pertumbuhan tidak normal dari sel-sel jaringan tubuh yang berubah menjadi sel kanker. Dalam perkembangannya, sel-sel kanker ini dapat menyebar ke bagian tubuh lainnya sehingga dapat menyebabkan kematian. Kanker sering dikenal oleh masyarakat sebagai tumor, padahal tidak semua tumor adalah kanker. Tumor adalah segala benjolan tidak normal atau abnormal. Tumor dibagi dalam 2 golongan, yaitu tumor jinak dan tumor ganas. Kanker adalah istilah umum untuk semua jenis tumor ganas. Kanker dapat menimpa semua orang, pada setiap bagian tubuh, dan pada semua gologan umur, namun lebih sering menimpa orang yang berusia 40 tahun [6]. Umumnya sebelum kanker meluas atau merusak jaringan di sekitarnya, penderita tidak merasakan adanya keluhan ataupun gejala. Bila sudah ada keluhan atau gejala, biasanya penyakitnya sudah lanjut.

2.2.1 Kanker Darah (Leukemia)

Leukemia adalah kanker darah yang berawal dalam sumsum tulang belakang, tempat sel darah dibuat. Darah mengandung sel merah, sel putih dan trombosit. Pada leukemia, sumsum tulang

belakang membuat sel darah putih yang belum matang yang disebut sel leukemik. Sel yang belum matang ini tidak berfungsi secara normal, dan mengerubungi sel yang sehat. Leukemia bisa bersifat akut (memburuk secara cepat), atau kronis (memburuk secara perlahan). Jenis-jenis leukemia adalah:

1. Acute Lymphoblastic Leukemia (ALL)

Jenis leukemia yang paling umum pada anak kecil. Ini juga mempengaruhi orang dewasa, terutama yang berusia 65 tahun ke atas. Subtipe meliputi dari ALL meliputi:

a. *Precursor B acute lymphoblastic leukemia*
b. *Precursor T acute lymphoblastic leukemia*
c. *Burkitt's leukemia*
d. *Acute biphenotypic leukemia*

2. Chronic Lymphocytic Leukemia (CLL)

Paling sering menyerang orang dewasa di atas usia 55 tahun. Kadang kala terjadi pada orang dewasa muda, tapi hampir tidak pernah menyerang anak-anak. Ini tidak dapat disembuhkan, namun ada banyak perawatan yang efektif. Salah satu subtipenya adalah *B-cell prolymphocytic leukemia* yang merupakan penyakit yang lebih agresif.

3. Acute Myeloid Leukemia (AML)

Lebih sering pada orang dewasa dari pada pada anak-anak, dan lebih sering pada pria dari pada wanita. Hal ini diobati dengan kemoterapi. Subtipe AML meliputi :

a. *Acute promyelocytic leukemia*
b. *Acute myeloblastic leukemia*
c. *Acute megakaryoblastic leukemia*

4. Chronic Myeloid Leukemia (CML)

Terjadi terutama pada orang dewasa, sejumlah kecil anak juga mengalami penyakit ini. Salah satu subtipe adalah *Chronic Myelomonocytic Leukemia*.

2.2.2 Proses Diagnosa Kanker

Jenis kanker dan pengobatannya bervariasi, maka mendiagnosis adanya kanker dan menentukan jenisnya merupakan hal yang sangat penting. Hal ini hampir selalu memerlukan pengambilan contoh jaringan kanker untuk diperiksa dibawah mikroskop. Sejumlah tes khusus terhadap contoh jaringan kanker mungkin diperlukan untuk menggambarkan lebih jauh mengenai kanker yang ditemukan. Bila jenis kankernya diketahui, akan membantu dokter dalam menentukan pemeriksaan yang akan dilakukan, karena setiap kanker cenderung mengikuti suatu pola pertumbuhan dan penyebaran tertentu. Tenaga medis biasanya dapat menentukan jenis tumor utamanya dengan melakukan biopsi dari kanker yang bermetastase dan memeriksanya dibawah mikroskop. Namun identifikasi kanker tidak selalu mudah dan pasti.

2.2.3 Stadium Kanker

Setelah mendiagnosa adanya kanker dan jenis kanker oleh medis, pemeriksaan selanjutnya dengan tahap penentuan stadium kanker. Serangkaian pemeriksaan digunakan untuk menentukan lokasi tumor, ukurannya, pertumbuhannya ke jaringan di sekitar dan penyebarannya ke bagian tubuh yang lain. Pemeriksaan untuk menentukan stadium kanker biasanya melalui beberapa macam pemeriksaan yaitu:

a. Scan

Scaning ultrasonik merupakan prosedur non-invasif dan tidak menimbulkan nyeri, yang menggunakan gelombang suara untuk menunjukkan struktur organ dalam. Pemeriksaan ini membantu dalam menentukan ukuran kanker tertentu, terutama kanker ginjal, hati, panggul dan prostat.

b. Pewarnaan

Limfangiogram adalah suatu pemeriksaan dimana zat warna disuntikkan ke dalam kaki lalu dilakukan pemotretan rontgen. Pemeriksaan ini membantu menemukan kelainan dalam kelenjar getah bening perut dan menentukan stadium dari penyakit Hodgkin dan kanker buah zakar.

c. CT (computed temography) atau MRI (magnetic resonance imaging)

CT scan digunakan untuk menemukan kanker di otak, paru-paru dan organ perut, termasuk kelenjar adrenal, kelenjar getah bening, hati dan limpa. Dengan prosedur MRI bisa menemukan kanker otak, tulang dan korda tulang belakang

d. Mediastinoskopi

e. Biopsi sumsum tulang.

Pembedahan kadang juga diperlukan untuk menentukan stadium kanker. Misalnya suatu laparotomi (pembedahan perut) memungkinkan ahli bedah untuk mengangkat atau mengobati kanker usus besar sambil menentukan penyebaran kanker ke kelenjar getah bening terdekat. Analisa kelenjar getah bening yang diangkat dari ketiak pada saat dilakukan mastektomi, membantu menentukan seberapa jauh kanker payudara telah menyebar dan apakah diperlukan terapi paska pembedahan.

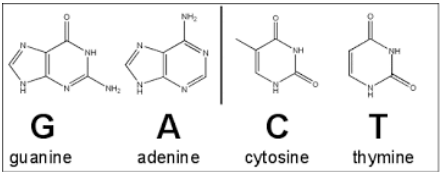
2.3 DNA dan RNA

Deoksirebonukleat acid dan rebonukleat acid merupakan materi genetic dari organisme. Setiap organisme memiliki materi genetik ini yang berada pada sel-sel organisme.

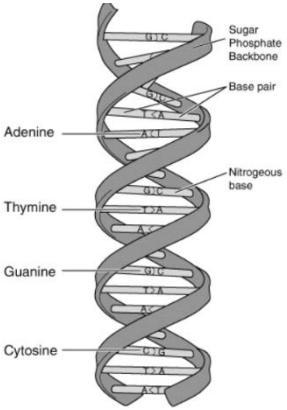
2.3.1 Asam Deoksirebonukleat

Asam Deoksirebonukleat atau biasa disebut DNA adalah sejenis biomolekul yang menyandikan instruksi-intruksi genetika pada setuap organisme. Kebanyakan DNA terdiri dari dua untai biopolymer yang saling memilin membentuk *double helix*. Dua

untai dikenal dengan polinukleotida karena keduanya terdiri dari satuan-satuan molekul nukleotida. Setiap nukleotida terdiri dari salah satu basa nitrogen yaitu Adenin (A), Sitosin (C), Guanin (G) dan Timin (T) yang dapat dilihat pada Gambar 2.1.



Gambar 2.1. Macam-macam basa nitrogen



Gambar 2.2. Ikatan basa nitrogen

Dua unting DNA bersifat anti-paralel, yang berarti bahwa keduanya berpasangan secara berlawanan. Pasangan basa nitrogennya adalah Adenin berpasangan dengan Timin dan Sitosin berpasangan dengan Guanin seperti yang digambarkan pada Gambar 2.2. Pada setiap gugus gula, terikat salah satu dari empat jenis nukleobasa. Urutan-urutan empat nukleobasa di sepanjang rantai punggung DNA inilah yang menyimpan kode informasi biologis. Melalui proses biokimia yang disebut transkripsi, unting DNA digunakan sebagai *template* untuk membuat unting RNA.

Unting RNA ini kemudian ditranslasikan untuk menentukan urutan asam amino protein yang dibangun. Proses tersebut dinamakan Sintesis protein.

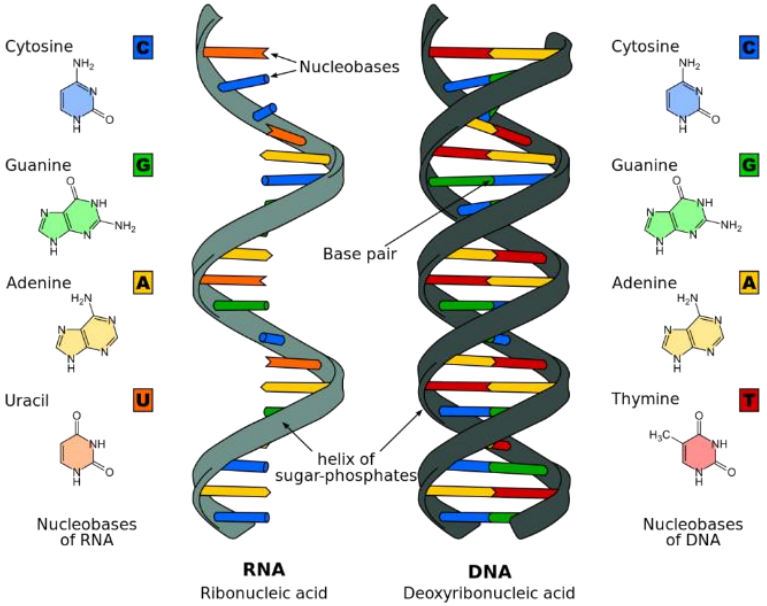
2.3.2 Asam Ribonukleat

ARN atau biasa disebut asam ribonukleat (RNA) merupakan molekul polimer yang terlibat dalam berbagai peran biologis dalam mengkode, dekode, regulasi, dan ekspresi gen. RNA dan DNA adalah asam nukleat, bersama dengan protein dan karbohidrat membentuk empat makromolekul utama yang penting untuk semua bentuk kehidupan yang diketahui. Seperti DNA, RNA dirakit sebagai rantai nukleotida, tetapi tidak seperti DNA, RNA lebih sering ditemukan di alam sebagai untai tunggal yang melipat ke dirinya sendiri, daripada untai ganda berpasangan. Beberapa molekul RNA berperan aktif dalam sel dengan mengkatalis reaksi biologis, mengendalikan ekspresi gen, atau merasakan dan mengkomunikasikan tanggapan terhadap sinyal seluler.

Struktur dasar RNA mirip dengan DNA. RNA merupakan polimer yang tersusun dari sejumlah nukleotida. Setiap nukleotida memiliki satu gugus fosfat, satu gugus pentosa, dan satu gugus basa nitrogen (basa N). Polimer tersusun dari ikatan berselang-seling antara gugus fosfat dari satu nukleotida dengan gugus pentosa dari nukleotida yang lain. Perbedaan RNA dengan DNA terletak pada satu gugus hidroksil cincin gula pentosa, sehingga dinamakan ribosa, sedangkan gugus pentosa pada DNA disebut deoksiribosa[12]. Basa nitrogen pada RNA sama dengan DNA, kecuali basa timina pada DNA diganti dengan urasil pada RNA. Jadi tetap ada empat pilihan: adenina, guanina, sitosina, atau urasil untuk suatu nukleotida.

Perbedaan dari DNA dan RNA bisa dilihat dari Gambar 2.3. RNA merupakan unting tunggal dari DNA dimana RNA merupakan pembawa pesan dari DNA ke ribosom untuk pembentukan protein (sintesis protein). Pada urutan basa nitrogen

RNA terdapat urasil, dimana pada saat transkripsi DNA oleh RNA basa timin pada DNA diubah menjadi basa urasil.



Gambar 2.3 Perbedaan DNA dan RNA

2.3.2.1 Macam-macam RNA

Dalam mengantarkan kode genetik, terutama pada sintesis protein, RNA terbagi menjadi tiga jenis yaitu:

a. *Messengger*-RNA (mRNA)

mRNA merupakan RNA yang sintesisnya diarahkan oleh gen pada berkas DNA sebagai pembawa pesan. Dengan kata lain, mRNA adalah RNA yang merupakan hasil transkripsi DNA dan menjadi perantara pembawa urutan protein dalam proses translasi. mRNA memiliki tugas utama menjadi pembawa pesan kode dari DNA. Panjang pendeknya mRNA berhubungan dengan panjang pendeknya rantai polipeptida yang akan disusun. Urutan asam

amino yang menyusun rantai polipeptida itu sesuai dengan urutan kodon yang terdapat di dalam molekul mRNA yang bersangkutan. Fungsi utama mRNA adalah membawa kode-kode genetik dari DNA di inti sel menuju ke ribosom di sitoplasma.

b. Ribosomal-RNA (rRNA)

rRNA merupakan RNA polimer yang berada di ribosom. mRNA yang telah mentranskrip DNA melekat pada enzim RNA polymerase sehingga membentuk rRNA. Fungsi dari RNA ribosom adalah sebagai mesin perakit dalam sintesis protein yang bergerak ke satu arah sepanjang mRNA.

c. Transfer-RNA (tRNA)

tRNA merupakan pembawa pesan yang mengkodekan asam amino menjadi protein. Di dalam sitoplasma banyak terdapat tRNA, asam-asam amino dan lebih dari 20 enzim-enzim amino hasil sintesis. tRNA merupakan RNA terpendek dan bertindak sebagai penerjemah kodon dari mRNA. Fungsi lain tRNA adalah mengikat asam-asam amino di dalam sitoplasma yang akan disusun menjadi protein dan mengangkutnya ke ribosom. Bagian tRNA yang berhubungan dengan kodon dinamakan antikodon.

2.4 Sempel Data

Data DNA diambil dari NCBI(*National Center of Biotechnology Information*) atau European Molecular Biology Laboratory (EMBL). Kedua sumber data tersebut merupakan DNA bank untuk penelitian genetic didunia. Sehingga pada penelitian ini data yang ada pada NCBI ataupun EMBL merupakan data sekunder pada penelitian ini. Data yang disediakan oleh NCBI dan EMBL berformat .FASTA atau .txt, Sampel data juga dapat diakses pada MATLAB online. Data sekuens DNA berupa replica yang telah menjadi monohelix saja, yang berupa mRNA, rRNA ataupun tRNA. Sehingga lebih memudahkan dalam mengekstraksi

ciri dari tipe gen leukemia. Contoh dari data sekuens yang berformat .FASTA dapat dilihat pada gambar 2.4.

```
>ENA|X79549|X79549.1 H.sapiens AML1 mRNA

GCGGGCGCGGGCGGGACGGGCGCCCCGCGGCCGGACCCAGCCAGGGCACCACGCTGCC
CGGCCCTGCGCCGCCAGGCACTTCTTTCCGGGGCTCCTAGGGACGCCAGAAGGAAGTC
AACCTCTGCTGCTTCTCCTTGGCCTGCGTTGGACCTTCCTTTTTTTGTTGTTTTTTTTGTT
TTTCCCCTTTCTTCCTTTTGAATTAACTGGCTTCTTGGCTGGATGTTTTCAACTTCTTTCCT
GGCTGCGAACTTTTCCCCAATTGTTTTCCTTTTACAACAGGGGGAGAAAGTGCTCTGTG
GTCCGAGGCGAGCCGTGAAGTTGCGTGTGCGTGGCAGTGTGCGTGGCAGGATGTGCGT
GCGTGTGTAACCCGAGCCGCCCGATCTGTTTCGATCTGCGCCGCGGAGCCCTCCCTCAA
GGCCCGCTCCACCTGCTGCGGTTACGCGGCGCTCGTGGGTGTTCGTGCCTCGGAGCAGC
TAACCGGCGGGTGCTGGGCGACGGTGGAGGAGTATCGTCTCGCTGCTGCCCGAGTCAG
GGCTGAGTCACCCAGCTGATGTAGACAGTGGCTGCCTTCCGAAGAGTGCGTGTTTGCAT
GTGTGTGACTCTGCGGCTGCTCAACTCCCAACAAACCAGAGGACCAGCCACAAACTTA
ACCAACATCCCCAAACCCGAGTTCACAGATGTGGGAGAGCTGTAGAACCCTGAGTGTC
ATCGACTGGGCCTTCTTATGATTGTTGTTTTAAGATTAGCTGAAGATCTCTGAAACGCTG
AATTTTCTGCACTGAGCGTTTTGACAGAATTCATTGAGAGAACAGAGAACATGACAAGT
ACTTCTAGCTCAGCACTGCTCCAACTACTGAAGCTGATTTTCAAGGCTACTTAAAAAAA
TCTGCAGCGTACATTAATGGATTTCTGTTGTGTTTAAATTCTCCACAGATTGTATTGTAA
ATATTTTATGAAGTAGAGCATATGTATATATTTATATATACGTGCACATACATTAGTAGC
ACTACCTTTGGAAGTCTCAGCTCTTGCTTTTCGGGACTGAAGCCAGTTTTGCATGATAAA
AGTGGCCTTGTTACGGGAGATAATTGTGTTCTGTTGGGACTTTAGACAAAACTCACCTG
CAAAAAACTGACAGGCATTAACTACTGGAACTTCCAAATAATGTGTTTGCTGATCGTTT
TACTCTTCGCATAAATATTTTAGGAAGTGTATGAGAATTTTGCCTTCAGGAACTTTTCTA
ACAGCCAAAGACAGAACTTAACCTCTGCAAGCAAGATTCGTGGAAGATAGTCTCCACT
TTTTAATGCACTAAGCAATCGGTTGCTAGGAGCCCATCCTGGGTCAGAGGCCGATCCGC
AGAACCAGAACGTTTTCCCCTCCTGGACTGTTAGTAACTTAGTCTCCCTCCTCCCCTAAC
CACCCCCGCCCCCCCCCACCCCCCGCAGTAATAAAGGCCCCTGAACGTGTATGTTGGTC
TCCCGGGAGCTGCTTGCTGAAGATCCGCGCCCCTGTCGCCGTCTGGTAGGAGCTGTTTG
CAGGGTCCTAACTCAATCGGCTTGTTGTGATGCGTATCCCCGTAGATGCCAGCACGAGC
CGCCGCTTCACGCCGCCTTCCACCGCGCTGAGCCCAGGCAAGATGAGCGAGGCGTTGCC
GCTGGGCGCCCCGGACGCCGGCGCTGCCCTGGCCGGCAAGCTGAGGAGCGGCGACCGC
AGCATGGTGGAGGTGCTGGCCGACCACCCGGGCGAGCTGGTGCGCACCGACAGCCCCA
ACTTCCTCTGCTCCGTGCTGCCTACGCACTGGCGCTGCAACAAGACCCTGCCCATCGCTT
TCAAGGTGGTGGCCCTAGGGGATGTTCCAGATGGCACTCTGGTCACTGTGATGGCTGGC
AATGATGAAAACTACTCGGCTGAGCTGAGAAATGCTACCGCAGCCATGAAGAACCAGG
TTGCAAGATTTAATGACCTCAGGTTTGTCGGTCGAAGTGGAAGAGGGAAAAGCTTCACT
CTGACCATCACTGTCTTCACAAACCCACCGCAAGTCGCCACCTACCACAGAGCCATCAA
AATCACAGTGGATGGGCCCCGAGAACCTCGAAGACATCGGCAGAAACTAGATGATCAG
ACCAAGCCCGGGAGCTTGTCCTTTTCCGAGCGGCTCAGTGAACTGGAGCAGCTGCGGCG
CACAGCCATGAGGGTCAGCCCACACCACCCAGCCCCCACGCCCAACCCTCGTGCCTCCC
TGAACCACTCCACTGCCTTTAACCCTCAGCCTCAGAGTCAGATGCAGGATACAAGGCAG
ATCCAACCATCCCCACCGTGGTCCTACGATCAGTCCT
```

Gambar 2.4 Data asli dari NCBI

2.5 Ekstraksi Ciri Rantai Markov

Rantai markov merupakan metode stokastik untuk menentukan deskripsi *state* yang sesuai, sehingga proses yang berpaduan akan benar-benar memiliki sifat markov (*marcovian property)*, yaitu pengetahuan terhadap state untuk memprediksi prilaku stokastik yang akan datang. Suatu rantai markov dikatakan diskrit jika ruang markov tersebut adalah himpunan terbatas atau tercacah. Jika nilai dari suatu *state* hanya bergantung pada satu periode sebelumnya, maka disebut rantai orde satu dan jika suatu *state* pada periode tertentu bergantung pada *m* periode sebelumnya maka dinamakan rantai markov orde *m*. Rantai markov orde 1 dirumuskan:

$$P\{X_{n+1} = j | X_n = i\} \quad n = 1,2,3, \dots, l$$

dimana X_n merupakan state tertentu pada nilai n, dan X_{n+1} merupakan state *n+1* yang mempengaruhi X_n. Nilai dari probabilitas dihitung terus sampai l dimana panjang sekuens. Sedangkan untuk rantai markov orde m dirumuskan sebagai berikut :

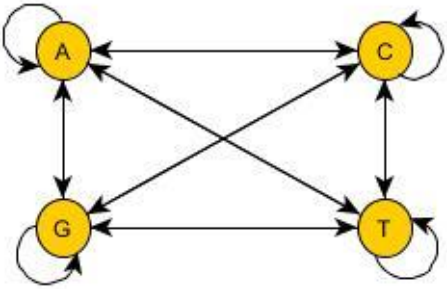
$$P\{X_{n+1} = j | X_{(n+1)-m} = i_1, X_{(n+1)-(m+1)} = i_2, \dots, X_n = i_n\}$$

hasil dari probabilitas setiap komponen rantai akan dijadikan matriks transisi, yang disebut peluang matriks transisi *P*, yang setiap unsurnya adalah P_{ij}, seperti pada bentuk matriks dibawah ini:

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \dots & P_{nn} \end{bmatrix}$$

syarat dari matriks transisi bersifat Markovian adalah jumlah dari setiap baris bernilai 1.

Pada penelitian ini, matriks peluang transisi dibentuk dari peluang-peluang munculnya basa tertentu dengan basa tertentu sebelumnya. Dengan mengacu pada komponen basa DNA ada 4 jenis basa yaitu, Adenin (A), Timin (T), Guanin (G) dan sitosin (C).



Gambar 2.5 Graf pembentuk markovian

Dengan demikian matrik transisi rantai markov orde satu terdiri atas $P(A|A)$, yaitu peluang munculnya basa A setelah sebelumnya basa A, $P(C|A), P(T|A), P(G|A)$ dan seterusnya. Sehingga pada matriks transisi markov orde 1 terdiri dari 16 elemen (Robin et al.2005). Matriks transisi orde satu untuk basa DNA dapat ditulis sebagai berikut :

$$P = \begin{bmatrix} P(A|A) & P(C|A) & P(G|A) & P(T|A) \\ P(A|C) & P(C|C) & P(G|C) & P(T|C) \\ P(A|G) & P(C|G) & P(G|G) & P(T|G) \\ P(A|T) & P(C|T) & P(G|T) & P(T|T) \end{bmatrix}$$

2.6 Principle Component Analysis

PCA adalah teknik statistik yang sudah digunakan secara luas baik dalam hal pengenalan wajah maupun pengenalan pola dari sebuah gambar. Metode *Principal Component Analysis* (PCA) dibuat pertama kali oleh para ahli statistik dan ditemukan oleh Karl

Pearson pada tahun l90l yang memakainya pada bidang biologi. Kemudian tidak ada perkembangan baru pada teknik ini, dan. perkembangannya baru mulai pesat pada akhir tahun l930 dan awal 1940. Setelah itu perkembangannya berkurang sebentar sampai komputer telah berhasil didesain sehingga dapat mengaplikasikan teknik ini pada masalah-masalah yang masuk akal. Pada tahun 1947 teori ini muncul lagi dan cukup independen sebagai teori probabilitas yang ditemukan oleh Karhunen, dan kemudian dikembangkan oleh Loeve pada tahun l963, sehingga teori ini juga dinamakan Karhunen-Loeve transform pada bidang ilmu telekomunikasi.

Principle Component Analysis atau disingkat PCA adalah sebuah teknik untuk membangun variable-variable baru yang merupakan kombinasi linear dari variable-variable asli. Jumlah maximum dari variable-variable baru ini akan sama dengan jumlah dari variable lama, dan variable-variable baru ini tidak saling berkorelasi satu sama lain.

Metode ekstraksi ciri PCA digunakan untuk menyederhanakan matriks ciri yang dibentuk oleh rantai markov. Penggunaan matriks yang sudah direduksi untuk memunculkan data leukemia pada grafik 2 Dimensi dan 3 Dimensi. Sehingga untuk grafik 2 dimensi ciri yang dibutuhkan untuk setiap data training hanya 2 ciri sedangkan untuk 3 dimensi membutuhkan 3 ciri data training.

2.7 *Machine Learning*

Machine Learning merupakan salah satu cabang dari disiplin ilmu Kecerdasan Buatan (*Artificial Intellegence*) yang membahas mengenai pembangunan sistem yang berdasarkan pada data. Banyak hal yang dipelajari, akan tetapi pada dasarnya ada 4 hal pokok yang dipelajari dalam machine learning.

1. Pembelajaran Terarah (Supervised Learning)
2. Pembelajaran Tak Terarah (Unsupervised Learning)

3. Pembelajaran Semi Terarah (Semi-supervised Learning)
4. *Reinforcement Learning*

Salah satu dampak positif dari machine learning adalah menjadi peluang bagi para wirausahawan dan praktisi teknologi untuk terus berkarya dalam mengembangkan teknologi *machine learning*. Dengan adanya *machine learning* biaya penelitian konvensional yang tergolong sangat mahal dan memakan banyak waktu yang lama dapat tekan biaya menjadi murah dan waktu yang cepat.

2.7.1 *Support Vector Machine*

Support Vector Machine (SVM) pertama kali diperkenalkan oleh Vapnik pada tahun 1992 sebagai rangkaian harmonis konsep-konsep unggulan dalam bidang *pattern recognition*. Sebagai salah satu metode *pattern recognition*, usia SVM terbilang masih relatif muda. Walaupun demikian, evaluasi kemampuannya dalam berbagai aplikasinya menempatkannya sebagai *state of the art* dalam pattern recognition, dan dewasa ini merupakan salah satu tema yang berkembang dengan pesat. SVM adalah metode *machine learning* yang bekerja atas prinsip *Structural Risk Minimization* (SRM) dengan tujuan menemukan *hyperplane* terbaik yang memisahkan dua buah class pada *input space*.

Berbeda strategi dengan *neural network* yang berusaha mencari *hyperplane* pemisah antar class, SVM berusaha menemukan *hyperplane* yang terbaik pada input space. Prinsip dasar SVM adalah *linear classifier*, dan selanjutnya dikembangkan agar dapat bekerja pada problem non-linear dengan memasukkan konsep kernel trick pada ruang kerja berdimensi tinggi.

Support Vector Machine (SVM) juga dikenal sebagai teknik pembelajaran mesin paling mutakhir setelah pembelajaran mesin sebelumnya yang dikenal neural network (NN). Kedua Metode tersebut menunjukkankeberhasilan dalam penggunaan

pengklasifikasian dan pengenalan pola. Proses pembelajaran dilakukan dengan menggunakan pasangan data input dan dataoutput sebagai sasaran yang diinginkan. Proses pembelajaran ini dikenal dengan *Supervised Learning*. Dari proses inilah akan diperoleh fungsi yang menggambarkan ketergantungan input dan outputnya. Selanjutnya, diharapkan fungsi yang dihasilkan dapat mengeneralisasi data dengan baik. Sehingga fungsi yang digunakan nantinya untuk data input diluar data training.

Kelebihan *support vector machine* adalah:

1. Efektif di ruang berdimensi tinggi
2. Masih efektif jika jumlah dimensi lebih besar dari jumlah sampelnya
3. Menggunakan subset dari titik latih pada fungsi keputusannya, sehingga memori yang digunakan lebih effisien.
4. Serbaguna, karena fungsi kernel yang berbeda dapat menentukan untuk pengambilan fungsi keputusan.

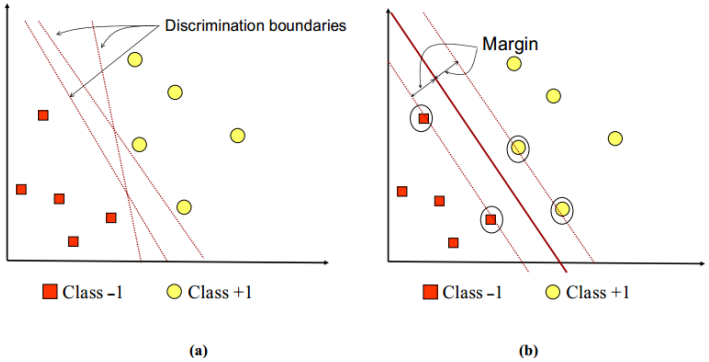
Kelemahan dari *support vector machine* adalah:

1. jika jumlah fitur lebih besar dari jumlah sempel, tidak bisa menerapkan fungsi kernel dan sangat penting menggunakan regularisasi.
2. SVM tidak secara langsung memberikan perkiraan probabilitas, probabilitas dihitung dengan menggunakan validasi.

2.7.1.1 *Support Vector Classification* (SVC)

SVM yang digunakan untuk melakukan klasifikasi disebut dengan *Support Vector Classification* (SVC). Untuk melakukan klasifikasi SVC menggunakan persamaan matematika. Konsep SVC dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah class pada *input space*. Gambar 2.6-a memperlihatkan beberapa pattern yang merupakan anggota dari dua buah class : +1 dan –1. Pattern

yang tergabung pada class –1 disimbolkan dengan warna merah (kotak), sedangkan pattern pada class +1, disimbolkan dengan warna kuning(lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.6-a.



Gambar 2.6 SVM Berusaha menemukan hyperplane terbaik yang memisahkan kedua class-1 dan +1

Hyperplane pemisah terbaik antara kedua class dapat ditemukan dengan mengukur margin *hyperplane* dan mencari titik maksimalnya. Margin adalah jarak antara *hyperplane* tersebut dengan pattern terdekat dari masing-masing class. Pattern yang paling dekat ini disebut sebagai *support vector*. Garis solid pada Gambar 2.6-b menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua class, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi hyperplane ini merupakan inti dari proses pembelajaran pada SVC.

Data yang tersedia dinotasikan sebagai $\vec{x}_i \in \mathcal{R}^d$ sedangkan label masing-masing dinotasikan $y_i \in \{-1, +1\}$ untuk $i = 1,2, \dots, l$, yang mana l adalah banyaknya data. Diasumsikan kedua

class –1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi d , yang didefinisikan,

$$\vec{w}.\vec{x}_i + b = 0 \quad (2.1)$$

$\vec{w}$ merupakan vektor dari pemisah kedua kelas, $\vec{x}_i$ merupakan data latih dan *b* adalah nilai bias. Nilai $\vec{w}$ didapatkan dari,

$$\vec{w} = \sum_i \alpha_i \vec{s_i} \quad , \quad i = 1,2, \dots n(support\ vector) \quad (2.2)$$

α_i merupakan variabel dari persamaan linear dan s_i adalah banyaknya vektor pembantu yang digunakan untuk membentuk pemisah kedua kelas.

Pattern $\vec{x}_i$ yang termasuk class –1 (sampel negatif) dapat dirumuskan sebagai pattern yang memenuhi pertidaksamaan,

$$\vec{w}.\vec{x}_i + b \leq -1 \quad (2.3)$$

sedangkan *pattern* $\vec{x}_i$ yang termasuk class +1 (sampel positif),

$$\vec{w}.\vec{x}_i + b \leq 1 \quad (2.4)$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara hyperplane dan titik terdekatnya. Hal ini dapat dirumuskan sebagai Quadratic Programming (QP) problem, yaitu mencari titik minimal persamaan (2.5), dengan memperhatikan Batasan persamaan (2.6).

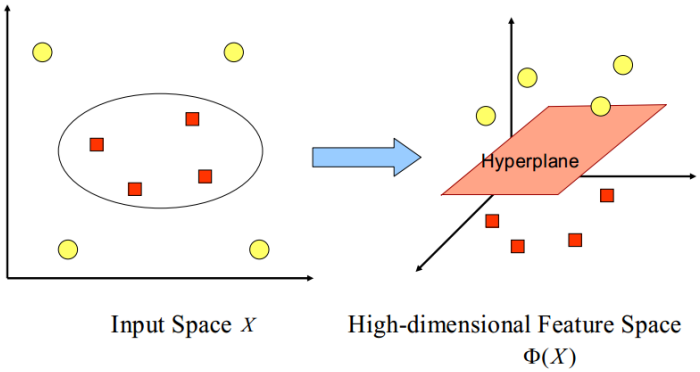
$$min_{\vec{w}}\ \tau(w) = \frac{1}{2}\|\vec{w}\|^2 \quad (2.5)$$

$$y_i(\vec{x}_i.\vec{w} + b) - 1 \geq 0\ , \forall i = 1,2, \dots, l \quad (2.6)$$

2.7.1.2 Kernel

Masalah dalam domain kehidupan nyata jarang sekali bersifat linier. Kebanyakan permasalahan yang muncul memiliki domain tak linier, permasalahan tak linier ini pada SVM dimodifikasi dengan memasukkan fungsi kernel. Dalam non linier

SVM, data $\vec{x}$ dipetakan oleh fungsi $\Phi(\vec{x})$ ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini *hyperplane* yang memisahkan kedual *class* tersebut dapat dikontruksikan. Ilustrasi dari konsep ini dapat dilihat pada Gambar 2.7 diperlihatkan bahwa pada gambar tersebut fungsi Φ memetakan data ke ruang vektor yang berdimensi lebih tinggi, sehingga kedua kelas dapat dipisahkan secara linier oleh sebuah hyperplane.



Gambar 2.7 Pemetaan fungsi kernel

Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik support vector, hanya bergantung pada dot product dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi, yaitu $\Phi(\vec{x}_i)\Phi(\vec{x}_j)$. Umumnya transformasi Φ ini sulit diketahui sehingga dapat digantikan fungsi kernel $K(\vec{x}_i, \vec{x}_j)$ yang mendefinisikan secara implisit dari transformasi Φ. Hal tersebut disebut *kernel Trick*, yang dirumuskan:

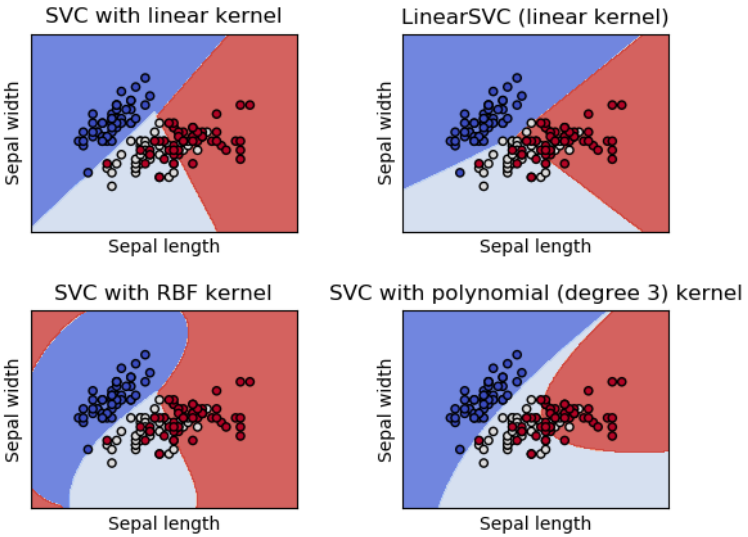
$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i)\Phi(\vec{x}_j)$$

Beberapa kernel yang digunakan pada SVM adalah :

Tabel 2.1 Kernel Pada SVM

Jenis Kernel	Definisi
Linear	$K(\bar{x}_i, \bar{x}_j) = \langle \bar{x}_i, \bar{x}_j \rangle$
Polynomial	$K(\bar{x}_i, \bar{x}_j) = (\gamma \langle \bar{x}_i, \bar{x}_j \rangle + r)^d$
Gaussian Radial Basic Function	$K(\bar{x}_i, \bar{x}_j) = e^{-\frac{1}{2\sigma^2}\|\bar{x}_i - \bar{x}_j\|^2}$
Sigmoid	$K(\bar{x}_i, \bar{x}_j) = \tanh(\gamma \langle \bar{x}_i, \bar{x}_j \rangle + \beta)$

Contoh dari hasil pengklasifikasin dengan menggunakan kernel bisa dilihat pada Gambar 2.8. Dimana pada Gambar 2.8 menunjukkan perbedaan support vektor antara masing-masing kernel.



Gambar 2.8 SVC dengan menggunakan kernel linear, RBF dan Polynomial

2.7.1.3 Multi Klasifikasi

Support Vector Machines (SVM) pada awalnya dirancang untuk klasifikasi biner. Mencari cara yang efektif untuk klasifikasi multi kelas masih merupakan isu penelitian yang sedang berlangsung. Saat ini ada dua tipe pendekatan untuk multi class SVM. Salah satunya adalah dengan membangun dan menggabungkan beberapa penggolong biner sedangkan yang lainnya adalah dengan langsung mempertimbangkan semua data dalam satu formulasi optimasi. Sampai saat ini masih belum ada perbandingan yang mencakup sebagian besar metode ini [13].

Perumusan untuk memecahkan masalah multi-kelas SVM dalam satu tahap memiliki variabel sebanding dengan jumlah kelas. Oleh karena itu, untuk metode SVM multi-kelas, beberapa klasifikasi biner harus dibangun atau diperlukan masalah optimasi

yang lebih besar. Oleh karena itu pada umumnya secara komputasi lebih sulit untuk memecahkan masalah *multiclass* daripada masalah biner dengan jumlah data yang sama. Hingga saat ini percobaan terbatas pada kumpulan data kecil [13].

Klasifikasi multiklas berarti tugas klasifikasi dengan lebih dari dua kelas, misalnya mengklasifikasi sekumpulan gambar buah yang mungkin jeruk, apel, atau pir. Klasifikasi multiklas membuat asumsi bahwa setiap sampel diberikan satu dan hanya satu label, buah bisa berupa apel atau pir tapi tidak keduanya pada saat bersamaan.

Klasifikasi multilabel menentukan masing-masing sampel satu set label target. Ini bisa dianggap sebagai ramalan properti dari data-point yang tidak saling eksklusif, seperti topik yang relevan untuk sebuah dokumen. Sebuah teks mungkin tentang agama, politik, keuangan atau pendidikan pada saat bersamaan atau tidak sama sekali.

2.7.1.4 Proses Klasifikasi

Klasifikasi yang digunakan pada penelitian ini menggunakan model klasifikasi SVM. Klasifikasi SVM termasuk dalam "Supervised Learning" dimana pada data pelatihan (training) disertai target, misalkan fungsi target yang memetakan setiap data/vektor (set fitur) X ke dalam satu dari sejumlah label kelas output yang tersedia. Tujuan penggunaan model SVM ini adalah untuk membangun model yang dapat menghasilkan output yang benar untuk suatu data input dalam hal ini untuk penklasifikasi data baru dengan tepat.

Proses klasifikasi dimana proses learning dan testing dilakukan menggunakan input data. Proses learning merupakan proses pemilihan model dan penentuan parameter data berdasarkan data pelatihan (training data), misalkan (X_train,y). Sedangkan

testing merupakan proses pengujian metode dengan menggunakan data penguji (testing data), misalkan (X_test) , sehingga diperoleh nilai estimasi untuk kapabilitas generalisasi dari model prediksi dari algoritma yang digunakan. Pada fit training , dilakukan proses pembangunan model dimana selama proses pelatihan diperlukan algoritma pelatihan salah satunya adalah SVM (Support Vector Machine). Pada akhirnya model yang dibangun pada saat pelatihan digunakan dalam memetakan setiap vektor /data input ke label kelas dari data baru yang belum diketahui label kelasnya sehingga mendapatkan label kelas keluaran yang benar.

2.8 Python

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Versi terakhir yang dikeluarkan CWI adalah 1.2. Tahun 1995, Guido pindah ke CNRI sambil terus melanjutkan pengembangan Python. Versi terakhir yang dikeluarkan adalah 1.6. Tahun 2000, Guido dan para pengembang inti Python pindah ke BeOpen.com yang merupakan sebuah perusahaan komersial dan membentuk BeOpen PythonLabs. Python 2.0 dikeluarkan oleh BeOpen. Setelah mengeluarkan Python 2.0, Guido dan beberapa anggota tim PythonLabs pindah ke DigitalCreations.

Saat ini pengembangan Python terus dilakukan oleh sekumpulan pemrogram yang dikoordinir Guido dan Python Software Foundation. Python Software Foundation adalah sebuah organisasi non-profit yang dibentuk sebagai pemegang hak cipta intelektual Python sejak Versi 2.1 dan dengan demikian mencegah Python dimiliki oleh perusahaan komersial. Saat ini distribusi Python sudah mencapai Versi 2.6.1 dan Versi 3.0.

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang

menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python mendukung multi paradigma pemrograman, namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

Bahasa python sangat menerapkan konsep pemrograman berorientasi objek. Dimana python menyediakan package yang dapat digunakan programmer, sehingga programmer tidak perlu membuat fungsi lagi dalam membuat aplikasi. Pada python fungsi yang dibuat dari awal dapat digunakan oleh semua orang sebagai referensi dengan mengimportkan package tersebut sebagi penerapan dari pemrograman berorientasi objek. Contoh perbandingan *coding* Java dengan Python:

Tabel 2.2 Perbandingan Bahasa java dan python

Java	Python
Package hallo;	print(‘Hello Wolrd’)
public class hallo{	

```
  public static void
main(String[]

  args) {

  System.out.print(“Hello
Wolrd”);

  }

}
```

Bahasa python lebih mudah dan effisien dalam pengkodean algoritma. Dalam contoh diatas bisa dilihat bahwa Bahasa java harus mendeklarasikan package dan class dari program untuk menjalankan program “Hello Wolrd”. Sedangkan pada python hanya menuliskan code print dan tanpa deklarasi class sudah bisa dijalankan.

Seperti bahasa-bahasa lainnya, kelebihan bahasa pemrograman Python menyebabkan perusahaan teknologi meliriknya untuk digunakan dalam mengembangkan platformnya. Kelebihan-kelebihan bahasa pemrograman Python tersebut antara lain adalah sebagai berikut:

1. Mudah Dipelajari

Mudah dipelajari sudah melekat sebagai salah satu kelebihan bahasa pemrograman Python diantara bahasa pemrograman lainnya. Bahasa ini memiliki sintaks-sintaks yang cukup sederhana dan mudah dimengerti. Bahasa ini merupakan bahasa yang sangat dinamis yang dibangun berdasarkan tingkat keterbacaan kode yang tinggi. Filososfi ini menjadikan bahasa pemrograman Python memiliki kelebihan seperti yang telah dijelaskan sebelumnya.

2. Mudah Digunakan

Bahasa pemrograman ini merupakan bahasa yang mudah untuk digunakan dalam mengembangkan sebuah produk, baik itu web, software, aplikasi web, maupun video game. Python memiliki kode tingkat paling tinggi, sehingga kode mudah dipahami, bahasa pemrograman ini memiliki library yang sangat banyak dan luas. Berbagai macam jenis library ini memuat sangat banyak perlengkapan dan fungsionalitas yang dangat luar biasa, sehingga kemudahan membangun program menjadi salah satu yang ditawarkan oleh bahasa pemrograman tersebut.

3. Mendukung *Internet of Things* Dengan Baik

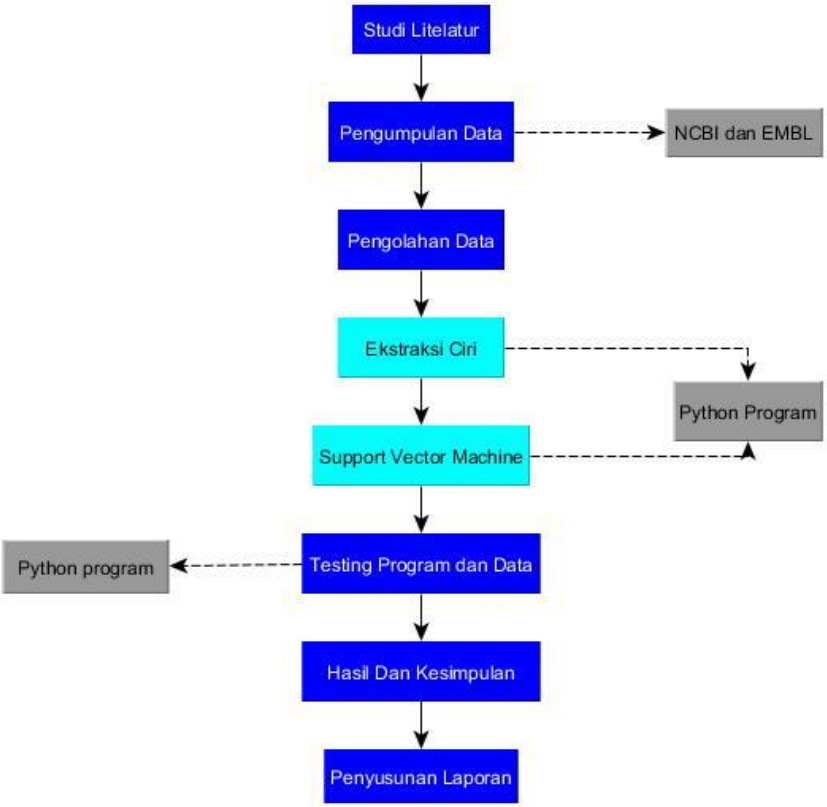
Merupakan bahasa yang mendukung ekosistem *Internet of Things* dengan sangat baik. Internet of Things sedang marak dibicarakan atau bahkan dikembangkan di kalangan maker. *Internet of Things* merupakan sebuah teknologi yang menghubungkan benda-benda di sekitar kita ke dalam sebuah jaring-jaring yang menghubungkan satu sama lain.

Teknologi yang mengusung semua benda dapat terhubung dalam satu jaringan internet ini tidak terlepas dari kebutuhan akan bahasa pemrograman dalam mengembangkan sistemnya. Dan bahasa pemrograman Python menawarkan dukungan yang sangat baik terhadap teknologi ini. Bahasa ini menjadi sangat populer, karena banyak sistem yang mengusung *Internet of Things* menggunakan bahasa ini. Terdapat berbagai macam board yang digunakan menjalankan sistem Internet of Things menggunakan bahasa pemrograman ini sebagai basisnya, termasuk di dalamnya adalah Raspberry Pi.

Halaman ini sengaja dikosongkan

BAB III
METODOLOGI PENELITIAN

Pada bab ini menjelaskan langkah-langkah dalam mengerjakan penelitian tugas akhir ini. Dimana langkah-langkah dan metode yang digunakan pada penelitian ini dapat dilihat pada Gambar 3.1



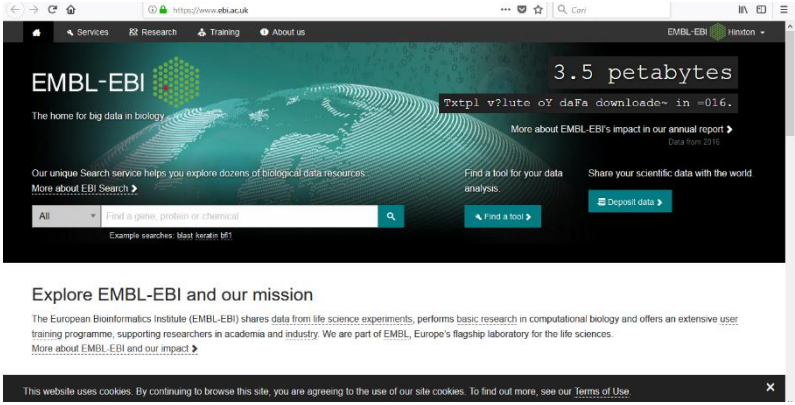
Gambar 3.1 Flowchart metodologi penelitian

3.1 Studi Litelatur

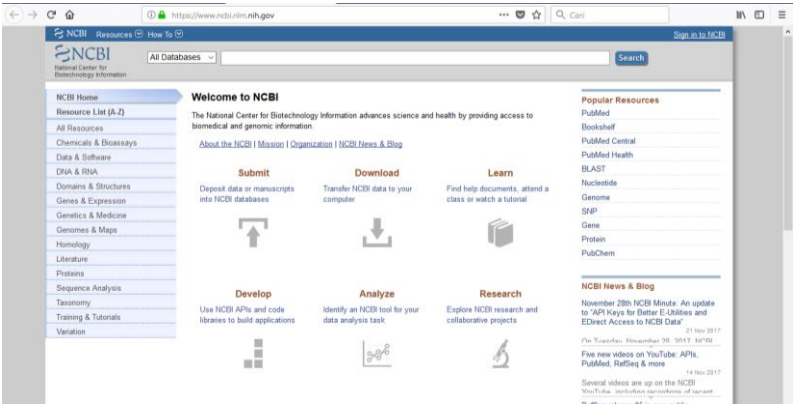
Studi litelatur merupakan langkah awal untuk mencari sumber-sumber teori yang bisa dijadikan referensi dalam mengerjakan penelitian Tugas Akhir ini. Studi litelatur digunakan untuk mengidentifikasi permasalahan yang dijadikan objek penelitian yang didapatkan dari penelitian terdahulu seperti Tugas Akhir, Tesis, Disertasi, Journal internasional dll, serta berupa teori dasar ilmiah dari buku-buku maupun artikel ilmiah yang ada di internet.

3.2 Pengumpulan Data

Pengumpulan data merupakan tahap mencari dan mengumpulkan data yang diperlukan untuk bahan penelitian tersebut. Dalam penelitian ini data berupa data sekunder dari sekuens DNA/RNA yang disediakan NCBI (*National Center of Biotechnology Information*) dan EMBL (*European Molekuler Biotecnology Laboratory*). Setelah pengambilan data, data akan dipilah menjadi dua, yaitu data trining dan testing.



Gambar 3.2 Website EMBL



Gambar 3.3 Website NCBI

3.3 Pengolahan Data

Setelah mengumpulkan data, data akan diolah untuk menjadikan suatu informasi yang diingikan. Pada tahap ini, data akan diolah sebanyak dua kali. Dari data berupa sekuens DNA/RNA yang berupa format string akan di ekstrak terlebih dahulu menjadi numerik, setelah itu dilakukan pengklasifikasian menggunakan *support vector machine*. Pada tahap ini untuk mempermudah dilakukan pemrograman dengan Bahasa phyton Versi 3.6 dan juga Anaconda sebagai IDE.

3.3.1 Ekstraksi Ciri

Ekstraksi ciri digunakan untuk menentukan ciri dari setiap data yang digunakan. Pada penelitian ini digunakan ekstraksi ciri rantai markov dan juga *Principle Component Analysis* (PCA) untuk mereduksi ekstraksi ciri markov dan sebagai visualisasi data.

a. Rantai Markov

Pada ekstraksi ciri data DNA/RNA yang berupa tipe data string akan di ekstrak menjadi data numerik. Dengan menggunakan probabilitas rantai markov. Pada penelitian ini rantai markov orde 2 digunakan dengan mendapatkan matriks transisi sebanyak 64 komponen. Berpacu pada penelitian sebelumnya (Fajar ML,2013), markov orde 2 lebih memberikan ciri yang spesifik jika dibandingkan rantai markov orde 1. Pada markov orde 2, setiap basa nitrogen tertentu dipengaruhi oleh dua basah nitrogen selanjutnya. Contoh, kemungkinan basah nitrogen A (adenin) jika 2 basah nitrogen selanjutnya A (adenin) dan A (adenin) , sehingga bisa dituliskan P(AA|A). Bentuk dari matriks markov orde 2 seperti Gambar 3.4.

$$P = \begin{bmatrix} P(AA|A) & P(AA|C) & P(AA|G) & P(AA|T) \\ P(AC|A) & P(AC|C) & P(AC|G) & P(AC|T) \\ P(AG|A) & P(AG|C) & P(AG|G) & P(AG|T) \\ P(AT|A) & P(AT|C) & P(AT|G) & P(AT|T) \\ P(CA|A) & P(CA|C) & P(CA|G) & P(CA|T) \\ P(CC|A) & P(CC|C) & P(CC|G) & P(CC|T) \\ P(CG|A) & P(CG|C) & P(CG|G) & P(CG|T) \\ P(CT|A) & P(CT|C) & P(CT|G) & P(CT|T) \\ P(GA|A) & P(GA|C) & P(GA|G) & P(GA|T) \\ P(GC|A) & P(GC|C) & P(GC|G) & P(GC|T) \\ P(GG|A) & P(GG|C) & P(GG|G) & P(GG|T) \\ P(GT|A) & P(GT|C) & P(GT|G) & P(GT|T) \\ P(TA|A) & P(TA|C) & P(TA|G) & P(TA|T) \\ P(TC|A) & P(TC|C) & P(TC|G) & P(TC|T) \\ P(TG|A) & P(TG|C) & P(TG|G) & P(TG|T) \\ P(TT|A) & P(TT|C) & P(TT|G) & P(TT|T) \end{bmatrix}^T$$

Gambar 3.4 Matriks transisi rantai markov orde 2

Hasil dari ekstraksi ciri tersebut akan dijadikan data training dan data testing dari proses selanjutnya. Data testing digunakan untuk membentuk support vektornya dan data trining digunakan untuk mengetest kevalidan dari program tersebut.

b. *Principle Component Analysis*

PCA digunakan untuk mereduksi data training yang sudah di cirikan dengan menggunakan rantai markov. Dimana setiap data dengan 64 ciri akan direduksi hanya menjadi 2 ciri untuk memunculkan data dalam bidang 2 dimensi. Serta direduksi menjadi 3 ciri untuk memvisualisasikan data training pada ruang 3 dimensi. Untuk mengekstraksi ciri dengan PCA, penelitian ini menggunakan library python yang tersedia sehingga sudah terjamin penerapan PCA pada dataset.

3.3.2 Klasifikasi Support Vector Machine

Setelah mendapatkan ekstraksi ciri dari semua data uji. Data training akan dijadikan data klasifikasi untuk membentuk support vektornya. Pada proses klasifikasi ini, data kan diklasifikasikan menjadi 5 klasifikasi, yaitu acute lympoblastic leukemia (ALL), acute myeloid leukemia (AML), chronic lymphocytic leukemia (CLL) dan chronic myeloid leukemia (CML). Dengan memanfaatkan kernel memudahkan dalam mengklasifikasikan. Beberapa kernel seperti linier, RBF, polynomial. Pada proses pengklasifikasian ini data training digunakan untuk membentuk support vektornya.

3.4 Pengujian

Pada tahap ini, awalnya program akan ditest dengan data testing, untuk memastikan kevalidan algoritma program. Data testing yang digunakan merupakan data yang berbeda yang digunakan pada data trining. Karena fungsi dari data testing untuk

memvalidasi metode yang digunakan yaitu Rantai markov dan SVC (*Support Vector Classification*). Dengan pengujian didapatkan seberapa besar kemungkinan program tersebut memprediksi dengan benar data yang diolah.

3.5 Hasil dan Kesimpulan

Setelah program diuji dan berjalan dengan baik. Dilakukan analisis output dari program tersebut. Analisa output dilakukan untuk mengevaluasi metode yang digunakan serta mengevaluasi keberhasilan program dalam membaca keakuratan data. Sehingga pada proses ini bisa disimpulkan bagaimana kinerja metode rantai markov dan juga *support vector machine* dalam melakukan tugas.

3.6 Penyusunan Laporan Tugas Akhir

Setelah semua aspek metodologi selesai sampai dengan kesimpulan, akan dibuat sebuah laporan Tugas Akhir. Penyusunan laporan seperti yang tertera pada BAB I. Pendahuluan di subbab Sistematika Penulisan Tugas Akhir.

BAB IV
PERANCANGAN DAN PROSES SISTEM

Pada bab ini akan dijelaskan proses-proses pada perancangan sistem. Perangcangan sistem memuat rekayasa perangkat lunak mulai dari analisis kebutuhan hingga data flow diagram program.

4.1 Deskripsi Program

Perangkat lunak merupakan bagian dari sistem komputer yang tidak berwujud. Perangkat lunak bisa berbentuk aplikasi, driver, sistem database, dll. Perangkat lunak pada penelitian ini berupa penerjemah dari sifat sekuens DNA/RNA leukemia dengan mengkodekan perilaku dari empat basa nitrogen yang ada (adenin, sitosin, guanin dan timin). Sekuens leukemia tersebut dibaca oleh komputer dengan tipe data string. Sehingga metode rantai markov mengkonfersi string tersebut ke bentuk numerik dengan menerjemahkan urutan ciri basa nitrogen sekuens terhadap probabilitas dari kemunculan basa nitrogen. Output dari program ini berupa grafik dan prediksi dari kecenderungan sekuens. Sehingga dengan melihat grafik dan kecenderungan pada program, peneliti dapat memvalidasi. Dengan menggunakan parameter-parameter dari SVM yang diubah-ubah, diharapkan mendapatkan parameter yang sesuai terhadap prediksi leukemia baru. Program ini menggunakan bahasa pemrograman python dan menggunakan *library* (*scikit learn*) yang merupakan modul *machine learning* yang disediakan *platform.*

4.2 Analisis Kebutuhan Sistem

Dalam rekayasa sistem dan rekayasa perangkat lunak, analisis kebutuhan mencakup pekerjaan-pekerjaan penentuan kebutuhan atau kondisi yang harus dipenuhi dalam suatu program, yang mempertimbangkan berbagai kebutuhan yang bersinggungan antar berbagai pemangku kepentingan. Kebutuhan dari hasil analisis ini

harus dapat dilaksanakan, diukur, diuji, terkait dengan kebutuhan yang teridentifikasi, serta didefinisikan sampai tingkat detail yang memadai untuk desain sistem.

4.2.1 Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak sangat mendukung dalam menjalankan suatu program. Pada penelitian ini program membutuhkan beberapa perangkat lunak, yaitu:

a. Sistem Operasi : Windows XP/7/8/10
b. Microsoft Office
c. Python 3.6
d. Anaconda3 5.0.1
e. Spyder (python 3.6)
f. Browser : Chrome/Mozila Firefox
g. yEd-3.17.2 (Graph Editor)

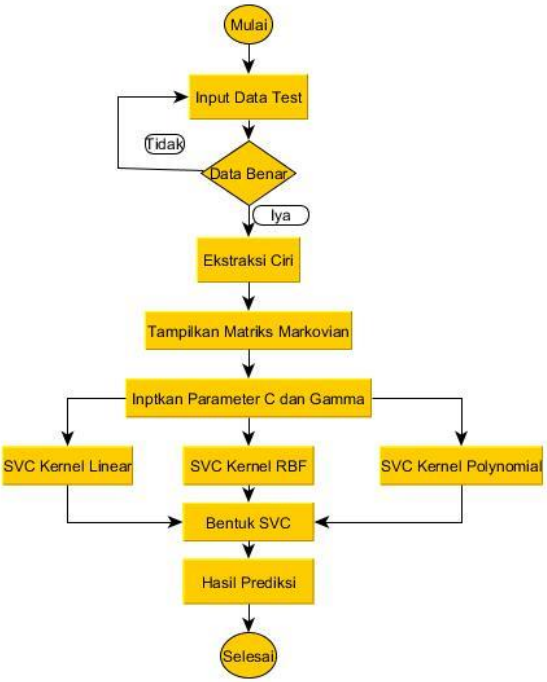
4.2.2 Analisis Kebutuhan Perangkat Keras

Program memerlukan alat-alat perangkat keras sebagai tempat menjalankan suatu program. Perangkat keras mendukung performa program, beberapa yang dibutuhkan yaitu:

a. Processor intel i3/i5 atau lebih tinggi
b. Random Acces Memori (RAM DDR3) 2 gb atau lebih
c. Router
d. Printer
e. Hardisk
f. LAN (internet connection)

4.3 Diagram Aktifitas

Diagram aktifitas merupakan alur dari penggunaan suatu program. Diagram ini memuat bagaimana program dijalankan, inputan program, proses suatu inputan, hingga ke output dari program. Runtutan dari penggunaan program dapat dilihat pada Gambar 4.1.



Gambar 4.1 Diagram aktifitas dari program

Ketika program dijalankan maka yang pertama akan menginputkan data test. Data test akan di validasi bentuk dari data,dimana akan dihilangkan keterangan-keterangan yang tidak dibutuhkana dalam proses pengolahan sekuens DNA/RNA leukemia. Validasi data juga menghilangkan spasi skuens dari data asli. Jika data mengalami banyak error dan error kode pada sekuens maka akan dilakukan input ulang data dengan mengganti data baru.

Setelah data test benar, maka program akan memproses data ke dalam ekstraksi ciri. Ekstraksi ciri disini membentuk 64 ciri dari data test dengan menggunakan rantai markov orde 2. Program akan menampilkan hasil ke 64 ciri dengan membentuk matriks markovian. Proses selanjutnya yaitu memasukkan nilai-nilai

parameter dari *support vector classification* yaitu nilai *C* dan *gamma*. Program akan memproses data kedalam masing-masing kernel yang digunakan dengan parameter yang diinginkan oleh user. Setelah itu program akan memunculkan hasil dari SVC untuk setiap kernel dan akan memprediksi data test tersebut masuk kedalam kelas dari jenis leukemia yang mana (ALL, AML, CLL dan CML) dengan ketentuan parameter-parameter SVC.

4.4 Data Flow Diagram

Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem, agar bisa membantu penggunanya untuk memahami sistem secara logika, tersruktur dan jelas. DFD mempunya 3 level, dimana level menandakan kedetilan dari apa yang ada didalam program tersebut. Level-level dari DFD adalah :

1. Diagram Konteks : menggambarkan satu lingkaran besar yang dapat mewakili seluruh proses yang terdapat di dalam suatu sistem. Merupakan tingkatan tertinggi dalam DFD dan biasanya diberi nomor 0 (nol). Semua entitas eksternal yang ditunjukkan pada diagram konteks berikut aliran-aliran data utama menuju dan dari sistem. Diagram ini sama sekali tidak memuat penyimpanan data dan tampak sederhana untuk diciptakan.

2. Diagram Nol (diagram level-1) : merupakan satu lingkaran besar yang mewakili lingkaran-lingkaran kecil yang ada di dalamnya. Merupakan pemecahan dari diagram Konteks ke diagram Nol. di dalam diagram ini memuat penyimpanan data.

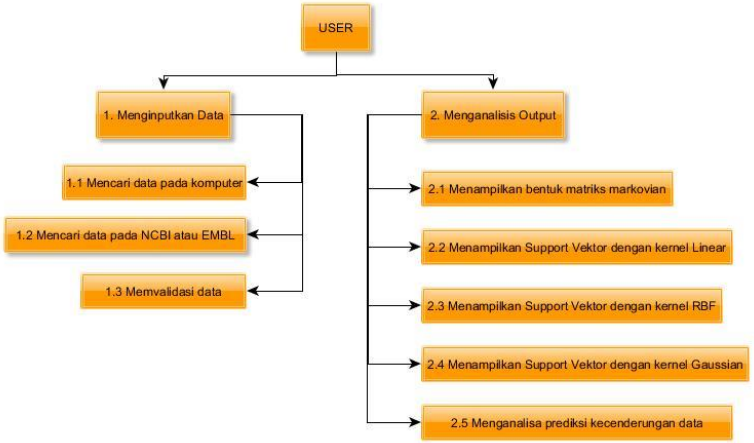
3. Diagram Rinci : merupakan diagram yang menguraikan proses apa yang ada dalam diagram Nol.

Data Flow Diagram (DFD) merupakan alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi.

DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

4.4.1 Data Flow Diagram Level 1



Gambar 4.2 DFD level 1 untuk program prediksi leukemia

Proses-proses yang dilakukan user ada dua macam, yaitu:
1. Menginputkan data

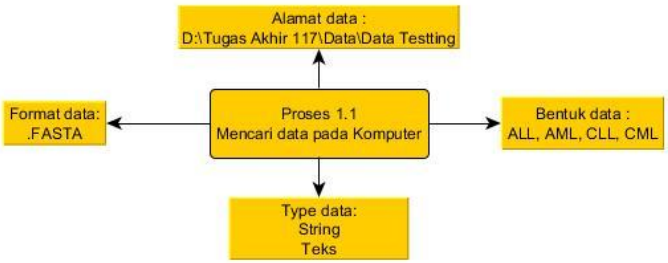
Pertama user menginputkan data DNA/RNA leukemia pada program. Pada proses 1.1 data test dapat diambil dari komputer karena sudah tersedia sebelumnya, sehingga user hanya mencari alamat pada komputer (D:\Tugas Akhir 117\Data\Data Testting). Pada proses 1.2 data test juga bisa diambil dari NCBI atupun EMBL, dengan mengakses langsung alamat web kedua sumber data. Pada proses 1.3 data yang diinputkan harus divalidasi terhadap standar yang bisa digunakan dalam proses program:
a. tidak ada keterangan dari sekuens
b. data test tanpa spasi
c. tidak ada error data berupa error coding sekuens
d. data yang digunakan hanya sekuens dari leukemia berupa basa nitrogen (A,C,G dan T)
2. Menganalisis output

Pada proses ini user akan memproses data test untuk menampilkan informasi yang akan didapatkan. Pada proses 2.1 program akan mengekstraksi ciri dari data test. Hasilnya akan ditampilkan oleh program berupa matriks markovian. Pada proses 2.2-2.4 program akan menampilkan grafik dari SVC dengan parameter-parameter yang diinputkan user. Proses terakhir (proses 2.5) program akan memprediksi data test tersebut masuk dalam kategori dari salah satu kelas leukemia.

4.4.2 DFD Level 2

Data flow diagram level 2 merupakan pengembangan dari data flow diagram level 1. Dimana pada DFD level 2 keterangan data akan lebih diperdetail informasinya dari pada DFD level 1. Pada DFD level 2 proses 1.1-1.3 dan proses 2.1-2.5 akan diinformasikan lebih detail lagi.

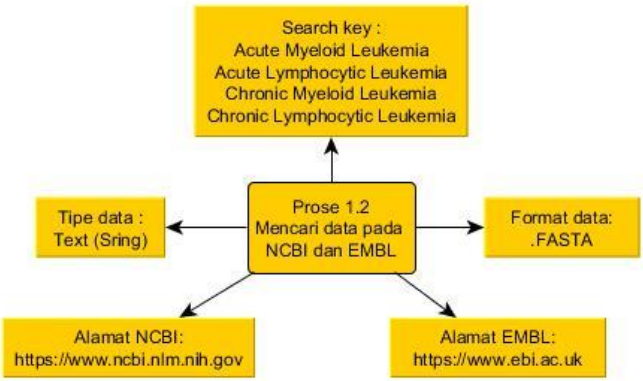
❖ Proses 1.1-Mencari Data Pada Komputer



Gambar 4.3 Data flow diagram level 2 proses 1.1

Pada proses 1.1 user akan mencari data pada komputer dengan alamat data seperti pada gambar. Nantinya user ambil salah satu data yang terdiri dari jenis-jenis leukemia (ALL, AML, CLL dan CML) yang ada pada tempat tersebut untuk dijadikan data tes. Format data berupa .FASTA yang merupakan format untuk bioinformasi dari NCBI dan EMBL yang bisa dibuka melalui notepad. Ketika dibuka di notepad tipe data yang terlihat berupa teks (String) dengan urutan basa nitrogen dari jenis leukemia.

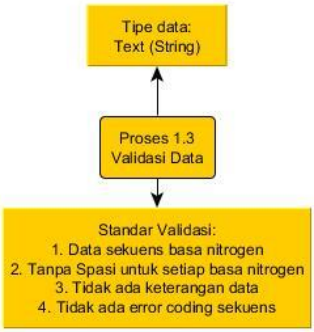
❖ Proses 1.2-Mencari data pada NCBI dan EMBL



Gambar 4.4 DFD Level 2 proses 1.2

Proses 1.2 ini data test diambil dari web langsung, NCBI atau EMBL. Dimana data bisa dicari dengan kata kunci *acute myeloid leukemia* untuk jenis AML, *acute lymphocytic leukemia* untuk jenis ALL, *chronic myeloid leukemia* untuk jenis CML dan *chronic lymphocytic leukemia* untuk jenis CLL. Data pada web NCBI dan EMBL dapat diunduh dengan format data .FASTA, yaitu format khusus pada bioinformasi.

❖Proses 1.3-Validasi Data



Gambar 4.5 DFD Level 2 proses 1.3

Pada validasi data, dengan manual user memvalidasi apakah data yang masuk sudah benar. Jika data yang diinputkan salah maka user harus memasukkan ulang data dan juga membenarkan data. Validasi dilakukan dengan memperhatikan standar yang dibutuhkan program. Jika sudah benar maka data akan diinputkan lagi dan divalidasi lagi apakah data sudah sesuai dengan apa yang diingkan untuk proses selanjutnya. Proses validasi ini dilakukan agar proses pembacaan program tidak mengalami gangguan dan error pada running program. Pada gambar 4.6 dan 4.7 mencontohkan data yang benar dan yang salah.

```
>ENA|AB464288|AB464288.1 Synthetic construct DNA, clone:
pF1KB9678, Homo sapiens TAL2 gene for T-cell acute lymphocytic
leukemia 2, without stop codon, in Flexi system.

GCGATCGCCATGACCAGGAAGATCTTCACAAATACCAGGGAG
CGGTGGAGGCAGCAGAATGTCAACAGCGCCTTTGCCAAGCTG

AGGAAGCTCATCCCCACTCACCCTCCAGACAAAAAGCTGAGC
AAAAATGAAACGCTTCGCCTGGCAATGAGGTATATCAACTTCT
TGGTCAAGGTCTTGGGGGAGCAAAGCCTGCAACAAACGGGAG

TGGCTGCTCAGGGGAACATTCTGGGGCTCTTCCCTCAAGGACC
CCACCTGCCAGGCCTGGAGGACAGAACTCTGCTTGAGAACTAC

CAGGTTCCTTCACCTGGTCCAAGCCACCACATTCCTGTTTAAAC
```

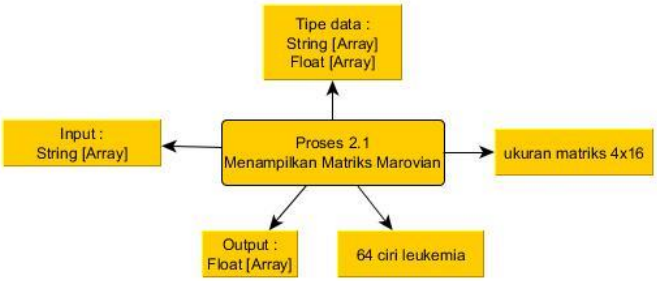
Gambar 4.6 Data belum divalidasi

Data diatas merupakan data yang belum disesuaikan dengan struktur yang diinginkan program. Pada Gambar 4.6 akan disesuaikan terlebih dahulu dengan menghilangkan title dari data dan menghilangkan spasi pada basa nitrogen. Sehingga hanya tersisa sekuens saja seperti Gambar 4.6. Setelah data sesuai dengan format program, barulah data bisa diproses dan dibaca oleh program.

```
GCGATCGCCATGACCAGGAAGATCTTCACAAATACCAGGGAG
CGGTGGAGGCAGCAGAATGTCAACAGCGCCTTTGCCAAGCTG
AGGAAGCTCATCCCCACTCACCCTCCAGACAAAAAGCTGAGC
AAAAATGAAACGCTTCGCCTGGCAATGAGGTATATCAACTTC
TTGGTCAAGGTCTTGGGGGAGCAAAGCCTGCAACAAACGGGA
GTGGCTGCTCAGGGGAACATTCTGGGGCTCTTCCCTCAAGGA
CCCCACCTGCCAGGCCTGGAGGACAGAACTCTGCTTGAGAAC
TACCAGGTTCCTTCACCTGGTCCAAGCCACCACATTCCTGTTT
AAAC
```

Gambar 4.7 Data setelah disesuaikan divalidasi

❖ Proses 2.1-Menampilkan Bentuk Matriks Markovian



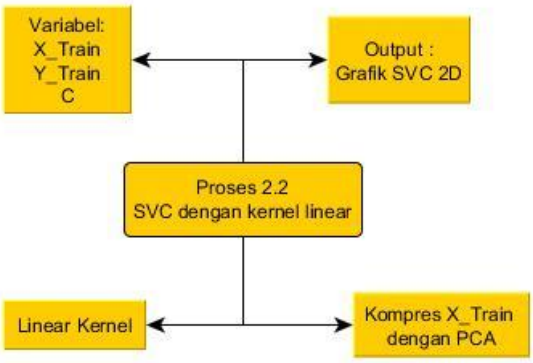
Gambar 4.8 DFD level 2 proses 2.1

Pada tahap proses, data akan dikonversi terlebih dahulu dengan metode markovian pada Bahasa pemrograman python. Data string dari DNA/RNA leukemia berupa text sekuens diolah dalam bentuk list terlebih dahulu. Dari list akan dihitung panjang sekuens dan juga kemunculan dari suatu basa nitrogen dengan dua basa nitrogen sebelumnya. Setelah mendapatkan jumlahnya, program akan menghitung probabilitas untuk setiap basa nitrogen tersebut. Lalu akan disusun menjadi matriks berukuran 4x16. Contoh dari matriks dapat dilihat pada Gambar 4.8.

```
Matrix Markovian =
[ 0.01169591  0.04481793  0.02662722  0.0078125 ]
[ 0.04093567  0.03921569  0.05325444  0.03125   ]
[ 0.0877193   0.07002801  0.09171598  0.0078125 ]
[ 0.02339181  0.02240896  0.0295858   0.0390625 ]
[ 0.06432749  0.07282913  0.05621302  0.0546875 ]
[ 0.0877193   0.15406162  0.15976331  0.140625  ]
[ 0.06432749  0.10644258  0.11242604  0.0625    ]
[ 0.03508772  0.06442577  0.06508876  0.046875  ]
[ 0.09356725  0.04481793  0.06804734  0.109375  ]
[ 0.20467836  0.10644258  0.12130178  0.1484375 ]
[ 0.0877193   0.10644258  0.1035503   0.1796875 ]
[ 0.03508772  0.00840336  0.03550296  0.0390625 ]
[ 0.00584795  0.0140056   0.00591716  0.0234375 ]
[ 0.05263158  0.05042017  0.0147929   0.0546875 ]
[ 0.0994152   0.07002801  0.0443787   0.03125   ]
[ 0.00584795  0.02521008  0.01183432  0.0234375 ]
```

Gambar 4.9 Matriks 4x16 Rantai Markov orde 2

❖ Proses 2.2-Support Vektor dengan Kernel Linear



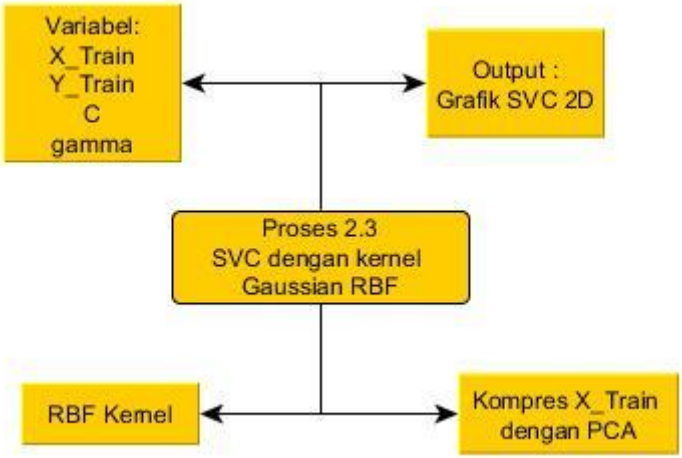
Gambar 4.10 DFD level 2 proses 2.2

Pada proses 2.2 pengklasifikasian dilakukan pada data training dengan SVC menggunakan kernel linear. Data trining (X_Train) yang berukuran 64x40 dimana 64 adalah banyaknya ciri dan 40 adalah banyaknya data training. Data tersebut akan diplotkan kedalam grafik 2 dimensi, sehingga dibutuhkan kompresi data. Kompresi data yang digunakan dengan metode *principle component analysis* (PCA) yang akan merubah data dari ukuran 64x40 menjadi 2x40. Sehingga bentuk data dengan matriks 2x40 ini bisa divisualisasikan kedalam grafik 2 dimensi. Nantinya pada proses training data dengan kernel linear, *hyperplane* yang dihasilkan berupa garis lurus (linear). Bentuk dari *hyperplane* tergantung dari parameter yang dimasukkan kedalam program.

```
# Membuat klasifikasi dengan kernel linear pada akut dan cronis
svm = SVC(kernel='linear', C=int(C))
svm.fit(X_xor, y_xor)
# Visualisasi klasifikasi
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Linear")
print("==============================")
plt.show()
```

Gambar 4.11 Code pembentuk SVC
dengan kernel linear

❖ Proses 2.3-Support Vektor dengan Kernel RBF



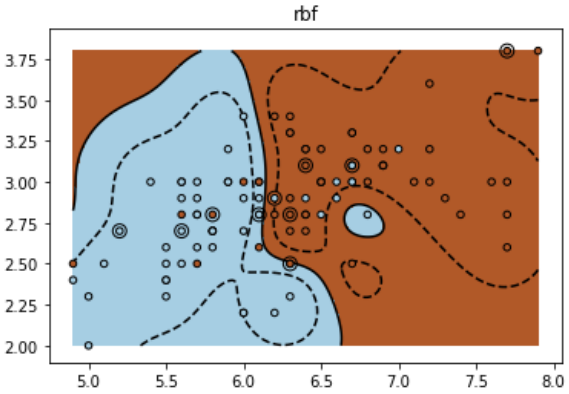
Gambar 4.12 DFD level 2 Proses 2.3

Mirip dengan proses pada penggunaan kernel linear, pada kernel Gaussian RBF (*Radial Basis Function*) hanya mengganti fungsi pada pemanggilan kernel. Data training akan ditrnasformasikan menjadi fungsi Gaussian. Pada kernel rbf, variable parameter dari SVC ada dua yaitu *C* dan *gamma*. Data training yang berukuran besar dimensinya dikompres terlebih dahulu sehingga nantinya data bisa divisualisasikan pada grafik 2 dimendi dengan tidak menghilangkan ciri dari setiap data. Code pemebentuk SVC dengan kernel rbf pada grafik dapat dilihat pada Gambar 4.13.

```
# Membuat klasifikasi dengan kernel RBF pada akut dan cronis
svm = SVC(kernel='rbf', gamma=int(Gamma), C=int(C))
svm.fit(X_xor, y_xor)
# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel RBF")
print("================================")
plt.show()
```

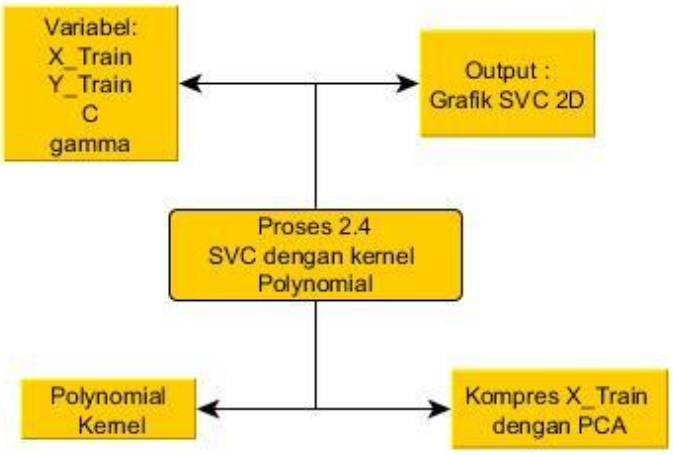
Gambar 4.13 Code pembentuk SVC
dengan kernel rbf

Hasil dari bentuk klasifikasi dengan kernel RBF biasa dilihat pada Gambar 4.14.



Gambar 4.14 Contoh hasil dari SVC dengan Kernel RBF

❖ Proses 2.4-Support Vektor dengan Kernel Polynomial



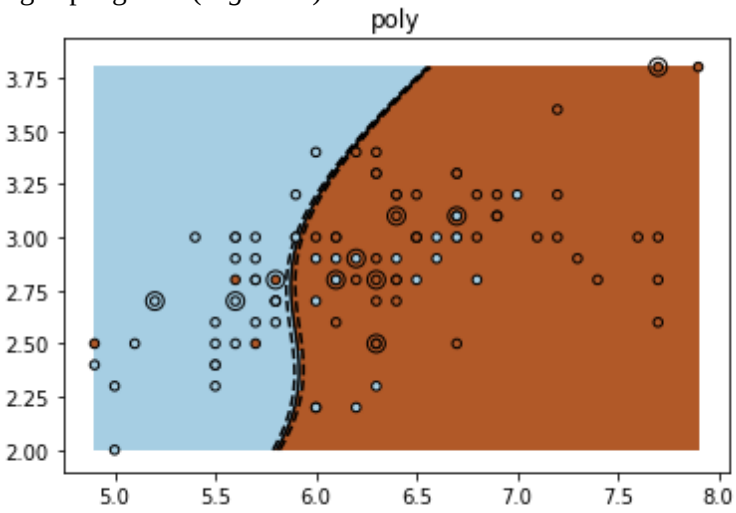
Gambar 4.15 DFD level 2 proses 2.4

Proses 2.4 merupakan proses dimana penggunaan kernel polynomial pada SVC. Sama dengan kernel RBF, pada kernel polynomial terdapat 2 parameter untuk pembentuk kelas, yaitu *C* dan *gamma*. Code pada program python dapat dilihat pada Gambar 2.16.

```
# Membuat klasifikasi dengan kernel polynomial pada akut dan cronis
svm = SVC(kernel='poly', degree=3, gamma=int(Gamma), C=int(C))
svm.fit(X_xor, y_xor)
# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Polynomial")
print("==============================")
plt.show()
```

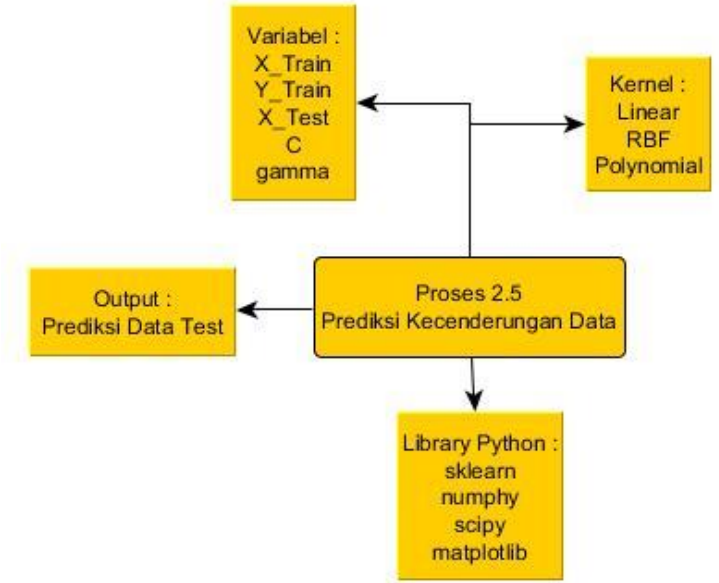
Gambar 4.16 Code pembentuk SVC dengan kernel polynomial

Contoh hasil dari penggunaan kernel polynomial pada SVC seperti pada Gambar 2.17. Dimana polynomial yang digunakan dengan pangkat 3 (*degree*=3).



Gambar 4.17 Contoh hasil dari SVC dengan Kernel Polynomial

❖ Proses 2.5-Hasil Prediksi DNA/RNA



Gambar 4.18 DFD level 2 proses 2.5

Pada prosses yang terakhir program akan memprediksi kedekatan ciri dari DNA/RNA leukemia test terhadap data training. Data test merupakan validasi dari metode support vektor classification. Dengan adanya data test maka bisa tarik kesimpulan apakah program dengan markov dan SVM ini bisa diterapkan. Dan juga dapat diuji pada parameter berapa performa SVC dapat bekerja dangan baik. Pada proses ini output program berupa teks dengan tipe prediksi dan juga keterangan dari prediksi. Seperti ditunjukkan Gambar 4.19. yang merupakan output dari hasil prediksi dari program ini.

```
=====PREDIKSI DATA LEUKEMIA TES PADA KE-4 KELAS=====
=====================================================

================================
====SVC Dengan Kernel Linear====
================================
Prediksi dari DNA/RNA tersebut adalah jenis leukemia AML
ACUTE MYELOID LEUKEMIA

==================================
===SVC Dengan Kernel Gaussian RBF===
==================================
Prediksi dari DNA/RNA tersebut adalah jenis leukemia AML
ACUTE MYELOID LEUKEMIA

================================
===SVC Dengan Kernel Polynomial===
================================
Prediksi dari DNA/RNA tersebut adalah jenis leukemia AML
ACUTE MYELOID LEUKEMIA
```

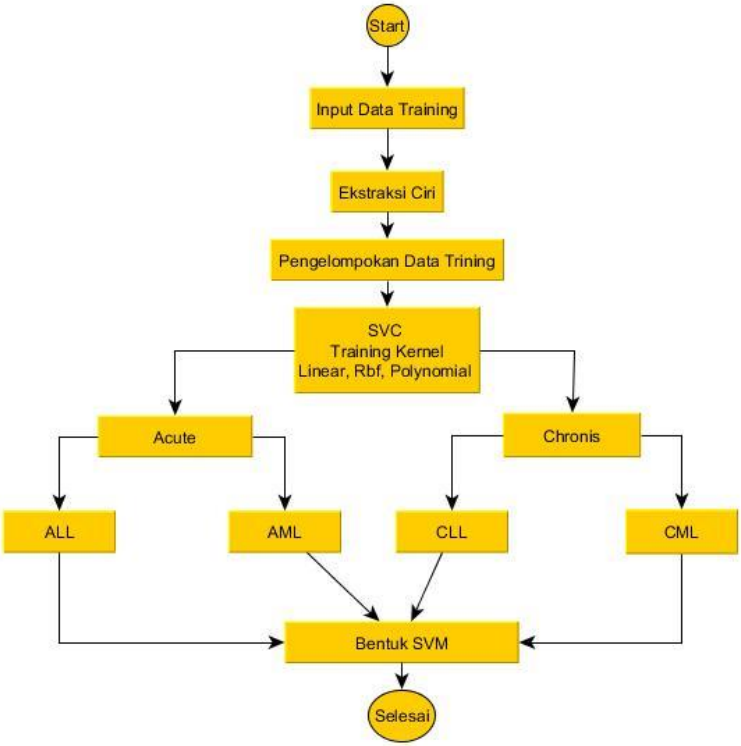
Gambar 4.19 Output dari prediksi program

Prediksi dari metode *machine learning* tidak 100% benar. Dengan mengandalkan data trining dan testing dapat dicari berapa persen keberhasilan program dalam menyelesaikan masalah. Semakin banyak data trining dan testing. Semakin akurat dan sensitive hasil dari penelitian. Dengan membandingkan penggunaan kernel pula bisa membuat prediksi dari program berbeda, karena setiap kernel memiliki support vektor yang berbeda. Sehingga memungkinkan terjadinya perbedaan prediksi dari setiap kernel. Maka dari pada itu testing program akan memberikan kualitas persentasi keberhasilan program.

4.5 Proses Pengolahan Data

Proses pengolahan data merupakan proses dari pengolahan data training. Data trining dari awal (mentah) diproses dalam program sehingga membentuk vektor pembantu dalam menunjukkan kelas-kelas dari leukemia. Dalam proses pengolahan data pada program, menggunakan konsep desain waterfall. Pada desain waterfall ini akan menunjukkan bagaimana data training awal lalu bagaimana proses klasifikasi hingga akhirnya menjadi

empat kelas dan bagaimana pembentukan SVC terhadap masing-masing kernel yang digunakan. Gambar 4.20 menunjukkan desain waterfall dari proses program.



Gambar 4.20 Desain Waterfall Program

4.5.1 Input Data

Input data berupa proses memasukkan data pada program. Data yang dimasukkan adalah data training sekuens DNA/RNA leukemia. Data tersebut yang sudah diedit dari aslinya yang diambil di labolatorium bioinformasi NCBI dan EMBL. Contoh dari data yang asali seperti pada Gambar 4.21. dimana masih ada keterangan jenis data dan panjang sekuens.

```
>ENA|X90979|X90979.1 H.sapiens mRNA for an acute myeloid leukaemia
protein (346bp)

CACAAGTTGGGTAGCCTGGCAGTGTCAGAAGTCTGAACCCAGCAT
AGTGGTCAGCAGGCAGGACGAATCACACTGAATGCAAACCACAGG
GTTTCGCAGCGTGGTGAGCATCACCCCACCACAGCCAAGGCGGCG
CTGGCTTTTTTTTTTTTTTTAATCTTTAACAATTTGAATATTTGTTT
TTACAAAGGTAAAAGAAATCATTGAGTCCCCCGCCTTCAGAAGAG
GGTGCATTTTCAGGAGGAAGCGATGGCTTCAGACAGCATATTTGA
GTCATTTCCTTCGTACCCACAGTGCTTCATGAGAGAATGCATACTT
GGAATGAATCCTTCTAGAGACGTCCAC
```

Gambar 4.21 Data Asli dari NCBI/EMBL dengan format .FASTA

Bisa dilihat dari gambar bahwa data tersebut berupa mRNA yang membentuk protein dari Acute Myeloid Leukemia (AML) dengan panjang 346bp. Dari data tersebut yang akan digunakan hanya sekuensnya saja. Sehingga keterangan dari data akan dihilangkan. Seperti pada Gambar 4.22. Hanya sekuens dari mRNA saja yang akan digunakan sebagai inputan program.

```
CACAAGTTGGGTAGCCTGGCAGTGTCAGAAGTCTGAACCCAGCAT
AGTGGTCAGCAGGCAGGACGAATCACACTGAATGCAAACCACAG
GGTTTCGCAGCGTGGTGAGCATCACCCCACCACAGCCAAGGCGGC
GCTGGCTTTTTTTTTTTTTTTAATCTTTAACAATTTGAATATTTGT
TTTTACAAAGGTAAAAGAAATCATTGAGTCCCCCGCCTTCAGAAG
AGGGTGCATTTTCAGGAGGAAGCGATGGCTTCAGACAGCATATTT
GAGTCATTTCCTTCGTACCCACAGTGCTTCATGAGAGAATGCATA
CTTGGAATGAATCCTTCTAGAGACGTCCAC
```

Gambar 4.22 Data yang digunakan untuk inputan

Setelah data tinggal bentuk rantai sekuens, data akan dimasukkan ke program ekstraksi ciri. Dimana data string dari sekuens akan di jadikan data ciri numerik dengan menggunakan array list.

4.5.2 Ekstraksi Ciri

Ekstraksi ciri bertujuan membentuk ciri atau model matematik dari sekuens. Sehingga diperoleh matriks atau vektor yang dapat digunakan dalam proses pengklasifikasian. Metode yang digunakan dalam ekstraksi ciri adalah rantai markov orde 2. Dimana pencirian diambil dari probabilitas kemungkinan 2 basa nitrogen yang keluar setelah setelah suatu basah nitrogen. Dengan matriks yang digunakan dalam markov chains orde 2 sebagai berikut:

$$M = \begin{bmatrix} P(AA|A) & P(AA|C) & P(AA|G) & P(AA|T) \\ P(AC|A) & P(AC|C) & P(AC|G) & P(AC|T) \\ P(AG|A) & P(AG|C) & P(AG|G) & P(AG|T) \\ P(AT|A) & P(AT|C) & P(AT|G) & P(AT|T) \\ P(CA|A) & P(CA|C) & P(CA|G) & P(CA|T) \\ P(CC|A) & P(CC|C) & P(CC|G) & P(CC|T) \\ P(CG|A) & P(CG|C) & P(CG|G) & P(CG|T) \\ P(CT|A) & P(CT|C) & P(CT|G) & P(CT|T) \\ P(GA|A) & P(GA|C) & P(GA|G) & P(GA|T) \\ P(GC|A) & P(GC|C) & P(GC|G) & P(GC|T) \\ P(GG|A) & P(GG|C) & P(GG|G) & P(GG|T) \\ P(GT|A) & P(GT|C) & P(GT|G) & P(GT|T) \\ P(TA|A) & P(TA|C) & P(TA|G) & P(TA|T) \\ P(TC|A) & P(TC|C) & P(TC|G) & P(TC|T) \\ P(TG|A) & P(TG|C) & P(TG|G) & P(TG|T) \\ P(TT|A) & P(TT|C) & P(TT|G) & P(TT|T) \end{bmatrix}^T$$

dimana matriks M memiliki kolom =4 dan baris =16. P(AA|A) merupakan peluang keluarnya basa nitrogen AA setelah A dan seterusnya.

contoh:

A	A	C	T	T	C	G	A	T	G

Pada contoh sekuens diatas yang sudah berupa bentuk array. Sehingga array [0] = A, array[1] = A array[2]=C dan seterusnya. Dalam program akan membaca array = [0] akan dipengaruhi array [1] dan [2], lalu loop ke array = [1] akan dipengaruhi array [2] dan [3] seperti itu seterusnya hingga akhir sekuens.

Pada python data tersebut akan dijadikan array terlebih dahulu.

```
data=input("Masukkan data : ")
L=list(data)
```

Lalu akan di hitung probabilitas dari setiap ciri. Gambar 4.23 menunjukkan output dari suatu program markovian :

```
>>>
 RESTART: C:\Users\Ferocrash\AppData\Local\Programs\Python\Python36-32\Markov2.py
Masukkan data : CACAAGTTGGGTAGCCTGGCAGTGTCAGAAGTCTGAACCCAGCATAGTGGTCAGCAGGCAGGACGAATCACAC
TGAATGCAAACCACAGGGTTTCGCAGCGTGGTGAGCATCACCCCACCACAGCCAAGGCGGCGCTGGCTTTTTTTTTTTTTTTAATCTT
TAACAATTTGAATATTTGTTTTTACAAAGGTAAAAGAAATCATTGAGTCCCCCGCCTTCAGAAGAGGGTGCATTTTCAGGAGGAAGCGA
TGGCTTCAGACAGCATATTTGAGTCATTTCCTTCGTACCCACAGTGCTTCATGAGAGAATGCATACTTGGAATGAATCCTTCTAGAGAC
GTCCAC
Data yang anda masukkan =  ['C', 'A', 'C', 'A', 'A', 'G', 'T', 'T', 'G', 'G', 'G', 'T', '
A', 'G', 'C', 'C', 'T', 'G', 'G', 'C', 'A', 'G', 'T', 'G', 'T', 'C', 'A', 'G', 'A', 'A',
'G', 'T', 'C', 'T', 'G', 'A', 'A', 'C', 'C', 'C', 'A', 'G', 'C', 'A', 'T', 'A', 'G', 'T',
'G', 'G', 'T', 'C', 'A', 'G', 'C', 'A', 'G', 'G', 'C', 'A', 'G', 'G', 'A', 'C', 'G', 'A',
'A', 'T', 'C', 'A', 'C', 'A', 'C', 'T', 'G', 'A', 'A', 'T', 'G', 'C', 'A', 'A', 'A', 'C',
'C', 'A', 'C', 'A', 'G', 'G', 'G', 'T', 'T', 'T', 'C', 'G', 'C', 'A', 'G', 'C', 'G', 'T',
'G', 'G', 'T', 'G', 'A', 'G', 'C', 'A', 'T', 'C', 'A', 'C', 'C', 'C', 'C', 'A', 'C', 'C',
'A', 'C', 'A', 'G', 'C', 'C', 'C', 'A', 'A', 'G', 'G', 'C', 'G', 'G', 'C', 'G', 'C', 'T', 'G',
'G', 'C', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T',
'A', 'A', 'T', 'C', 'T', 'T', 'T', 'A', 'A', 'C', 'A', 'A', 'T', 'T', 'T', 'G', 'A', 'A',
'T', 'A', 'T', 'T', 'T', 'G', 'T', 'T', 'T', 'T', 'C', 'A', 'C', 'A', 'A', 'A', 'G', 'G',
'T', 'A', 'A', 'A', 'A', 'G', 'A', 'A', 'A', 'T', 'C', 'A', 'T', 'T', 'G', 'A', 'G', 'T',
'C', 'C', 'C', 'C', 'C', 'G', 'C', 'C', 'T', 'T', 'C', 'A', 'G', 'A', 'A', 'G', 'A', 'G',
'G', 'G', 'T', 'G', 'C', 'A', 'T', 'T', 'T', 'T', 'C', 'A', 'G', 'G', 'A', 'G', 'G', 'A',
'A', 'G', 'C', 'G', 'A', 'T', 'G', 'G', 'C', 'T', 'T', 'C', 'A', 'G', 'A', 'C', 'A', 'G',
'C', 'A', 'T', 'A', 'T', 'T', 'T', 'G', 'A', 'G', 'T', 'C', 'A', 'T', 'T', 'T', 'C', 'C',
'T', 'T', 'C', 'G', 'T', 'A', 'C', 'C', 'C', 'A', 'C', 'A', 'G', 'T', 'G', 'C', 'T', 'T',
'C', 'A', 'T', 'G', 'A', 'G', 'A', 'G', 'A', 'A', 'T', 'G', 'C', 'A', 'T', 'A', 'C', 'T',
'T', 'G', 'G', 'A', 'A', 'T', 'G', 'A', 'A', 'T', 'C', 'C', 'T', 'T', 'C', 'T', 'A', 'G',
'A', 'G', 'A', 'C', 'G', 'T', 'C', 'C', 'A', 'C']
Panjang sequense adalah =  346
Banyak basa Adenin =  93
Banyak basa Citosin =  77
Banyak basa Guanin =  81
Banyak basa Timin =  93
```

Gambar 4.23 List Sekuens

Output dari program berupa list data dan hitungan dari panjang sekuens dan juga banyak basa nitrogen yang ada pada sekuens. Dan juga banyak probabilitas dari kemungkinan basa nitrogen yang muncul sehingga nantinya akan dijadikan matriks transisi.

Seperti pada gambar dibawah ini.

```
Matrix Markovian =
[ 0.01169591  0.04481793  0.02662722  0.0078125 ]
[ 0.04093567  0.03921569  0.05325444  0.03125   ]
[ 0.0877193   0.07002801  0.09171598  0.0078125 ]
[ 0.02339181  0.02240896  0.0295858   0.0390625 ]
[ 0.06432749  0.07282913  0.05621302  0.0546875 ]
[ 0.0877193   0.15406162  0.15976331  0.140625  ]
[ 0.06432749  0.10644258  0.11242604  0.0625    ]
[ 0.03508772  0.06442577  0.06508876  0.046875  ]
[ 0.09356725  0.04481793  0.06804734  0.109375  ]
[ 0.20467836  0.10644258  0.12130178  0.1484375 ]
[ 0.0877193   0.10644258  0.1035503   0.1796875 ]
[ 0.03508772  0.00840336  0.03550296  0.0390625 ]
[ 0.00584795  0.0140056   0.00591716  0.0234375 ]
[ 0.05263158  0.05042017  0.0147929   0.0546875 ]
[ 0.0994152   0.07002801  0.0443787   0.03125   ]
[ 0.00584795  0.02521008  0.01183432  0.0234375 ]
```

Gambar 4.24 Matriks markovian dari data

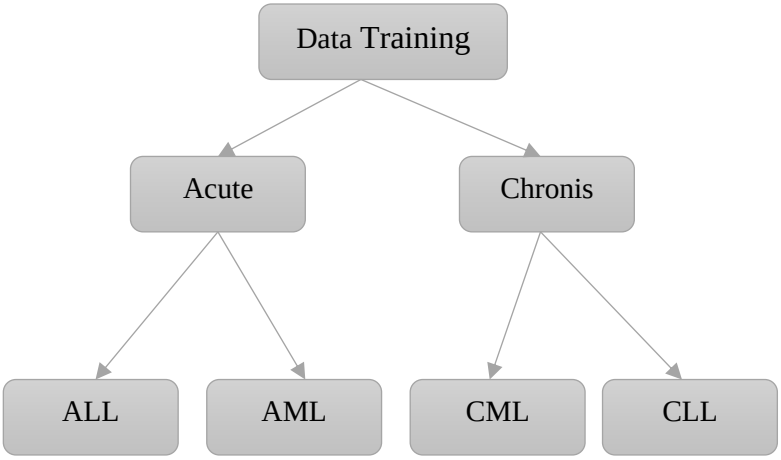
Data diatas menghasilkan matriks seperti pada gambar diatas. Sehingga data diatas yang akan digunakan sebagai data inputan training ataupun data testing nantinya.

4.5.3 *Support Vector Classification*

Support vector classification (SVC) merupakan bagian dari SVM. SVC bertugas mengklasifikasikan data dari matriks yang disajikan. Support vektor classification membutuhkan data training dan juga target training. Sehingga nantinya data training tersebut akan digunakan dalam membentuk support vektornya. Pada program ini SVC yang digunakan berupa library python yang sudah disediakan. Beberapa library yang digunakan dalam program ini.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import svm
```

Pada proses pengklasifikasian, diterapkan SVC bertingkat dengan membagi data terhadap kelas-kelas. Pertama data akan diklasifikasikan kedalam jenis leukemia *acute* dan *chronis*. Lalu tahap ke-2 klasifikasi dilakukan terhadap jenis ALL dan AML pada data yang termasuk *acute* dan pada data *chronis* diklasifikasikan ke dalam CLL dan CML. Klasifikasi bertahap ini untuk memudahkan dalam menunjukkan hasil plot dan mempermudah pengklasifikasian. Karena penerapan multi klasifikasi dengan metode SVM masih dengan pendekatan metode. Sehingga menyulitkan dalam menampilkan gambar bentuk support vektor itu sendiri.



Gambar 4.25 Tahap pembentukan support vektor oleh data training

4.5.3.1 Data Training dan Target

Data training merupakan data acuan dimana data tersebut yang akan membuat support vektor dari program. Data dari leukemia yang digunakan untuk mentraining support vector sebanyak 40 data. Data training berupa kumpulan dari data-data tiap sekuens jenis leukemia dengan label-label berbeda. Label tersebut tang dimakan target data. Sehingga data tersebut bisa didefinisikan jenis data apa dari target data tersebut. Dimana ALL=[0] , AML=[1], CLL=[2], CML=[3]. Bentuk dari data training dan target datadapat dilihat dari gambar dibawah ini :

```
Nilai X  :  [[ 0.11235955  0.13978495  0.08536585 ...,  0.10752688  0.01219512
   0.01639344]
 [ 0.01724138  0.04432133  0.02631579 ...,  0.02493075  0.01461988
   0.03053435]
 [ 0.11235955  0.13978495  0.08536585 ...,  0.10752688  0.01219512
   0.01666667]
 ...,
 [ 0.07518797  0.04841713  0.07706767 ...,  0.04841713  0.04511278
   0.08759124]
 [ 0.10526316  0.10638298  0.14933333 ...,  0.10638298  0.048       0.09494949]
 [ 0.112       0.10833333  0.08547009 ...,  0.04166667  0.03418803
   0.06730769]]
Nilai Target  :  [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

Gambar 4.26 Data Training dan Target

4.5.3.2 SVC Dengan Kernel Linear

Kernel linear merupakan formulasi untuk membentuk sesuatu model yang tidak linear menjadi bentuk yang linear. Sehingga algorithma yang dihasilkan terbatas untuk kasus-kasus yang linier. Karena itu, bila suatu kasus klasifikasi memperlihatkan ketidaklinieran, algorithma seperti perceptron tidak bisa mengatasinya. Secara umum, kasus-kasus di dunia nyata adalah kasus yang tidak linier. Kernel linear digunakan untuk membentuk support vektor ke dalam bentuk linear support vektor linear.

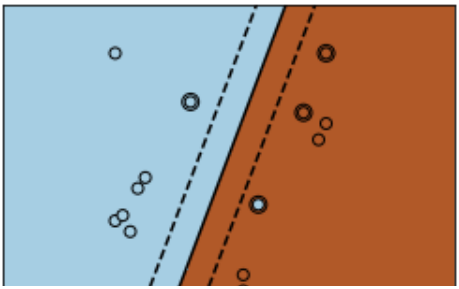
$$K(x_1, x_2) = \langle x_1 . x_2 \rangle$$

Kode dalam dalam program dapat diimplementasikan pada Gambar 4.27 dibawah ini.

```
svc1=OneVsRestClassifier(estimator=SVC(C=int(C), cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo', gamma=int(Gamma), kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)).fit(X_xor,y1)
```

Gambar 4.27 Kode SVC Kernel Linear

dengan *svc* sebagai variable baru dan *svm* sebagai metod dari library sklearn dengan pengklasifikasi SVC. Dimana SVC menggunakan kernel 'linear' dan penalty error C yang berasal dari inputan. Metod *fit(X_xor,y)* adalah target yang akan diklasifikasikan berupa *X* adalah data training dan *y* adalah target data.



Gambar 4.28 Sampel Support Vektor dengan kernel Linear

4.5.3.3 SVC Dengan Kernel Gaussian RBF

Sama seperti penggunaan kernel linear, kernel Gaussian RBF membentuk support vektor dengan persamaan.

$$K(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$$

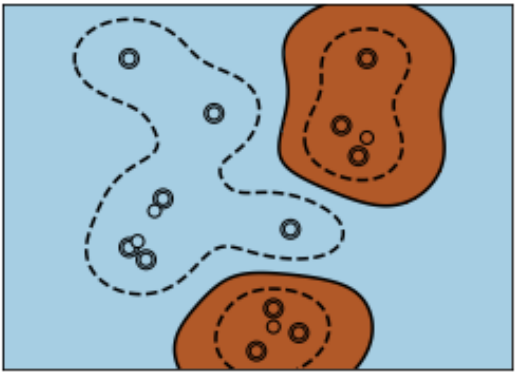
Pada program dituliskan dalam bentuk source code pada Gambar 4.29 sebagai berikut:

```
svc2=OneVsRestClassifier(estimator=SVC(C=int(C), cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovo',degree=3, gamma=int(Gamma), kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)).fit(X_xor,y1)
```

Gambar 4.29 Kode SVC Kernel rbf

Parameter C memberikan klasifikasi data latih terhadap *decision surface*. C rendah membuat *decision surface* menjadi halus, sementara C tinggi bertujuan untuk mengklasifikasikan semua contoh pelatihan dengan benar dengan memberi kebebasan model untuk memilih lebih banyak sampel sebagai vektor pendukung.

Parameter gamma menjadi parameter pembentuk hyperplane terhadap kernel yang digunakan. Gamma kecil berarti pembentukan hyperplane terhadap vektor pembantu lebih lebar sedangkan gamma besar membuat bentuk hyperplane menjadi presisi terhadap masing-masing vektor pembantu.



Gambar 4.30 Sampel Support Vektor dengan kernel RBF

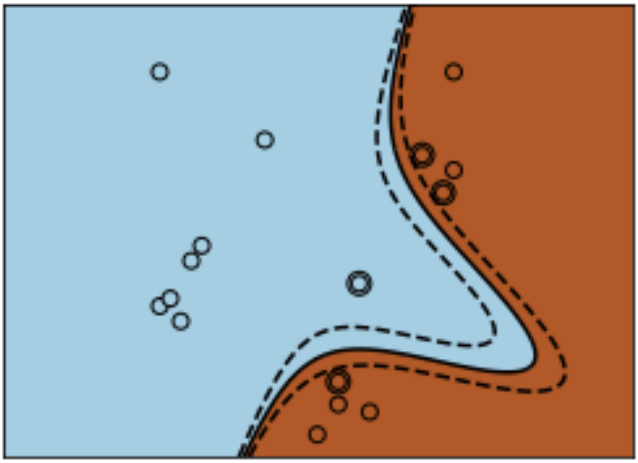
4.5.3.4 SVC Dengan Kernel Polynomial

Kernel Polynomial dinyatakan dalam persamaan :

$$K(x_1, x_2) = (\gamma\langle x_1 . x_2\rangle + r)^d$$

dimana d adalah perpangkatan dari persamaan dan r merupakan nilai kostanta sembarang dalam polinomial. Dan sama seperti kernel RBF. Gamma sebagai penentu seberapa jauh pengaruh dari data trining tercapai. dengan nilai rendah berarti jauh dan nilai tinggi yang berarti dekat (presisi). Parameter gamma dapat dilihat sebagai kebalikan dari radius pengaruh sampel yang dipilih oleh model sebagai vektor pendukung.

Pada penelitian ini nilai d dan r ditetapkan yaitu d=3 dan r=1. Sehingga nantinya yang menjadi parameter/variabel bebas yaitu hanya gamma pada persamaan kernel.



Gambar 4.31 Sampel Support Vektor dengan kernel Polynomial

BAB V
HASIL DAN PEMBAHASAN PENGUJIAN

Pada bab ini akan dibahas bagaimana hasil dari penelitian yang dilakukan dengan hasil pengujian pada data DNA/RNA. Dimana juga dilakukan perbandingan hasil antar kernel yang digunakan dengan ujicoba pada data test.

5.1 DataTraining

Training *support vector classification* (SVC) merupkan pokok dari penelitian. Karena pada proses ini *support vektor* dibentuk dari berbagai data training. Data training yang terbentuk menggunakan 40 sample data training. Data training tersebut terdiri dari 10 data *Acute myeloid Leukemia (AML)*, 10 data *Acute Lymphocytic Leukemia (ALL)*, 10 data *Chronic Lymphocytic Leukemia (CLL)* dan 10 data *Chronic Myelocytic Leukemia (CML)*. Dari setiap data sekuensnya akan dibentuk menjadi data training. Tabel 5.1 menunjukkan data training yang digunakan dalam membuat support vektor classification.

Tabel 5.1 Data Training

No	Jenis Leukemia	Data Training			
		ALL	**AML**	**CLL**	**CML**
1	**K**	AAI26374	CAA56092	BC140252	AAB60388
2	**O**	AB464287	CAA56093	BC146525	AAB60393
3	**D**	BC069422	L34598	BC148493	AAK73017
4	**E**	BC126373	X79549	BC153086	AAM69373
5		BC160033	X79550	DJ352105.1	AAW83120
6	**D**	CAB72103	X90976	DJ352106.1	AF285785
7	**A**	EAW59002	X90977	DJ352109.1	AY762229
8	**T**	EAX06875	X90979	NR_027932.1	EHH64370

9	**A**	HQ258056	X90980	NR_027933.1	HV197178
10		Y00811	X90981	S82592.1	M25946.1

Kode data merupakan ID data yang berada pada database NCBI dan EMBL. Data tersebut kemudian dijadikan data training dengan masing-masing dari anggota jenis leukemia 10 data training. Gambar 5.1 Meninjukkan bentuk data training.

```
[[ 0.11235955  0.13978495  0.08536585 ...,  0.10752688  0.01219512
   0.01639344]
 [ 0.01724138  0.04432133  0.02631579 ...,  0.02493075  0.01461988
   0.03053435]
 [ 0.11235955  0.13978495  0.08536585 ...,  0.10752688  0.01219512
   0.01666667]
 ...,
 [ 0.10018215  0.10334347  0.1530343  ...,  0.10638298  0.05013193
   0.09533469]
 [ 0.07518797  0.04841713  0.07706767 ...,  0.04841713  0.04511278
   0.08759124]
 [ 0.112       0.10833333  0.08547009 ...,  0.04166667  0.03418803
   0.06730769]]
[[-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]
 [-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]
 [-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]
 ...,
 [-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]
 [-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]
 [-1.29712064 -1.29612064 -1.29512064 ...,  1.40087936  1.40187936
   1.40287936]]
```

Gambar 5.1 Data training pada program

Matriks data training diatas merupakan ciri dari masing-masing dari setiap data training. Dengan target data yaitu kode dimana ciri tersebut menunjukkan kode data training jenis leukemia. Gambar 5.2 menunjukkan target data dari matriks ciri dari data training.

```
Nilai Target  :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,
2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

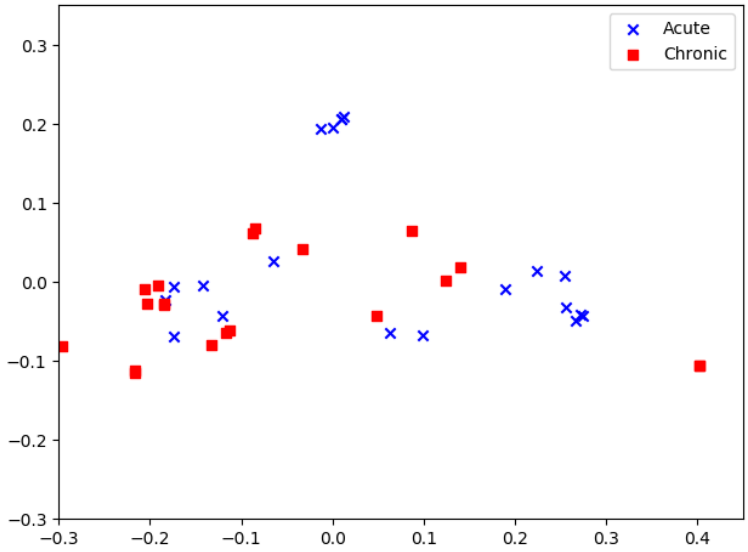
Gambar 5.2 Target data.

Target data kode 0 merupakan ciri dari jenis leukemia *Acute Lymphocytic Leukemia (ALL)*, kode 1 merupakan data dari *Acute myeloid Leukemia (AML)*, kode 2 merupakan data dari *Chronic Lymphocytic Leukemia (CLL)* dan kode 3 merupakan data dari *Chronic Myelocytic Leukemia (CML)*.

5.2 Hasil SVC

Hasil *support vector classification* divisualisasikan pada bidang 2 Dimensi. Dimana data training yang berdimensi besar dikompres dengan menggunakan metode *principle component analysis* (PCA).

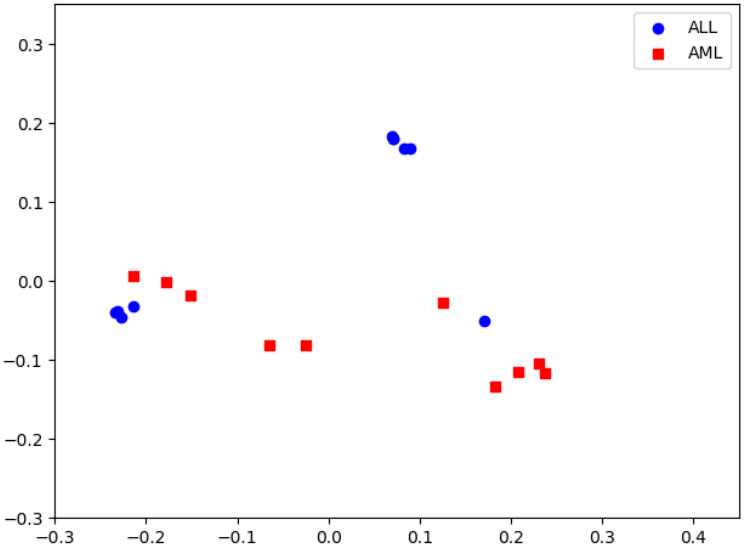
- Plot Data Training Acute dan Chronic



Gambar 5.3 Plot data training untuk akute dan chronic

Klasifikasi tahap pertama dengan mentraining data terhadap data jenis leukemia akute dan chronic. Data training yang berdimensi besar akan dikompres terlebih dahulu sehingga hanya ada 2 ciri utama dan juga 40 data training. Dimana setiap data training akan di plot untuk menjadikan vektor pendukung. Gambar 5.3 menunjukkan data training untuk masing-masing jenis leukemia. Jenis leukemia akute dengan warna data biru dan leukemia jenis chronic dengan data merah.

- Plot Data Training ALL dan AML



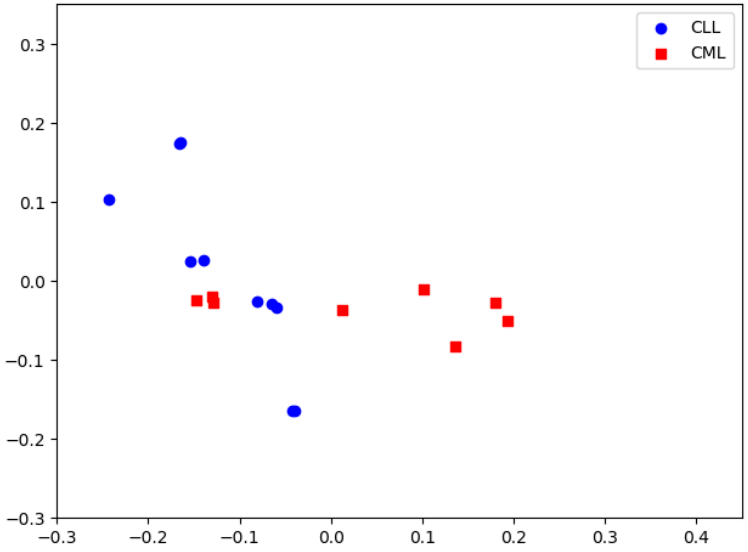
Gambar 5.4 Plot data training untuk ALL dan AML pada jenis leukemia Akute.

Setelah mentraining terhadap akute dan chronic. Data yang tergolong ke jenis akute akan diklasifikasikan lagi menjadi 2. Yaitu jenis *Acute Lymphocytic Leukemia* (ALL) dan *Acute Myeloid Leukemia* (AML). Data yang tergolong dalam ALL dengan warna

biru sedangkan data training untuk AML dengan warna merah. Nantinya data training tersebut digunakan untuk klasifikasi ke-2 jika data test masuk jenis acute maka akan dicari lagi data test masuk jenis *akute lymphocytic* atau *myeloid*.

- Plot Data Training CLL dan CML

Jenis leukemia Chronis terbagi menjadi 2 jenis lagi yaitu Chronic Lymphocytic Leukemia (CLL) dan Chronic Myelocytic Leukemia (CML). Seperti pada tahap pengklasifikasian jenis acute. Pada jenis chronis juga dilakukan klasifikasi ke-2 untuk menentukan jenis salah satu chronis. Sehingga nantinya data akan terklasifikasi menjadi 4 kelas.



Gambar 5.5 Plot data training untuk CLL dan CML pada jenis leukemia Chronic.

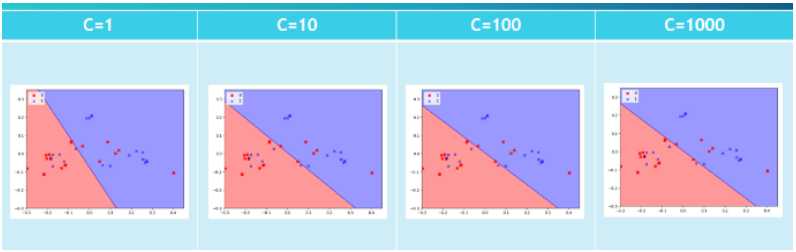
5.1.1 Hasil SVC Dengan Kernel Linear

Hasil klasifikasi digambarkan dalam bentuk grafik 2 dimensi. Dimana grafik akan menunjukkan pada setiap jenis klasifikasi dan pada setiap kernel yang digunakan.

- Klasifikasi Acute dan Chronic

Hasil dari SVC dengan menggunakan kernel linear tampak seperti pada Tabel 5.2. Dimana kernel linear diujikan dengan mengubah nilai parameter *C*. Nilai parameter diubah-ubah untuk mencari nilai yang tepat dan performa SVC yang terbaik jika digunakan kernel linear.

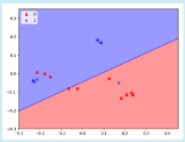
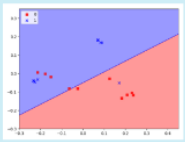
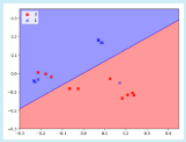
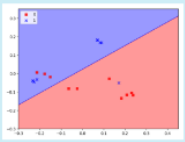
Tabel 5.2 Hasil klasifikasi dengan kernel linear pada data training Akut dan Kronis



Dapat dilihat dari gambar yang ada pada Tabel 5.2 diatas bahwa, nilai parameter C sangat mempengaruhi terhadap pembentukan dari vektor pembantu. Semakin besar nilai C vektor pembantu yang digunakan semakin menyeluruh. Sehingga area kedua kelas semakin bagus.

- Klasifikasi ALL dan AML

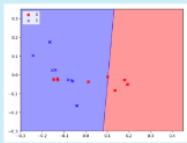
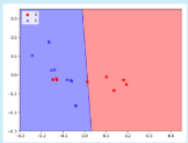
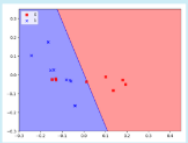
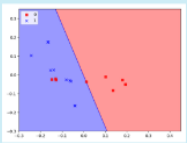
Tabel 5.3 Hasil klasifikasi dengan kernel linear pada data training ALL dan AML

C=1	C=10	C=100	C=1000
			

Pada Tabel 5.3 diatas menunjukkan hasil *hyperplane* terhadap vektor pendukung dari kelas ALL dengan AML dengan menggunakan kernel linear. Sama seperti waktu mengklasifikasikan pada akut dan kronis. Vektor pendukung akan lebih merata jika nilai *C* semakin besar.

- Klasifikasi CLL dan CML

Tabel 5.4 Hasil klasifikasi dengan kernel linear pada data training CLL dan CML

C=1	C=10	C=100	C=1000
			

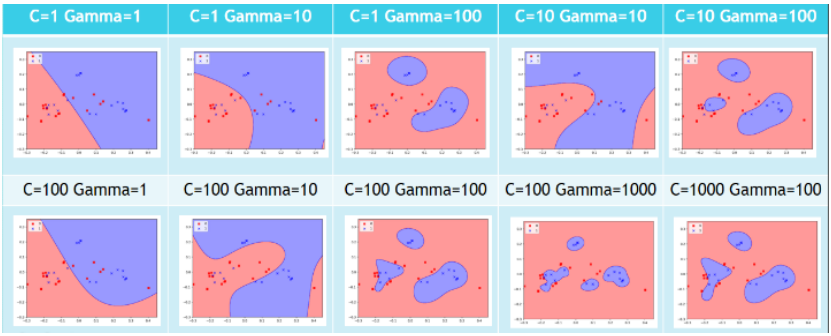
Pada klasifikasi tahap-2 yaitu pengklasifikasian terhadap jenis CML dan CLL dengan menggunakan kernel Linear. Hasil dari SVC terhadap data training seperti gambar pada Tabel 5.4. Sama seperti klasifikasi terhadap kernel linear sebelumnya. Semakin

besar nilai *C* maka semakin menyeluruh dan merata vektor pendukung yang digunakan.

5.1.2 Hasil SVC Dengan Kernel RBF

- Klasifikasi Acute dan Chronic

Tabel 5.5 Hasil klasifikasi dengan kernel Gaussian RBF pada data training Acute dan Chronic



Dengan menggunakan persamaan kernel *gaussian radial basic function*.

$$K(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$$

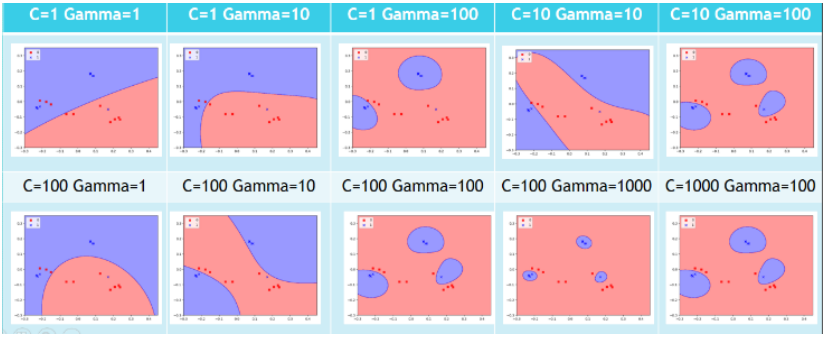
Didapatkan gambar *hyperplane* pemisah kedua kelas seperti pada gambar di Tabel 5.5. Uji coba dengan mengguunakan kernel RBF dengan menggunakan data training terhadap jenis Acute dan Chronic. Parameter *C* dan parameter *Gamma* diubah-ubah sedemikian rupa untuk mendapatkan hasil *hyperplane* terbaik agar nantinya hasil prediksi menjadi semakin akurat dan sensitif. Pada Tabel 5.5 diatas dapat dilihat bahwa semakin besar nilai *C* yang digunakan maka vektor pendukung yang digunakan semakin detail. Dengan meminimalkan nilai error terhadap pengklasifikasian. Semakin besar nilai *gamma* maka semakin presisi atau semakin pendek bentuk hyperplane terhadap data training. Gambar pada Tabel 5.5 diatas ketika nilai *C*=1 vektor pendukung yang digunakan pada data dengan nilai vektor mayoritas dan masih terdapat beberapa error training. Semakin

besar nilai *C*=100, vektor pendukung pada data training semakin detail dan rinci dengan meminimalisir nilai error training. Sehingga hampir semua data training menjadi vektor pendukung.

Nilai gamma yang kecil menunjukkan bentuk *hyperplane* terhadap kernel yang digunakan kurang sensitif. Pada nilai *Gamma*=1 bentuk hyperplane kurang presisi. Nilai *gamma* besar contoh 1000. Menghasilkan bentuk hyperplane yang sensitif dan presisi. Bentuk yang presisi belum tentu menjadi bentuk yang optimal terhadap prediksi data test nantinya. Pada subbab selanjutnya akan dibahas apakah semakin presisi suatu hyperplane akan semakin akurat prediksi data.

- Klasifikasi ALL dan AML

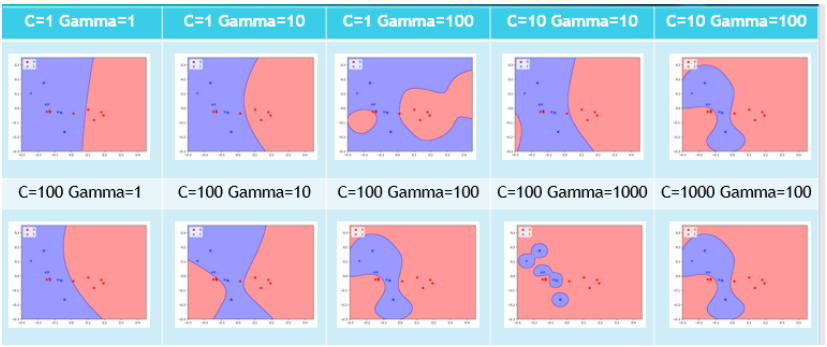
Tabel 5.6 Hasil klasifikasi dengan kernel Gaussian RBF pada data training ALL dan AML



Pada klasifikasi tahap-2 dengan membentuk kelas ALL dan AML pada data training jenis Akute. Pada kernel RBF menghasilkan bentuk hyperplane dan vektor pembantu pada Tabel 5.6. Sama dengan pada uji data training akute dan chronic. Nilai *C* dan *Gamma* sangat mempengaruhi bentuk dari hyperplane. Semakin besar nilai *C* semakin detain vektor pendukung yang digunakan terhadap data trining. Semakin Besar nilai *Gamma* semakin presisi bentuk hyperplane yang dihasilkan.

- Klasifikasi CLL dan CML

Tabel 5.7 Hasil klasifikasi dengan kernel Gaussian RBF pada data training CLL dan CML



Tabel 5.7 menunjukkan hasil klasifikasi dengan kernel RBF pada pengklasifikasian CLL dan CML pada data training jenis Chronic. Hyperplane yang dhasilkan pada kernel RBF menunjukkan bentuk presisi pada saat nilai *Gamma* semakin besar.

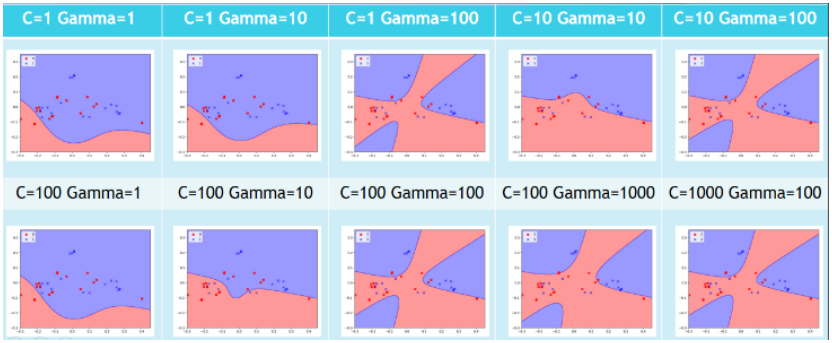
5.1.3 Hasil SVC Dengan Kernel Polynomial

Hasil dari SVC dengan polynomial menggunakan persamaan kernel :

$$K(x_1, x_2) = (\gamma\langle x_1, x_2\rangle + 1)^d$$

- Klasifikasi Acute dan Chronic

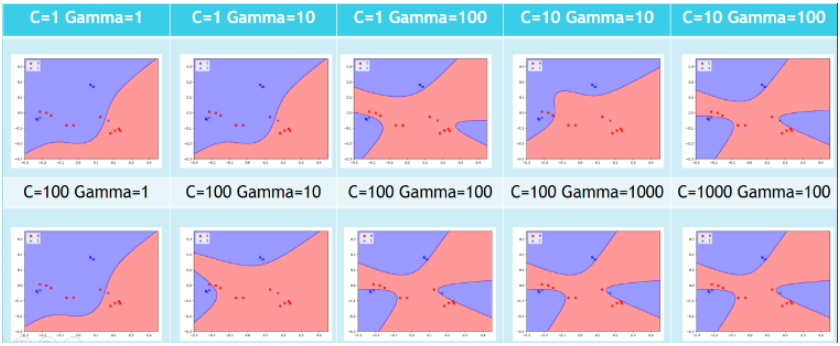
Tabel 5.8 Hasil klasifikasi dengan kernel Polynomial pada data training Acute dan Chronic



Penggunaan kernel selanjutnya yaitu polynomial. Kernel polynomial menggunakan 3 parameter yaitu nilai *C, Gamma* dan *Pangkat (Degree)*. Pada uji coba ini hanya mengubah-ubah nilai dari parameter *C* dan *Gamma* saja. Sedangkan parameter degree tetap yaitu 3. Pada Tabel 5.8 merupakan klasifikasi menggunakan kernel polynomial pada data train akute dan chronic. Hasil dari pembentukan hyperplane dengan kernel polynomial terlihat lebih buruk pada *C* dan *Gamma* kecil. Terlihat masih banyak terdapat error data training pada pengklasifikasian. Pada nilai *C* dan *Gamma* diperbesar, nilai error semakin kecil dan bentuk dari hyperplane terhadap vektor pendukung semakin presisi.

- Klasifikasi ALL dan AML

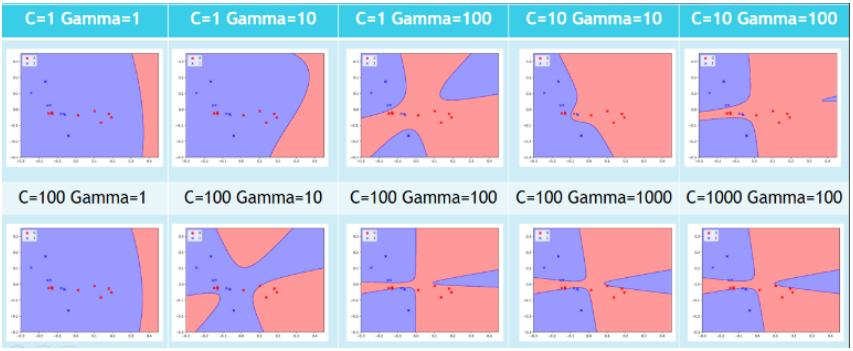
Tabel 5.9 Hasil klasifikasi dengan kernel Polynomial pada data training ALL dan AML



Tabel 5.9 menunjukkan hasil klasifikasi data trining ALL dan AML pada jenis acute dengan kernel polynomial degree=3. Sama seperti pada klasifikasi tahap-1 dengan kernel polynomial. Nilai *C* dan *Gamma* sangan mempengaruhi bentuk hyperplane dan vektor pendukung yang digunakan.

- Klasifikasi CLL dan CML

Tabel 5.10 Hasil klasifikasi dengan kernel Polynomial pada data training CLL dan CML



Nilai error data latih dapat dilihat pada saat nilai *C* dan *Gamma* kecil. Semakin besar nilai kedua parameter tersebut SVC akan meminimalisir nilai error dan membentuk hyperplane yang presisi. Tabel 5.9 diatas menunjukkan hasil trining SVC dengan kernel polynomial pada jenis data CLL dan CML.

5.3 Data Testing

Data Test digunakan untuk menguji seberapa besar keberhasilan program dalam memprediksi data. Dengan adanya data testing dapat juga membandingkan bagimana kinerja setiap kernel dan kernel yang mana yang bagus untuk digunakan dalam menyelesaikan permasalahan dari sequens DNA/RNA dari leukemia. Tabel 5.11 adalah data yang digunakan untuk testing program SVC dengan menggunakan 3 kernel (Linear,RBF, Polynomial).

Tabel 5.11 Data Test

No	Kode Data	Jenis Leukemia			
		ALL	AML	CLL	CML
1	AAI26376	✓	-	-	-
2	AB464288	✓	-	-	-
3	BC126373	✓	-	-	-
4	BC126375	✓	-	-	-
5	EAX06874	✓	-	-	-
6	EAX06877	✓	-	-	-
7	CAA56093	-	✓	-	-
8	X79549	-	✓	-	-
9	X90975	-	✓	-	-
10	X90978	-	✓	-	-
11	X90982	-	✓	-	-
12	X90983	-	✓	-	-
13	BC140252	-	-	✓	-
14	BC148493	-	-	✓	-
15	BC153086	-	-	✓	-
16	DJ352106.1	-	-	✓	-
17	DJ352109.1	-	-	✓	-
18	NR_027932.1	-		✓	-
19	AAB60389	-	-	-	✓
20	AAB60394	-	-	-	✓
21	AAM75154	-	-	-	✓
22	CDJ81798	-	✓	-	-
23	EHH28713	-	-	-	✓
24	HV197178	-	-	-	✓
25	AAA51720	-	✓	-	-

Data test terdiri dari 25 code sekuens leukemia. Dimana setiap datanya sudah diketahui jenis leukemia. Nantinya data test tersebut menjadi data validasi untuk prediksi dari SVC dengan menggunakan ketiga kernel yang dibuat oleh data training. Contoh sampel dari data test pada program dapat dilihat pada Gambar 5.6 .

```
Data Test =  [ 0.11764706  0.12195122  0.17204301  0.05633803  0.05042017
0.09756098
  0.05376344  0.07042254  0.1512605   0.09756098  0.06451613  0.02816901
  0.05042017  0.04878049  0.04301075  0.05633803  0.08403361  0.1097561
  0.08602151  0.04225352  0.02521008  0.07317073  0.12903226  0.04225352
  0.00840336  0.01219512  0.01075269  0.          0.07563025  0.09756098
  0.02150538  0.08450704  0.12605042  0.01219512  0.07526882  0.11267606
  0.05042017  0.          0.10752688  0.09859155  0.07563025  0.01219512
  0.09677419  0.11267606  0.03361345  0.01219512  0.01075269  0.07042254
  0.00840336  0.06097561  0.06451613  0.04225352  0.01680672  0.06097561
  0.01075269  0.05633803  0.10084034  0.07317073  0.04301075  0.08450704
  0.02521008  0.1097561   0.01075269  0.04225352]
```

Gambar 5.6 Sampel data test

Data tes diatas merupakan data yang siap untuk diproses (diprediksi) bahwa data dengan ciri tersebut masuk dalam klasifikasi yang mana. Dengan begitu program akan otomatis memplotkan data test terhadap support vektor sehingga masuk di klasifikasi yang mana antara ALL, AML, CLL atau CML.

5.4 Hasil Prediksi Data

Prediksi data merupakan hasil dari pengolahan data test pada pengklasifikasian. Prediksi data test memuat tiga kernel. Dengan menggunakan kernel tersebut akn diketahui bagaimana hasil prediksi dari masing-masing kernel. Karena setiap kernel memiliki bentuk support vektor yang berbeda, maka dimungkinkan adanya hasil yang berbeda dari setiap hasil kernel. Gambar 5.7 menunjukkan vektor dengan data test (CDJ81798). Sekuens data test seperti dibawah ini.

```
ATGGATATTCAGATATCGAAGCCGACACCCGGCATGAGCGTGTACCATAA
CGTACACGAGTTCCTGCATGCTAATAAAACACTGCTATTGAAATCGAGTAG
TCCAAACATTTTCTACACGAAATTACCGGAACATCATCGATCGAACAAGTC
TCTACCGTCTCCATTTACTGTACTTATTACCTCACCGGTACCGGATGGAAC
ACTGGTAACCGTAGCTGCCGGAAATGACGAGACACCGTGTGGTGAAGTTC
GTCATGATACCGCTAAAGTGGTTCGTCAAGTGGCACGATTTAGCGATTTAC
GATTCGTTGGAAAAAGCGGCAGAGGGAAGAATTTCCATCTAACGATATGT
GTATATACAAAGCCGATGATGATCGCAATGGTTAGCCGAGCTATAAAGGT
TACGGTAGATGGCCCACGTGAAGCACGTACTCCAAGCTCAGCTGTCAACC
ATCGGCGACGACCGTGTGTACCGTTCGTCAGCCCTGTGCCGCCGTTCTTTC
CCAGCTATCAATCCCTTATGTGGTTCACGCCAGACTTTCGAGCGATACCAG
CGTTACTGAGTCGACCAACGCCGAACGAGACTTCACCGGAGGGTGTGCGA
ACGGTCTATCAAATTCTTTACTGGAATTTGAGCTCTGGGGAAGAAATGAGG
TAA
```

Gambar 5.7 Sampel data test

Program akan mencirikan data diatas menjadi bentuk vektor dengan menggunakan metode rantai markov. Hasil yang berupa vektor (numerik list) akan menjadi data test yang diinputkan terhadap nilai fitness SVC pada program dengan penggunaan kernel yang diinginkan. Gambar 5.8 merupakan sampel data test dari sekuens pada Gambar 5.7.

```
Vektor dari DNA/RNA =  [0.07777777777777778, 0.0625, 0.10759493670886076, 0.05, 0.06666666666666667,
0.075, 0.05063291139240506, 0.1125, 0.06666666666666667, 0.04375, 0.0759493670886076, 0.03125,
0.05555555555555555, 0.0625, 0.08860759493670886, 0.06875, 0.05555555555555555, 0.06875,
0.0379746835443038, 0.075, 0.08888888888888889, 0.03125, 0.06329113924050633, 0.04375,
0.08333333333333333, 0.1125, 0.04430379746835443, 0.075, 0.05, 0.025, 0.056962025316455694, 0.05625,
0.044444444444444446, 0.1375, 0.06329113924050633, 0.06875, 0.08888888888888889, 0.03125,
0.03164556962025317, 0.0375, 0.022222222222222223, 0.0625, 0.02531645569620253, 0.075,
0.044444444444444446, 0.09375, 0.06962025316455696, 0.06875, 0.05555555555555555, 0.05625,
0.0759493670886076, 0.06875, 0.06111111111111111, 0.0375, 0.056962025316455694, 0.0875,
0.07222222222222222, 0.0625, 0.08860759493670886, 0.01875, 0.06666666666666667, 0.0375,
0.06329113924050633, 0.0625]
```

Gambar 5.8 Ciri dari data CDJ81798

Data test diatas akan diproses dalam program untuk diprediksi ciri dari data tersebut. Hasil dari pembacaan data test tersebut berupa prediksi data, belum 100% akurat. Pada pembelajaran mesin memerlukan data trining yang banyak untuk

mengambil semua ciri dari tiap jenis yang akan diklasifikasikan. Semakin banyak data training yang digunakan, maka nilai sensitifitas semakin besar terhadap prediksi data baru (data test). Dengan menggunakan C=1 dan gamma=10 program memprediksi data tes (CDJ81798) dengan ketiga kernel seperti pada gambar 5.9

```
SVC Dengan Kernel Linear
Prediksi dari DNA/RNA tersebut adalah jenis leukemia AML
ACUTE MYELOID LEUKEMIA

SVC Dengan Kernel Gaussian RBF
Prediksi dari DNA/RNA tersebut adalah jenis leukemia AML
ACUTE MYELOID LEUKEMIA

SVC Dengan Kernel Polynomial
Prediksi dari DNA/RNA tersebut adalah jenis leukemia CML
CHRONIC MYELOCYTIC LEUKEMIA
```

Gambar 5.9 Hasil prediksi data test dengan *C=1* dan *Gamma=1*

5.5 Perbandingan Hasil Kernel

Kernel sangat mempengaruhi hasil dari suatu prediksi data. Penggunaan kernel yang berbeda digunakan untuk menentukan akurasi yang dicapai. Dengan menggunakan 3 kernel (linear, Gaussian RBF dan Polynomial) akan diuji seberapa akurat masing-masing kernel terhadap 25 data test. Dengan mengubah-ubah nilai parameter C dan *Gamma* dimana nilai $0 < C < \infty$ dan $0 < Gamma < \infty$, kedua parameter tersebut bertujuan untuk apakah semakin bagus performa kernel tersebut jika bentuk hyperplane semakin presisi dan vektor pendukung yang digunakan semakin sensitif.

Tabel 5.12 Perbandingan Kernel dengan Tes C=1, Gamma=1

No	Kode Data	Jenis Kernel		
		LINEAR	GAUSIAN RBF	POLYNOMIAL
1	AAI26376	ALL	ALL	ALL
2	AB464288	ALL	ALL	ALL
3	BC126373	ALL	ALL	ALL
4	BC126375	ALL	ALL	ALL
5	EAX06874	ALL	ALL	ALL
6	EAX06877	ALL	ALL	ALL
7	CAA56093	AML	AML	AML
8	X79549	AML	AML	AML
9	X90975	AML	AML	AML
10	X90978	AML	AML	AML
11	X90982	AML	AML	AML
12	X90983	AML	AML	AML
13	BC140252	CLL	CLL	CLL
14	BC148493	CLL	CLL	CLL
15	BC153086	CLL	CLL	CLL
16	DJ352106.1	CLL	CLL	CLL
17	DJ352109.1	CLL	CLL	CLL
18	NR_027932.1	CLL	CLL	CLL
19	AAB60389	CML	CML	ALL
20	AAB60394	ALL	ALL	ALL
21	AAM75154	CML	CML	CML
22	CDJ81798	AML	AML	AML
23	EHH28713	CLL	CLL	CLL
24	HV197178	AML	AML	AML
25	AAA51720	AML	AML	AML

Pada uji pertama menerapkan nilai *C*=1 dan *Gamma*=1, dimana pada kernel linear terhadap 3 error test sama dengan nilai

pada kernel RBF. Pada kernel polynomial mengalami performa lebih buruk dari kernel yang lain dengan 4 error tes.

Tabel 5.13 Perbandingan Kernel dengan Tes C=1, Gamma=10

No	Kode Data	Jenis Kernel		
		LINEAR	GAUSIAN RBF	POLYNOMIAL
1	AAI26376	ALL	ALL	ALL
2	AB464288	ALL	ALL	ALL
3	BC126373	ALL	ALL	ALL
4	BC126375	ALL	ALL	ALL
5	EAX06874	ALL	ALL	ALL
6	EAX06877	ALL	ALL	ALL
7	CAA56093	AML	AML	AML
8	X79549	AML	AML	AML
9	X90975	AML	AML	AML
10	X90978	AML	AML	AML
11	X90982	AML	AML	AML
12	X90983	AML	AML	AML
13	BC140252	CLL	CLL	CLL
14	BC148493	CLL	CLL	CLL
15	BC153086	CLL	CLL	CLL
16	DJ352106.1	CLL	CLL	CLL
17	DJ352109.1	CLL	CLL	CLL
18	NR_027932.1	CLL	CLL	CLL
19	AAB60389	CML	CML	CML
20	AAB60394	ALL	CML	CML
21	AAM75154	CML	CML	CML
22	CDJ81798	AML	AML	CML
23	EHH28713	CLL	CML	CML
24	HV197178	AML	AML	CML
25	AAA51720	AML	AML	AML

Uji coba selanjutnya terhadap performa masing-masing kernel dengan mengubah nilai parameter *C*=1 dan *Gamma*=10. Pada uji coba yang ke-2 ini tampak performa mereka membaik. Dengan nilai error pada setiap kernel berkurang. Pada kernel linear terdapat 2 error, RBF 1 error dan polynomial 1 error. Nilai yang membaik tersebut dikarenakan nilai parameter yang semakin optimal.

Tabel 5.14 Perbandingan Kernel dengan Tes C=10 , Gamma=10

No	Kode Data	Jenis Kernel		
		LINEAR	GAUSIAN RBF	POLYNOMIAL
1	AAI26376	ALL	ALL	ALL
2	AB464288	ALL	ALL	ALL
3	BC126373	ALL	ALL	ALL
4	BC126375	ALL	ALL	ALL
5	EAX06874	ALL	ALL	ALL
6	EAX06877	ALL	ALL	ALL
7	CAA56093	AML	AML	AML
8	X79549	AML	AML	AML
9	X90975	AML	AML	AML
10	X90978	AML	AML	AML
11	X90982	AML	AML	AML
12	X90983	AML	AML	AML
13	BC140252	CLL	CLL	CLL
14	BC148493	CLL	CLL	CLL
15	BC153086	CLL	CLL	CLL
16	DJ352106.1	CLL	CLL	CLL
17	DJ352109.1	CLL	CLL	CLL
18	NR_027932.1	CLL	CLL	CLL
19	AAB60389	CML	CML	CML
20	AAB60394	ALL	CML	CML

21	AAM75154	CML	CML	CML
22	CDJ81798	AML	AML	CML
23	EHH28713	CML	CML	CML
24	HV197178	AML	CML	CML
25	AAA51720	AML	AML	AML

Uji coba yang ke-3 terhadap ketiga kernel dengan menerapkan nilai *C*=10 dan *Gamma*=10. Hasil dari prediksi membaik pada pada kernel RBF saja, dengan membaca 100% benar terhadap 25 data test yang digunakan. Sedangkan pada dua kernel yang lain mengalami error yang sama seperti pada uji ke-2.

Tabel 5.15 Perbandingan Kernel dengan Tes C=100 , Gamma=1000

No	Kode Data	Jenis Kernel		
		LINEAR	GAUSIAN RBF	POLYNOMIAL
1	AAI26376	ALL	ALL	ALL
2	AB464288	ALL	ALL	ALL
3	BC126373	ALL	ALL	ALL
4	BC126375	ALL	ALL	ALL
5	EAX06874	ALL	ALL	ALL
6	EAX06877	ALL	ALL	ALL
7	CAA56093	AML	AML	AML
8	X79549	AML	AML	AML
9	X90975	AML	AML	AML
10	X90978	AML	AML	AML
11	X90982	AML	AML	AML
12	X90983	AML	AML	AML
13	BC140252	CLL	CLL	CLL
14	BC148493	CLL	CLL	CLL
15	BC153086	CLL	CLL	CLL
16	DJ352106.1	CLL	CLL	CLL

17	DJ352109.1	CLL	CLL	CLL
18	NR_027932.1	CLL	CLL	CLL
19	AAB60389	CML	AML	CML
20	AAB60394	CML	CML	CML
21	AAM75154	CML	CML	CML
22	CDJ81798	CML	AML	CML
23	EHH28713	CML	CML	CML
24	HV197178	CML	CML	CML
25	AAA51720	AML	AML	AML

Uji terakhir (ke-4) menerapkan *C*=100 dan *Gamma*=1000, nilai yang besar bertujuan membentuk hyperplane semakin presisi dan vektor pendukung semakin rinci dan detail. Pada uji ke-4 ini didapatkan penurunan performa pada kernel RBF dan kenaikan performa pada kernel Linear dan pada kernel polynomial mengalami performa yang stabil dengan 1 nilai error tes.

Tabel 5.16 menunjukkan hasil prediksi untuk beberapa nilai parameter dimana $0 < C \leq 1000$ dan $0 < Gamma \leq 1000$.

Tabel 5.16 Perbandingan Hasil Prediksi ketiga kernel Dengan berbagai nilai parameter

No	Parameter		Kernel		
	C	Gamma	Linear	RBF	Poly
1	0.5	0.5	21	21	20
2		1	21	21	20
3		5	21	21	22
4		10	21	21	22
5		20	21	24	25
6		50	21	24	25
7		100	21	24	25
8		200	21	24	24
9		500	21	24	24

10	0.5	1000	21	24	24
11	1	0.5	22	21	20
12		1	22	22	21
13		5	22	22	24
14		10	22	24	24
15		20	22	24	24
16		50	22	25	24
17		100	22	25	24
18		200	22	25	24
19		500	22	24	24
20		1000	22	24	24
21	5	0.5	22	21	20
22		1	22	22	21
23		5	22	22	24
24		10	22	24	24
25		20	22	24	24
26		50	22	25	24
27		100	22	25	24
28		200	22	25	24
29		500	22	24	24
30		1000	22	24	24
31	10	0.5	23	22	20
32		1	23	22	21
33		5	23	24	24
34		10	23	25	24
35		20	23	25	24
36		50	23	25	24
37		100	23	25	24
38		200	23	25	24
39		500	23	24	24
40		1000	23	24	24

41	20	0.5	23	22	20
42		1	23	22	21
43		5	23	24	24
44		10	23	25	24
45		20	23	25	24
46		50	23	25	24
47		100	23	25	24
48		200	23	25	24
49		500	23	24	24
50		1000	23	24	24
51	50	0.5	23	22	20
52		1	23	22	21
53		5	23	24	24
54		10	23	25	24
55		20	23	25	24
56		50	23	25	24
57		100	23	25	24
58		200	23	25	24
59		500	23	24	24
60		1000	23	24	24
61	100	0.5	24	24	19
62		1	24	24	24
63		5	24	24	24
64		10	24	25	24
65		20	24	25	24
66		50	24	25	24
67		100	24	25	24
68		200	24	25	24
69		500	24	24	24
70		1000	24	24	24
71	200	0.5	24	24	19

72	200	1	24	24	24
73		5	24	24	24
74		10	24	25	24
75		20	24	25	24
76		50	24	25	24
77		100	24	25	24
78		200	24	25	24
79		500	24	24	24
80		1000	24	24	24
81	500	0.5	24	24	22
82		1	24	24	22
83		5	24	24	22
84		10	24	25	22
85		20	24	25	22
86		50	24	25	24
87		100	24	25	24
88		200	24	24	24
89		500	24	24	24
90		1000	24	22	24
91	1000	0.5	24	24	24
92		1	24	24	24
93		5	24	24	24
94		10	24	24	24
95		20	24	24	24
96		50	24	24	24
97		100	24	24	24
98		200	24	24	24
99		500	24	22	24
100		1000	24	22	24

Dengan menggunakan 25 data tes, di lakukan uji coba sebanyak 100 kali dengan mengganti-ganti nilai parameter dari kernel *Support Vector Classification*, dengan nilai $0 < C \leq 1000$ dan $0 < Gamma \leq 1000$. Hasilnya bahwa pada kernel linear dengan hanya menggunakan satu parameter saja yaitu *C* nilai terbaik pada $100 \leq C \leq 1000$ dengan memprediksi benar sebanyak 24 data tes.

Pada kernel RBF dengan memanfaatkan kedua parameter yang tersedia yaitu *C* dan *Gamma* , didapatkan nilai prediksi terbaik pada saat $20 \leq C \leq 200$ dan $10 < Gamma \leq 200$. Dengan kombinasi dari kedua parameter tersebut pada selang tersebut, kernel Gaussian RBF dapat memprediksi 100% atau 25 benar dari data tes. Sedangkan untuk kernel Polynomial, performa terbaiknya pada $C = 0.5$ dan nilai gamma berkisar antara $20 \leq Gamma \leq 100$.

Nilai *C* mengontrol trade off antara margin dan error klasifikasi. Nilai C yang besar berarti memberi pinalti yang lebih besar terhadap error klasifikasi dengan menggunakan data training sebagai vektor pendukung yang menyeluruh. Sedangkan nilai *Gamma* merupakan kostanta dari kernel RBF dan Polynomial. Dimana sebagai pembentuk *hyperplane* dari penggunaan kernel tersebut. Semakin besar nilai *Gamma, hyperplane* yang terbentuk menjadi presisi atas kompleksitas data training yang menjadi vektor pendukung. Sedangkan jika *Gamma* kecil klasifikasi yang terbentuk tidak bisa membaca kompleksitas data yang ada.

Nilai *Gamma* yang besar kurang bagus dalam hal klasifikasi. Karena bentuk yang presisi terhadap data training membuat penyempitan ciri dari setiap data. Sehingga data yang terklasifikasikan merupakan data yang memiliki kemiripan ciri yang besar dengan data training. Sedangkan data DNA leukemia

yang ada tidak selalu mirip dengan data training yang ada, dikarenakan DNA manusia yang berbeda-beda dari setiap manusia.

BAB VI
PENUTUP

Bab ini berisi kesimpulan dari hasil seluruh penelitian yang dilakukan dan saran dari peneliti untuk pembaca.

6.1 Kesimpulan

Ekstraksi ciri dengan rantai markov orde 2 menghasilkan ciri pada sekuens sebanyak 64 ciri. Metode klasifikasi *Support Vektor Machine* merupakan bagian dari *supervised Learning*. Dimana ada data training sebagai acuan dan data tes sebagai uji keberhasilan metode. SVM mengklasifikasikan data leukemia menjadi 4 kelas yaitu *Acute Lymphocytic Leukemia* (ALL), Acute *Acute Myelogenous Leukemia* (AML),*Chronic Lymphocytic Leukemia* (CLL) dan *Chronic Myelogenous Leukemia* (CML).

Pada metode SVM menggunakan kernel linear, RBF dan Polynomial. Parameter *C* dan *Gamma* sangat mempengaruhi hasil trining dan hasil prediksi data tes. Parameter *C* sebagai pembentuk vektor pendukung. Semakin besar nilai *C* semakin banyak data trining yang menjadi vektor pendukung. Parameter *Gamma* sebagin pembentuk *hyperplane* pemisah kedua kelas. Semakin Besar nilai *Gamma* maka *hyperplane* yang terbentuk semakin presisi terhadap masing-masing vektor pembantu.

Kernel linear memiliki performa terbaik pada $100 \leq C \leq 1000$ dengan berhasil memprediksi benar 24 data tes. Kernel RBF memiliki performa terbaik pada $20 \leq C \leq 200$ dan $10 < Gamma \leq 200$ dengan memprediksi benar 25 data tes. Kernel Polynomial memiliki performa terbaik pada $C = 0.5$ dan nilai gamma berkisar antara $20 \leq Gamma \leq 100$ dengan memprediksi benar 25 data tes.

Kernel terbaik yang digunakan dalam mengkalsifikasikan DNA jenis leukemia dengan menggunakan Gaussian RBF. Karena performanya yang bagus pada selang parameter $20 \leq C \leq 200$ dan $10 < Gamma \leq 200$.

6.2 Saran

Saran dari penulis terhadap penelitian selanjutnya adalah :

1. Menambah data Train dan Test untuk menambah sensifitas dari hyperplane yang terbentuk dari vektor pendukung.
2. Perlu adanya perbandingan metode untuk menunjukkan kualitas SVM dibanding metode yang lain dengan objek penelitian yang sama.
3. Data training yang digunakan bersifat statis.

DAFTAR PUSTAKA

[1] World Health Organization (WHO). 2009. “Universal Access To Cancer”. Switzerland: WHO.

[2] Kemenkes RI. 2015. “Pusat Data Dan Informasi Kementrian Kesehatan RI 2015”. Jakarta : Kementrian Kesehatan RI.

[3] Dyrskjot L, Zieger K, Real FX, dkk. 2007. “Gene Expression Signature Predict Outcome In Non-Muscle-Invasive Bladder Carcinoma: A Multicenter Validation Study”. Clinic cancer Res

[4] Ransohoff Df. 2004. “Rules of evidence for cancer molecular-marker discovery and Validation”. Nat rev cancer

[5] James W.F Catto, Maysam F.A, dkk. 2010. “The application of artificial intelligence to microarray data: Identification of a Novel Gene Signature to Identify Bladder cancer progression”. United Kingdom: European Urology.

[6] Yayasan Kanker Indonesia, .2017. http://yayasankankerindonesia.org/tentang-kanker, Jakarta : YKI Pusat; akses: 27 Agustus 2017/22.10.

[7] Nugroho A.S, Witarto A.B, Handoko Dwi. 2003. “Application of Support Vector Machine in Bioinformatics”, Gifu-Japan: proceding of Indonesian scientific Meeting in central japan.

[8] Abe Shigeo. 2005. “Support Vector Machine for Pattern Classification”, Japan : Department of Computer Science, University of Exeter, Exeter, EX4 4PT, UK

[9] Fajar, ML. 2013."Identifikasi DNA Bakteri Menggunakan Metode Ekstraksi Ciri Rantai Markov Dengan Probabilistik Neural Network Sebagai Classifier", Skripsi, Bogor : Institute Pertanian Bogor.

[10] Kusuma, WA. 2012. "Combined approaches for improving the performance of denovo DNA sequence assembly and metagenomic classification of short fragments from next generation sequencer ". Disertasi, Tokyo: Tokyo Institute of Technology.

[11] Robin S, Radolphe F, Schbath S. 2005. "DNA, Words and Models". Cambridge(UK): Cambridge University Press

[12] Anthony JF Griffiths, Jeffrey H Miller, David T Suzuki, Richard C Lewontin, and William M Gelbart (2000). An Introduction to Genetic Analysis. University of British Columbia, University of California, Harvard University (7 ed.) (W. H. Freeman). p. Properties of RNA. ISBN 0-7167-3520-2. Diakses tanggal 2010-08-24.

[13] Chih-Wei Hsu , Chih-Jen Lin. 2014. "A Comparison of Methods for Multi-class Support Vector Machines". Department of Computer Science and Information Engineering National Taiwan University. Taipei 106, Taiwan.

[14] "What is Python Good For?". General Python FAQ. Python Software Foundation. Diakses tanggal 2017-11-27. https://docs.python.org

[15] "What is Python? Executive Summary". Python documentation. Python Software Foundation. Diakses tanggal 2017-11-27. https://docs.python.org

[16] "General Python FAQ". python.org. Python Software Foundation. Diakses tanggal 2017-11-27. https://docs.python.org

[17] “Scikit Learn Documentation”. Modul Python. Machine learning in python. Diakses tanggal 2017-11-14. http://scikit-learn.org/stable/modules/svm.html.

Halaman ini sengaja dikosongkan

LAMPIRAN

Lampiran 1 . Data Diri

Nama : Moh. Hamim Zajuli Al Faroby
TTL : Banyuwangi, Juli 1995
Alamat : Gladag, Rogojampi, Kabupaten Banyuwangi, Jawa Timur
Email : alfa.crash@gmail.com
No. Hp : 081331653603
Hobi : Gaming and sport

Sebelum memulai pendidikan di Institut Teknologi Sepuluh Nopember, saya pernah mengemban pendidikan formal di MI Miftahul Ulum Mangir – Rogojampi lulusan 2007, Setelah itu melanjutkan ke tingkat sekolah menengah pertama di SMP Bustanul Makmur Genteng – Banyuwangi lulusan 2010 dan setelah itu di SMA Negeri 1 Glagah – Banyuwangi lulusan 2013. Selain pendidikan formal saya juga pernah menempuh pendidikan non formal di pondok pesantren yaitu PP. Bustanul Makmur II tahun 2007-2010 dan juga PP. Sirojut Tholibien 2010-2013.

Pengalaman organisasi saya sejak duduk di SMP. Saat itu mengikuti Organisasi SC (Student Council) nama OSIS di sekolah saya dan juga pengurus Takmir di SMP. Saat beranjak di SMA organisasi yang pernah saya ikuti di Takmir masjid Al-Hurriyah sebagai kepala dan ketua Ekstrakulikuler Bulutangkis. Saat berada di ITS saya ikut organisasi dibidang dakwah yaitu Ibnu Muqlah sejak tahun 2014-2016 dan juga pada organisasi UKM IBC (ITS Badminton Community) sebagai staf kepelatihan anggota.

Saya gemar sesuatu yang berhubungan dengan komputer, apakah itu programming atau gaming. Selain itu sejak kecil saya suka dengan olahraga terutama bulutangkis, banyak yang bilang saya multitalent karena bisa semua olahraga meskipun ada beberapa yang tidak tahu aturan permainannya.

Lampiran 2. Source Code Markovian

```
data=input("Masukkan data : ")
L=list(data)
print("Data yang anda masukkan = ",L)
print("Panjang sequense adalah = ",str(len(L)))
panjang=len(L)- 2
a=0
c=0
g=0
t=0
for j in range(panjang):
    if (L[j]=="A"):
        a=a+1

    if (L[j]=="C"):
        c=c+1

    if (L[j]=="G"):
        g=g+1

    if (L[j]=="T"):
        t=t+1

print("Banyak basa Adenin = ",a)
print("Banyak basa Citosin = ",c)
print("Banyak basa Guanin = ",g)
print("Banyak basa Timin = ",t)

aaa=0,aac=0,aag=0,aat=0,aca=0,acc=0,acg=0,act=0,aga=0,agc=0
agg=0,agt=0,ata=0,atc=0,atg=0,att=0,caa=0,cac=0,cag=0,cat=0
cca=0,ccc=0,ccg=0,cct=0,cga=0,cgc=0,cgg=0,cgt=0,cta=0,ctc=0
ctg=0,ctt=0,gaa=0,gac=0,gag=0,gat=0,gca=0,gcc=0,gcg=0,gct=0
gga=0,ggc=0,ggg=0,ggt=0,gta=0,gtc=0,gtg=0,gtt=0,taa=0,tac=0,
```

```
tag=0,tat=0,tca=0,tcc=0,tcg=0,tct=0,tga=0,tgc=0,tgg=0,tgt=0,
tta=0,ttc=0,ttg=0,ttt=0

for i in range(panjang):
    if (L[i]=="A" and L[i+1]=="A" and L[i+2]=="A"):
        aaa=aaa+1

    if (L[i]=="A" and L[i+1]=="A" and L[i+2]=="C"):
        aac=aac+1

    if (L[i]=="A" and L[i+1]=="A" and L[i+2]=="G"):
        aag=aag+1

    if (L[i]=="A" and L[i+1]=="A" and L[i+2]=="T"):
        aat=aat+1

    if (L[i]=="A" and L[i+1]=="C" and L[i+2]=="A"):
        aca=aca+1

    if (L[i]=="A" and L[i+1]=="C" and L[i+2]=="C"):
        acc=acc+1

    if (L[i]=="A" and L[i+1]=="C" and L[i+2]=="G"):
        acg=acg+1

    if (L[i]=="A" and L[i+1]=="C" and L[i+2]=="T"):
        act=act+1

    if (L[i]=="A" and L[i+1]=="G" and L[i+2]=="A"):
        aga=aga+1

    if (L[i]=="A" and L[i+1]=="G" and L[i+2]=="C"):
        agc=agc+1
```

```
    if (L[i]=="A" and L[i+1]=="G" and L[i+2]=="G"):
        agg=agg+1

    if (L[i]=="A" and L[i+1]=="G" and L[i+2]=="T"):
        agt=agt+1

    if (L[i]=="A" and L[i+1]=="T" and L[i+2]=="A"):
        ata=ata+1

    if (L[i]=="A" and L[i+1]=="T" and L[i+2]=="C"):
        atc=atc+1

    if (L[i]=="A" and L[i+1]=="T" and L[i+2]=="G"):
        atg=atg+1

    if (L[i]=="A" and L[i+1]=="T" and L[i+2]=="T"):
        att=att+1

    if (L[i]=="C" and L[i+1]=="A" and L[i+2]=="A"):
        caa=caa+1

    if (L[i]=="C" and L[i+1]=="A" and L[i+2]=="C"):
        cac=cac+1

    if (L[i]=="C" and L[i+1]=="A" and L[i+2]=="G"):
        cag=cag+1

    if (L[i]=="C" and L[i+1]=="A" and L[i+2]=="T"):
        cat=cat+1

    if (L[i]=="C" and L[i+1]=="C" and L[i+2]=="A"):
        cca=cca+1

    if (L[i]=="C" and L[i+1]=="C" and L[i+2]=="C"):
```

```
    ccc=ccc+1

if (L[i]=="C" and L[i+1]=="C" and L[i+2]=="G"):
    ccg=ccg+1

if (L[i]=="C" and L[i+1]=="C" and L[i+2]=="T"):
    cct=cct+1

if (L[i]=="C" and L[i+1]=="G" and L[i+2]=="A"):
    cga=cga+1

if (L[i]=="C" and L[i+1]=="G" and L[i+2]=="C"):
    cgc=cgc+1

if (L[i]=="C" and L[i+1]=="G" and L[i+2]=="G"):
    cgg=cgg+1

if (L[i]=="C" and L[i+1]=="G" and L[i+2]=="T"):
    cgt=cgt+1

if (L[i]=="C" and L[i+1]=="T" and L[i+2]=="A"):
    cta=cta+1

if (L[i]=="C" and L[i+1]=="T" and L[i+2]=="C"):
    ctc=ctc+1

if (L[i]=="C" and L[i+1]=="T" and L[i+2]=="G"):
    ctg=ctg+1

if (L[i]=="C" and L[i+1]=="T" and L[i+2]=="T"):
    ctt=ctt+1

if (L[i]=="G" and L[i+1]=="A" and L[i+2]=="A"):
    gaa=gaa+1
```

```
if (L[i]=="G" and L[i+1]=="A" and L[i+2]=="C"):
    gac=gac+1

if (L[i]=="G" and L[i+1]=="A" and L[i+2]=="G"):
    gag=gag+1

if (L[i]=="G" and L[i+1]=="A" and L[i+2]=="T"):
    gat=gat+1

if (L[i]=="G" and L[i+1]=="C" and L[i+2]=="A"):
    gca=gca+1

if (L[i]=="G" and L[i+1]=="C" and L[i+2]=="C"):
    gcc=gcc+1

if (L[i]=="G" and L[i+1]=="C" and L[i+2]=="G"):
    gcg=gcg+1

if (L[i]=="G" and L[i+1]=="C" and L[i+2]=="T"):
    gct=gct+1

if (L[i]=="G" and L[i+1]=="G" and L[i+2]=="A"):
    gga=gga+1

if (L[i]=="G" and L[i+1]=="G" and L[i+2]=="C"):
    ggc=ggc+1

if (L[i]=="G" and L[i+1]=="G" and L[i+2]=="G"):
    ggg=ggg+1

if (L[i]=="G" and L[i+1]=="G" and L[i+2]=="T"):
    ggt=ggt+1
```

```
if (L[i]=="G" and L[i+1]=="T" and L[i+2]=="A"):
   gta=gta+1

if (L[i]=="G" and L[i+1]=="T" and L[i+2]=="C"):
   gtc=gtc+1

if (L[i]=="G" and L[i+1]=="T" and L[i+2]=="G"):
   gtg=gtg+1

if (L[i]=="G" and L[i+1]=="T" and L[i+2]=="T"):
   gtt=gtt+1

if (L[i]=="T" and L[i+1]=="A" and L[i+2]=="A"):
   taa=taa+1

if (L[i]=="T" and L[i+1]=="A" and L[i+2]=="C"):
   tac=tac+1

if (L[i]=="T" and L[i+1]=="A" and L[i+2]=="G"):
   tag=tag+1

if (L[i]=="T" and L[i+1]=="A" and L[i+2]=="T"):
   tat=tat+1

if (L[i]=="T" and L[i+1]=="C" and L[i+2]=="A"):
   tca=tca+1

if (L[i]=="T" and L[i+1]=="C" and L[i+2]=="C"):
   tcc=tcc+1

if (L[i]=="T" and L[i+1]=="C" and L[i+2]=="G"):
   tcg=tcg+1

if (L[i]=="T" and L[i+1]=="C" and L[i+2]=="T"):
```

```
        tct=tct+1

    if (L[i]=="T" and L[i+1]=="G" and L[i+2]=="A"):
        tga=tga+1

    if (L[i]=="T" and L[i+1]=="G" and L[i+2]=="C"):
        tgc=tgc+1

    if (L[i]=="T" and L[i+1]=="G" and L[i+2]=="G"):
        tgg=tgg+1

    if (L[i]=="T" and L[i+1]=="G" and L[i+2]=="T"):
        tgt=tgt+1

    if (L[i]=="T" and L[i+1]=="T" and L[i+2]=="A"):
        tta=tta+1

    if (L[i]=="T" and L[i+1]=="T" and L[i+2]=="C"):
        ttc=ttc+1

    if (L[i]=="T" and L[i+1]=="T" and L[i+2]=="G"):
        ttg=ttg+1

    if (L[i]=="T" and L[i+1]=="T" and L[i+2]=="T"):
        ttt=ttt+1

p=[[aaa/a,caa/c,gaa/g,taa/t],
   [aac/a,cac/c,gac/g,tac/t],
   [aag/a,cag/c,gag/g,tag/t],
   [aat/a,cat/c,gat/g,tat/t],
   [aca/a,cca/c,gca/g,tca/t],
   [acc/a,ccc/c,gcc/g,tcc/t],
   [acg/a,ccg/c,gcg/g,tcg/t],
   [act/a,cct/c,gct/g,tct/t],
```

```
   [aga/a,cga/c,gga/g,tga/t],
   [agc/a,cgc/c,ggc/g,tgc/t],
   [agg/a,cgg/c,ggg/g,tgg/t],
   [agt/a,cgt/c,ggt/g,tgt/t],
   [ata/a,cta/c,gta/g,tta/t],
   [atc/a,ctc/c,gtc/g,ttc/t],
   [atg/a,ctg/c,gtg/g,ttg/t],
   [att/a,ctt/c,gtt/g,ttt/t],
   ]
print("")

print("Matrix Markovian = ")
for row in p:
    print (row)

print("")
pa= aaa/a + aac/a + aag/a + aat/a + aca/a + acc/a + acg/a + act/a +
aga/a + agc/a + agg/a + agt/a + ata/a + atc/a + atg/a + att/a
pc= caa/c + cac/c + cag/c + cat/c + cca/c + ccc/c + ccg/c + cct/c +
cga/c + cgc/c + cgg/c + cgt/c + cta/c + ctc/c + ctg/c + ctt/c
pg= gaa/g + gac/g + gag/g + gat/g + gca/g + gcc/g + gcg/g + gct/g +
gga/g + ggc/g + ggg/g + ggt/g + gta/g + gtc/g + gtg/g + gtt/g
pt= taa/t + tac/t + tag/t + tat/t + tca/t + tcc/t + tcg/t + tct/t + tga/t +
tgc/t + tgg/t + tgt/t + tta/t + ttc/t + ttg/t + ttt/t

print("Nilai A = ",pa)
print("Nilai C = ",pc)
print("Nilai G = ",pg)
print("Nilai T = ",pt)

vector=[aaa/a,caa/c,gaa/g,taa/t,aac/a,cac/c,gac/g,tac/t,aag/a,cag/c,gag
/g,tag/t,aat/a,cat/c,gat/g,tat/t,aca/a,cca/c,gca/g,tca/t,acc/a,ccc/c,gcc/g,t
cc/t,acg/a,ccg/c,gcg/g,tcg/t,act/a,cct/c,gct/g,tct/t,aga/a,cga/c,gga/g,tga
/t,agc/a,cgc/c,ggc/g,tgc/t,agg/a,cgg/c,ggg/g,tgg/t,agt/a,cgt/c,ggt/g,tgt/
```

```
t,ata/a,cta/c,gta/g,tta/t,atc/a,ctc/c,gtc/g,ttc/t,atg/a,ctg/c,gtg/g,ttg/t,att/a,
ctt/c,gtt/g,ttt/t]

print(vector)
```

Lampiran 3. Source Code Support Vector Machine

```
# Import packages untuk visualisasi data dan klasifikasi
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
import warnings

# Import packages untuk klasifikasi dan dataset
import numpy as np
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.multiclass import OneVsRestClassifier

def versiontuple(v):
    return tuple(map(int, (v.split("."))))

def plot_decision_regions(X, y, classifier, test_idx=None,
resolution=0.001):

    # set warna pada grafik
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(-0.3, 0.45)
    plt.ylim(-0.3, 0.35)

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap(idx),
                    marker=markers[idx], label=cl)

    # highlight test samples
    if test_idx:
        # plot all samples
        if not versiontuple(np.__version__) >= versiontuple('1.9.0'):
            X_test, y_test = X[list(test_idx), :], y[list(test_idx)]
            warnings.warn('Please update to NumPy 1.9.0 or newer')
        else:
            X_test, y_test = X[test_idx, :], y[test_idx]

        plt.scatter(X_test[:, 0],
                    X_test[:, 1],
                    c='',
                    alpha=1.0,
                    linewidths=0.2,
                    marker='o',
                    s=55, label='test set')

print('')
print('====================================')
print("===SUPPORT VECTOR CLASSIFICATION===")
print('====================================')
```

```
#input parameter klasifikasi
C=input("Masukkan Nilai C (C tidak boleh 0) : ")
Gamma=input("Masukkan Nilai Gamma (Gamma Tidak Boleh 0) :
")
print('==================================')
print('')

#dataset Leukemia
X_xor=np.array(X_Train) #ukuran data 64x40

y1=[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,
3,3,3,3,3,3]
#y1=LabelBinarizer().fit_transform(y1)
#y2=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1]
print('Data Training  : ',X_xor)
print('')
#print('Nilai Target  : ',y1)
print('')

#C=100
#svc1=svm.SVC(kernel='rbf',C=1,gamma=100).fit(X,y1)

#prediksi untuk data test
svc1=OneVsRestClassifier(estimator=SVC(C=int(C),
cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovo', gamma=int(Gamma), kernel='linear',
max_iter=-1, probability=False, random_state=None,
shrinking=True,tol=0.001, verbose=False)).fit(X_xor,y1)

svc2=OneVsRestClassifier(estimator=SVC(C=int(C),
cache_size=200, class_weight=None, coef0=0.0,
```

```
decision_function_shape='ovo',degree=3, gamma=int(Gamma),
kernel='rbf',max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)).fit(X_xor,y1)

svc3=OneVsRestClassifier(estimator=SVC(C=int(C),
cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovo', degree=3, gamma=int(Gamma),
kernel='poly', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)).fit(X_xor,y1)

#PCA dataset Akute dan cronis
pca=PCA(n_components=2)
X_xor=pca.fit(X_xor).transform(X_xor)
print("PCA Data Training = ",X_xor)
y_xor=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1]

#plot data training untuk Akute dan Cronis
y_xor = np.where(y_xor, 0, 1)
plt.scatter(X_xor[y_xor == 1, 0],
        X_xor[y_xor == 1, 1],
        c='b', marker='x',
        label='Acute')
plt.scatter(X_xor[y_xor == 0, 0],
        X_xor[y_xor == 0, 1],
        c='r',
        marker='s',
        label='Chronic')
plt.xlim([-0.3, 0.45])
plt.ylim([-0.3, 0.35])
plt.legend(loc='best')
plt.tight_layout()
print("")
print("==============================")
```

```
print("=KLASIFIKASI Acute dan Chronic=")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel linear pada akut dan cronis
svm = SVC(kernel='linear', C=int(C))
svm.fit(X_xor, y_xor)
# Visualisasi klasifikasi
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Linear")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel RBF pada akut dan cronis
svm = SVC(kernel='rbf', gamma=int(Gamma), C=int(C))
svm.fit(X_xor, y_xor)
# Visualize the decision boundaries
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel RBF")
print("==============================")
plt.show()

# Membuat klasifikasi dengan kernel polynomial pada akut dan
cronis
svm = SVC(kernel='poly', degree=3, gamma=int(Gamma),
C=int(C))
svm.fit(X_xor, y_xor)
# Visualize the decision boundaries
```

```
plot_decision_regions(X_xor, y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Polynomial")
print("==============================")
plt.show()

#dataset ALL dan AML
X_xor_a=np.array(X_Train_A) #data berukuran 64x20

#PCA dataset ALL dan AML
pca=PCA(n_components=2)
X_xor_a=pca.fit(X_xor_a).transform(X_xor_a)
y_xor_a=[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1]

#plot dataset ALL dan AML
y_xor_a = np.where(y_xor_a, 0, 1)
plt.scatter(X_xor_a[y_xor_a == 1, 0],
        X_xor_a[y_xor_a == 1, 1],
        c='b', marker='o',
        label='ALL')
plt.scatter(X_xor_a[y_xor_a == 0, 0],
        X_xor_a[y_xor_a == 0, 1],
        c='r',
        marker='s',
        label='AML')
plt.xlim([-0.3, 0.45])
plt.ylim([-0.3, 0.35])
plt.legend(loc='best')
plt.tight_layout()
print("")
print("==============================")
print("====KLASIFIKASI ALL dan AML====")
print("==============================")
```

```
plt.show()


# Membuat klasifikasi dengan kernel linear pada ALL dan AML
svm = SVC(kernel='linear', C=int(C))
svm.fit(X_xor_a, y_xor_a)
# Visualize the decision boundaries
plot_decision_regions(X_xor_a, y_xor_a, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Linear")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel rbf pada ALL dan AML
svm = SVC(kernel='rbf', gamma=int(Gamma), C=int(C))
svm.fit(X_xor_a, y_xor_a)
# Visualize the decision boundaries
plot_decision_regions(X_xor_a, y_xor_a, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel RBF")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel polynomial pada ALL dan
AML
svm = SVC(kernel='poly',degree=3, gamma=int(Gamma), C=int(C))
svm.fit(X_xor_a, y_xor_a)
# Visualize the decision boundaries
plot_decision_regions(X_xor_a, y_xor_a, classifier=svm)
plt.legend(loc='upper left')
```

```
plt.tight_layout()
print("Kernel Polynomial")
print("==============================")
plt.show()

#dataset CLL dan CML
X_xor_c=np.array(X_Train_C) #data berukuran 64X20

#PCA dataset CLL dan CML
pca=PCA(n_components=2)
X_xor_c=pca.fit(X_xor_c).transform(X_xor_c)
y_xor_c=[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1]

#plot dataset CLL dan CML
y_xor_c = np.where(y_xor_c, 0, 1)
plt.scatter(X_xor_c[y_xor_c == 1, 0],
        X_xor_c[y_xor_c == 1, 1],
        c='b', marker='o',
        label='CLL')
plt.scatter(X_xor_c[y_xor_c == 0, 0],
        X_xor_c[y_xor_c == 0, 1],
        c='r',
        marker='s',
        label='CML')
plt.xlim([-0.3, 0.45])
plt.ylim([-0.3, 0.35])
plt.legend(loc='best')
plt.tight_layout()
print("")
print("==============================")
print("====KLASIFIKASI CLL dan CML====")
print("==============================")
plt.show()
```

```
# Membuat klasifikasi dengan kernel linear pada CLL dan CML
svm = SVC(kernel='linear', C=int(C))
svm.fit(X_xor_c, y_xor_c)
# Visualize the decision boundaries
plot_decision_regions(X_xor_c, y_xor_c, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Linear")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel rbf pada CLL dan CML
svm = SVC(kernel='rbf', gamma=int(Gamma), C=int(C))
svm.fit(X_xor_c, y_xor_c)
# Visualize the decision boundaries
plot_decision_regions(X_xor_c, y_xor_c, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel RBF")
print("==============================")
plt.show()


# Membuat klasifikasi dengan kernel polynomial pada CLL dan
CML
svm = SVC(kernel='poly',degree=3, gamma=int(Gamma), C=int(C))
svm.fit(X_xor_c, y_xor_c)
# Visualize the decision boundaries
plot_decision_regions(X_xor_c, y_xor_c, classifier=svm)
plt.legend(loc='upper left')
plt.tight_layout()
print("Kernel Polynomial")
```

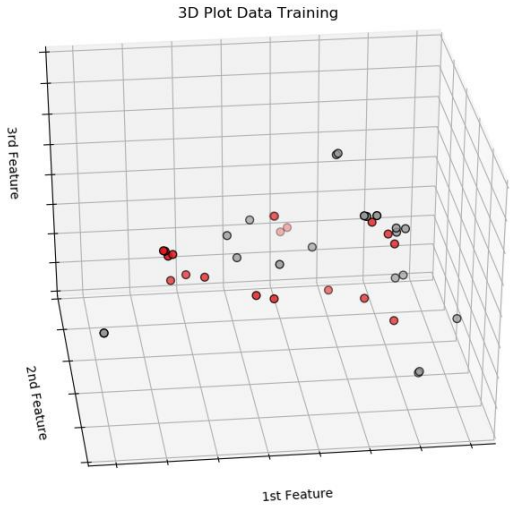
```
print("================================")
plt.show()

#Hasil prediksi data test pada SVC
print('')
print('')
print('=====PREDIKSI DATA LEUKEMIA TES PADA KE-4
KELAS=====')
print('==========================================
========')
Data_Test=vector
a=svc1.predict([Data_Test])
#print('Data Test = ',Data_Test)
print('')
print('================================')
print('====SVC Dengan Kernel Linear====')
print('================================')
#print('Prediksi = ',a)
if a==[0]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
ALL")
    print("ACUTE LYMPHOCYTIC LEUKEMIA")
elif a==[1]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
AML")
    print("ACUTE MYELOID LEUKEMIA")
elif a==[2]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CLL")
    print("CHRONIC LYMPHOCYTIC LEUKEMIA")
elif a==[3]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CML")
    print("CHRONIC MYELOCYTIC LEUKEMIA")
```

```
print('')

b=svc2.predict([Data_Test])
#print('Data Test = ',Data_Test)
print('====================================')
print('===SVC Dengan Kernel Gaussian RBF===')
print('====================================')
#print('Prediksi = ',b)
if b==[0]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
ALL")
    print("ACUTE LYMPHOCYTIC LEUKEMIA")
elif b==[1]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
AML")
    print("ACUTE MYELOID LEUKEMIA")
elif b==[2]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CLL")
    print("CHRONIC LYMPHOCYTIC LEUKEMIA")
elif b==[3]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CML")
    print("CHRONIC MYELOCYTIC LEUKEMIA")
print('')

d=svc3.predict([Data_Test])
#print('Data Test = ',Data_Test)
print('================================')
print('===SVC Dengan Kernel Polynomial===')
print('================================')
#print('Prediksi = ',d)
if d==[0]:
```

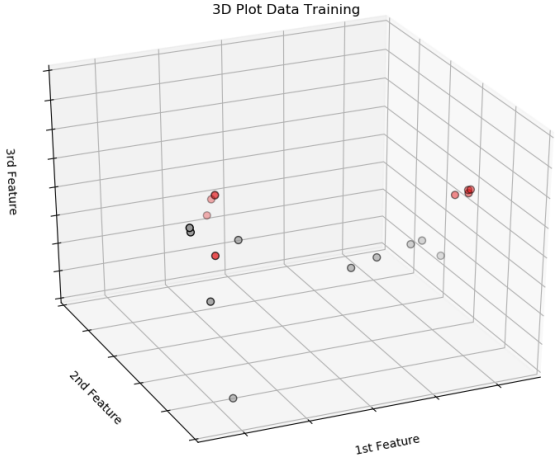
```
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
ALL")
    print("ACUTE LYMPHOCYTIC LEUKEMIA")
elif d==[1]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
AML")
    print("ACUTE MYELOID LEUKEMIA")
elif d==[2]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CLL")
    print("CHRONIC LYMPHOCYTIC LEUKEMIA")
elif d==[3]:
    print("Prediksi dari DNA/RNA tersebut adalah jenis leukemia
CML")
    print("CHRONIC MYELOCYTIC LEUKEMIA")
print(")
```

Lampiran 4. Data Pada Ruang 3 Dimensi

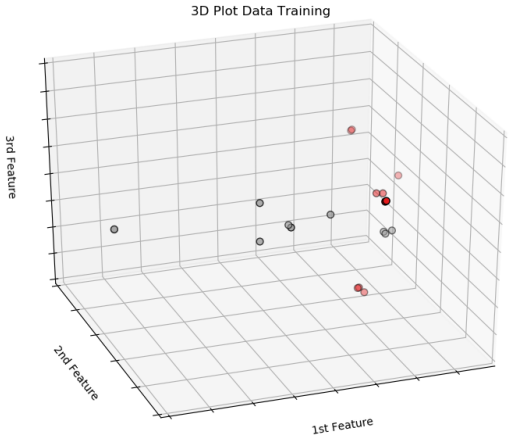
1. Plot Data Acute dan Chronic



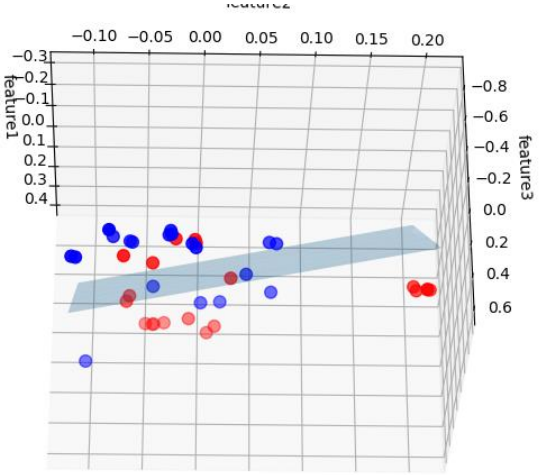
2. Plot Data ALL dan AML



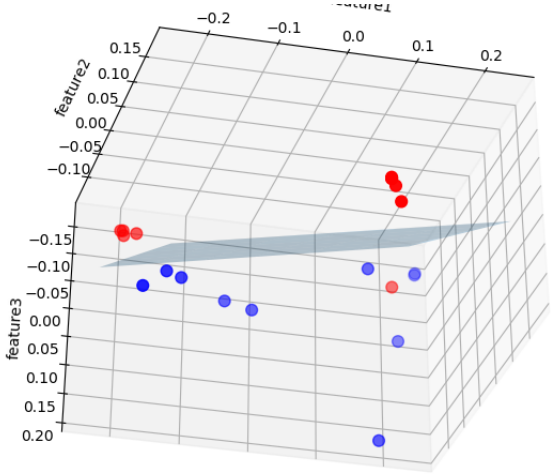
3. Plot Data CLL dan CML



4. SVC Acute dan Chronic



5. SVC ALL dan AML



6. SVC CLL dan CML