BAB 2

TINJAUAN PUSTAKA

2.1 Diabetes

Diabetes merupakan salah satu penyakit kronis yang ditandai dengan tingginya kadar gula darah atau glukosa. Sedangkan, glukosa sendiri merupakan sumber energi utama bagi sel tubuh manusia. Diabetes Melitus merupakan penyakit kelainan metabolik yang ditandai dengan hiperglikemia kronis serta kelainan metabolisme karbohidrat, lemak dan protein diakibatkan oleh kelainan sekresi insulin, kerja insulin maupun keduanya [11][12][13].

Beberapa faktor yang erat kaitannya dengan diabetes melitus antara lain: obesitas atau tingkat kegemukkan seseorang, hipertensi atau yang biasa disebut darah tinggi, riwayat keluarga atau faktor keturunan, dislipedimia atau kenaikan kadar lemak dalam darah, usia, riwayat persalinan, dan gaya hidup terutama dalam konsumsi makanan juga alkohol dan rokok [13].

Gejala diabetes melitus dapat dibedakan menjadi 2, yaitu akut dan kronik. Gejala pada diabetes melitus akut antara lain: poliphagia (banyak makan), polidipsia (banyak minum), poliuria (sering kencing terutama pada malam hari), nafsu makan bertambah namun berat badan turun dengan cepat (5-10 kg dalam waktu 2-4 minggu), dan mudah lelah. Sedangkan untuk gejala diabetes melitus kronik antara lain: kesemutan, kulit terasa panas atau seperti tertusuk tusuk jarum, rasa kebas di kulit, kram, kelelahan, mudah mengantuk, pandangan mulai kabur, gigi mudah goyah dan mudah lepas, kemampuan seksual menurun bahkan pada pria bisa terjadi impotensi, pada ibu hamil sering terjadi keguguran atau kematian janin dalam kandungan atau dengan bayi berat lahir lebih dari 4kg [13].

Tiga tipe diabetes utama dapat dibedakan menjadi 3 [14], yaitu :

1. Diabetes Tipe 1: dikenal juga sebagai diabetes remaja, tipe ini terjadi ketika tubuh gagal memproduksi insulin. Orang dengan diabetes tipe 1 ini bergantung pada insulin, yang berarti mereka harus mengonsumsi insulin buatan setiap hari agar tetap hidup.

1. Diabetes Tipe 2: diabetes tipe 2 ini memengaruhi cara tubuh menggunakan insulin, meskipun tubuh masih memproduksi insulin. Tidak seperti tipe 1, sel-sel dalam tubuh tidak meresponsnya seefektif dulu. Menurut National Institute of Diabetes and Digestive and Kidney Diseases, tipe ini adalah tipe diabetes yang paling umum, juga memiliki kaitan kuat dengan obesitas.
2. Diabetes gestasional: jenis ini terjadi pada wanita selama kehamilan ketika tubuh menjadi kurang sensitif terhadap insulin. Diabetes gestasional tidak terjadi pada semua wanita dan biasanya akan sembuh setelah melahirkan. Diabetes gestasional ini biasa disebut juga dengan diabetes mellitus.

2.2 Machine Learning

Machine learning atau dalam Bahasa Indonesia dikenal dengan Pembelajaran Mesin adalah aplikasi dari disiplin ilmu kecerdasan buatan (*Artificial Intelligence*). Konsep dari *machine learning* adalah memberikan kemampuan kepada komputer untuk belajar secara mandiri dari sekumpulan data yang sudah diberikan sebelumnya, dengan menggunakan algoritma dan model statistik untuk membuat prediksi. Fokus utama dari *machine learning* adalah untuk menemukan sebuah pola yang tepat dari suatu kumpulan data. Dengan adanya *machine learning* sebuah mesin atau komputer akan mampu mempelajari sejumlah data, sehingga dapat menghasilkan suatu model untuk melakukan proses *input-output* tanpa menggunakan kode program yang dibuat secara eksplisit [15]. Pada saat ini, *machine learning* telah diterapkan pada berbagai macam bidang, dan beberapa manfaat yang dapat dirasakan antara lain: deteksi *spam* pada email, deteksi berita

bohong (*hoax*), berbagai rekomendasi produk pada *e-commerce* atau aplikasi toko *online*, dan masih banyak lagi.

2.2.1 Cara Kerja Machine Learning

Secara sederhana, proses dari *machine learning* dibagi menjadi 3 bagian, yaitu: *input data*, abstraksi data, dan generalisasi. Untuk tahapan-tahapan atau alur kerja dari *machine learning* adalah sebagai berikut [16]:

A. Mengumpulkan Data

   Data yang digunakan beragam, mulai dari text dengan format *file* seperti .csv hingga berbentuk gambar seperti .jpg. Langkah ini merupakan dasar dari pembelajaran. Semakin banyak variasi, kepadatan dan volume data yang relevan, semakin baik prospek pembelajaran untuk mesin.

B. Mempersiapkan Data

   Setiap proses analitis berkembang dengan kualitas data yang digunakan. Pada tahapan ini, data dipersiapkan sedemikian rupa hingga memenuhi apa saja yang dibutuhkan. Tahapan ini diawali dengan analisis data untuk dapat menentukan kualitas data, kemudian melakukan perbaikan-perbaikan pada masalah yang ditemukan seperti hilangnya data dan lain-lain. Tahapan ini biasa disebut juga dengan *pre-processing*.

C. Melatih Sebuah Model

   Tahap ini melibatkan pemilihan algoritma dan data yang tepat dalam bentuk model. Data yang sudah disiapkan sebelumnya kemudian akan dibagi (*split*) menjadi dua bagian, yaitu: data latih (*train*) dan data uji (*test*). Bagian data latih (*training data*) akan digunakan untuk pengembangan model. Sedangkan bagian data uji (*testing data*) akan digunakan sebagai referensi.

D. Mengevaluasi Model

   Untuk menguji seberapa akurat model yang dibuat, bagian data uji (*testing data*) digunakan. Pada tahap ini akan ditentukan ketepatan dalam pemilihan algoritma berdasarkan hasil dari pengujian. Pengujian untuk ketepatan

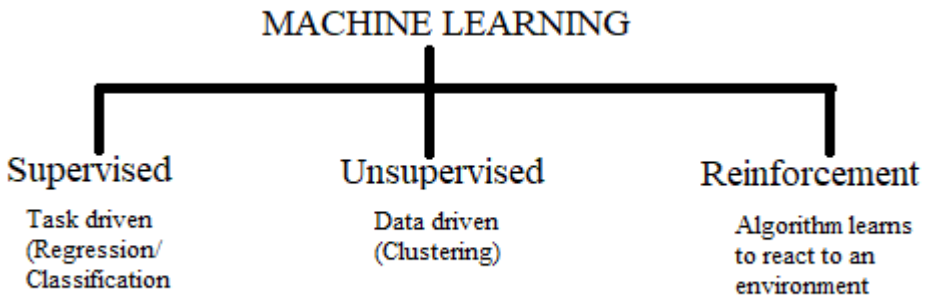
model ini dilakukan dengan cara melihat kinerja pada data yang tidak digunakan sama sekali selama pembuatan model.

E. Meningkatkan Kinerja

Tahap ini bisa jadi akan melibatkan pemilihan model yang lain, atau bisa juga memperkenalkan lebih banyak variabel untuk meningkatkan efisiensi.

2.2.2 Macam-Macam Algoritma Machine Learning

Secara umum algoritma *machine learning* dibagi menjadi 3 jenis, antara lain *Supervised* atau terarah, *Un-supervised* atau tak terarah, dan *Reinforcement* atau bala bantuan. Ada juga yang disebut dengan *Semi-supervised* yang mana merupakan gabungan dari algoritma *supervised* dan *unsupervised* [17]. Untuk lebih jelasnya, dapat dilihat pada Gambar 2.1 di bawah ini.



Gambar 2.1 Jenis-Jenis Algoritma Machine Learning

1. Model *Supervised Learning*

   Model *Supervised Learning* digunakan untuk melakukan prediksi atau hasil masa depan berdasarkan dari data historis. Untuk model prediktif, diberikan instruksi yang jelas dari awal, mulai dari apa yang perlu dipelajari, hingga bagaimana itu perlu dipelajari.

   Contoh algoritma: K-Nearest Neighbour, Naïve Bayes, Decision Tree, dan lainnya.

2. Model *Unsupervised Learning*

   Model *Unsupervised Learning* digunakan untuk melatih mesin agar dapat mencari dan menemukan pola yang sebelumnya tidak terdeteksi dalam

kumpulan data tanpa label yang sudah ada, dan dengan pengawasan manusia minimum.

Contoh algoritma: K-Means Clustering Algorithm.

3. Model *Reinforcement Learning*

   Model *Reinforcement Learning* merupakan pembelajaran mesin dimana mesin dilatih untuk mengambil keputusan secara spesifik atau tindakan di lingkungan dengan tujuan utama untuk memaksimalkan efisiensi. Model ini tidak memerlukan pasangan *input / output* berlabel yang disediakan, juga tidak membutuhkan tindakan sub-optimal untuk dikoreksi secara eksplisit. Fokus utama pada model ini adalah menemukan keseimbangan antara eksplorasi (wilayah yang belum dipetakan) dan eksploitasi (pengetahuan saat ini).

4. Model *Semi Supervised Learning*

   Model *semi supervised learning* merupakan pendekatan pembelajaran mesin yang menggabungkan sejumlah kecil data berlabel dengan sejumlah besar data tak berlabel selama pelatihan. Pembelajaran semi supervised berada di antara pembelajaran *unsupervised*(tanpa data pelatihan berlabel) dan pembelajaran yang *supervised*(dengan hanya data pelatihan berlabel).

2.3 Klasifikasi

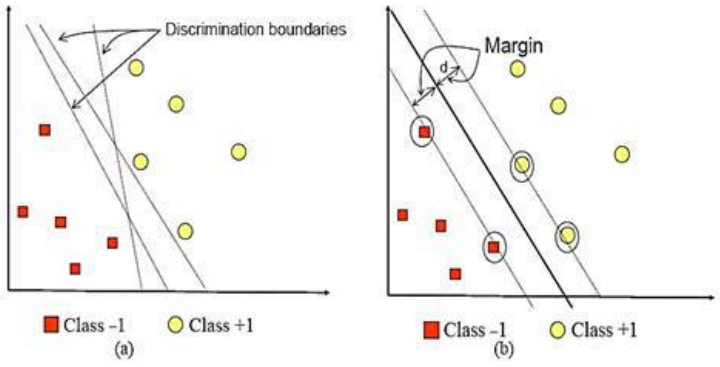
Klasifikasi merupakan sebuah proses untuk dapat menemukan atau menempatkan suatu objek atau konsep ke dalam satu set kategori berdasarkan objek atau konsep yang bersangkutan [1]. Dalam konsep pembelajaran mesin, klasifikasi merupakan bagian dari *supervised learning*, di mana tujuannya adalah untuk melakukan prediksi label kelas dari data baru yang belum dikenali sebelumnya, berdasarkan kumpulan data yang sudah ada. Dalam mencapai tujuan tersebut, proses klasifikasi membentuk suatu model yang mampu membedakan data ke dalam kelas – kelas yang berbeda berdasarkan aturan atau fungsi tertentu.

2.4 Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu algoritma pembelajaran mesin yang termasuk dalam jenis *supervised learning*. Algoritma ini

dikembangkan oleh Boser, Guyon, dan Vapnik, dan pertama kali diperkenalkan pada tahun 1992 dalam *Annual Workshop on Computational Learning Theory*. Konsep dasar dari metode ini adalah kombinasi dari teori-teori komputasi yang telah ada puluhan tahun sebelumnya, seperti *margin hyperplane*, *kernel*, dan konsep-konsep pendukung lainnya [18].

Support Vector Machines (SVM) masuk ke dalam kategori atau jenis model *supervised learning* karena pada proses pelatihannya diperlukan target pembelajaran tertentu. SVM juga merupakan algoritma yang bekerja menggunakan pemetaan *non-linear* untuk mengubah data pelatihan asli, menjadi bentuk dimensi yang lebih tinggi. Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* (garis pemisah) terbaik yang berfungsi sebagai pemisah dari dua buah *class* pada *input space*. Teknik SVM digunakan untuk mendapatkan fungsi pemisah (*hyperplane*) yang optimal untuk memisahkan observasi yang memiliki nilai variabel target yang berbeda. Untuk gambaran dari proses pencarian *hyperplane* dapat di lihat pada Gambar 2.2 di bawah ini.



Gambar 2.2 Proses Pencarian Hyperplane

2.5 Reduced Support Vector Machine

Reduced Support Vector Machine (RSVM) merupakan salah satu turunan dari algoritma pembelajaran mesin *Support Vector Machine* (SVM), sehingga algoritma ini masih termasuk ke dalam model *supervised learning*. RSVM pertama kali diperkenalkan oleh Lee dan Mangasarian pada tahun 2001. Tujuan dari RSVM ini adalah untuk mengatasi masalah kesulitan komputasi, dan untuk mengurangi

kompleksitas model saat dihadapkan dengan dataset yang berjumlah sangat besar [19].

Konsep dasar dari RSVM ini adalah melakukan proses klasifikasi hanya dengan menggunakan sebagian data yang dipilih secara acak. RSVM ini akan menghasilkan permukaan pemisah berbasis *kernel non-linier* yang membutuhkan sedikitnya 1% dari keseluruhan dataset untuk evaluasi eksplisitnya [20]. Untuk menghasilkan permukaan *non-linier* ini, seluruh dataset digunakan sebagai pembatas dalam masalah optimasi dengan sangat sedikit variabel yang sesuai dengan 1% dari data yang disimpan. Sisa data dapat dibuang setelah menyelesaikan masalah optimasi. Secara garis besar, tahapan atau langkah-langkah dalam algoritma RSVM adalah sebagai berikut [20]:

1) Pilih matriks subset acak $\bar{A} \in R^{\bar{m}xn}$ dari data matriks awal $A \in R^{mxn}$. Biasanya $\bar{m}$ sebesar 1% hingga 10% dari m.
2) Selesaikan persamaan SSVM yang telah dimodifikasi berikut dan diselesaikan dengan algoritma *Newton-Armijo*, di mana A' saja yang digantikan oleh $\bar{A}'$ dengan $\bar{D} \subset D$:

$$\min_{(\bar{u},\gamma)\in R^{\bar{m}+1}} \frac{v}{2}||p(e - D(K(A,\bar{A}')\bar{D}\bar{u} - e\gamma),\alpha)||_2^2 + \frac{1}{2}(\bar{u}'\bar{u} + \gamma^2) \quad \textbf{(2.1)}$$

3) Bidang pemisah dengan A' diganti dengan $\bar{A}'$ sebagai berikut:

$$K(x', \bar{A}')\bar{D}\bar{u} = \gamma \quad \textbf{(2.2)}$$

di mana $(\bar{u}, \gamma) \in R^{\bar{m}+1}$ adalah solusi unik dari persamaan 2.1, dan $x \in R^n$ adalah variabel ruang input bebas dari titik baru.

4) Titik input baru $x \in R^n$ diklasifikasikan ke dalam kelas +1 atau −1 tergantung pada fungsi berikut.

$$f(x) = (K(x', \bar{A}')\bar{D}\bar{u} - \gamma) \quad \textbf{(2.3)}$$

dengan f(x) adalah plus function yang masing-masing nilainya adalah +1 atau 0.

2.6 Pembersihan Data

Pembersihan data ata *cleaning data* merupakan salah satu tahapan awal atau biasa disebu dengan *pre-processing* yang dilakukan sebelum masuk ke tahap pembentukan model *machine learning*. Pembersihan data berisi berbagai proses yang bertujuan untuk melakukan perbaikan pada data yang akan diteliti [21]. Melakukan pembersihan data adalah salah satu hal yang penting, karena data yang masih mentah cenderung tidak siap dan perlu dikaji sedemikian rupa terlebih dahulu agar sesuai dengan kebutuhan sistem. Pada pembersihan data ini, kasus yang sering dijumpai adalah hilangnya sebagian data atau *missing values*. Maksud dari missing value adalah, adanya beberapa fitur yang tidak memiliki nilai informasi atau kosong, di mana fitur tersebut harusnya memiliki sebuah nilai tertentu dan mustahil memiliki nilai kosong.

2.7 Normalisasi Data

Tahap *pre-processing* merupakan salah satu tahapan yang penting dalam pembangunan sistem *machine learning*. Salah satu proses yang biasa dilakukan dalam tahap *pre-processing* adalah normalisasi data. Normalisasi data dilakukan agar nilai dari semua fitur yang ada pada dataset yang digunakan memiliki jangakauan yang sama. Secara singkat, normalisasi data akan menyeragamkan nilai dari fitur yang ada pada dataset yang akan digunakan (*scaling*). Untuk melakukan *scaling* tersebut, dapat digunakan teknik *min-max normalization* menggunakan persamaan berikut :

$$f(x_i) = \frac{x_i - min_A}{max_A - min_A}(max_B - min_B) + min_B \quad \textbf{..(2.4)}$$

2.8 Oversampling

Dalam mengatasi dataset yang tidak seimbang atau *imbalance* terdapat 2 cara, yaitu *Undersampling* dan *Oversampling*. Pada cara *Undersampling*, kelas yang dominan (jumlah data yang lebih banyak) akan dikurangi hingga jumlahnya sama seperti kelas minoritas (jumlah data yang lebih sedikit). Sedangkan pada *Oversampling*, kelas minoritas akan dibuatkan beberapa sampel tambahan sehingga

jumlah datanya akan menjadi sama dengan kelas yang dominan. Salah satu metode dalam *Oversampling* ini adalah *Synthetic Minority Oversampling Technique* (SMOTE). Penggunaan metode SMOTE memungkinkan akan adanya *overfitting*. *Overfiting* dapat terjadi karena adanya proses duplikasi data pada kelas minoritas, sehingga memungkinkan adanya data latih yang sama. SMOTE bekerja dengan memanfaatkan algoritma *K-Nearest Neighbor* (KNN) untuk membuat data sintetis. Metode SMOTE dimulai dengan

1. Memilih data acak dari kelas minoritas,
2. Menghitung jarak antar data dengan data minoritas yang sudah ditetapkan.
3. Menetukkan jumlah k terdekat,
4. Menciptakan data sintesis dengan persamaan berikut [22]:

$$Xsyn = Xi + (Xknn - Xi) * \delta \quad \textbf{..(2.5)}$$

Di mana:

- Xsyn adalah data sintesis yang akan diciptakan,
- Xi data yang akan direplikasi,
- Xknn data yang memiliki jarak terdekat dari data yang akan direplikasi,
- δ nilai random antara 0 dan 1.

2.9 Matriks Konfusi

Akurasi klasifikasi merupakan suatu ukuran ketepatan dalam melakukan klasifikasi yang menunjukkan performansi teknik klasifikasi secara keseluruhan. Semakin tinggi akurasi klasifikasi, maka artinya semakin baik pula performansi teknik klasifikasi [23]. Dalam melakukan pengukuran akurasi, dapat digunakan matriks konfusi atau *confusion matrix.* Matriks Konfusi adalah sebuah tabel yang memuat hasil klasifikasi, dengan 2 kelas, yaitu kelas aktual dan kelas prediksi yang masing-masing memiliki nilai positif dan negatif (dalam prediksi 2 kelas). Untuk penjelasan lebih lanjut, terdapat pada Tabel 2.1.

Tabel 2.1 Matriks Konfusi

Aktual	**Prediksi**	
	positif	**negatif**
positif	TP	FN
negatif	FP	TN

Keterangan:

- TP (*True Positive*) adalah jumlah prediksi benar pada kelas positif.
- FP : (*False Positive*) adalah jumlah prediksi salah pada kelas positif.
- FN : (*False Nefative*) adalah jumlah prediksi salah pada kelas negatif.
- TN : (True Negative) adalah jumlah prediksi benar pada kelas negatif.

2.10 K-Fold Cross Validation

Cross validation, atau dalam Bahasa Indonesia disebut dengan validasi silang adalah sebuah metode statistik yang digunakan untuk mengevaluasi kinerja dari suatu model atau algoritma. Teknik ini biasa digunakan untuk melakukan prediksi dan memperkirakan seberapa akurat sebuah model ketika dijalankan dalam praktiknya. Pada metode *cross validation*, data dipisahkan menjadi dua bagian, yaitu data latih dan data uji, dimana model atau algoritma tersebut dilatih oleh subset pelatihan dan divalidasi oleh subset validasi.

Salah satu metode *cross validation* yang biasa digunakan adalah *K-fold cross validation*. Pada metode ini dataset akan dibagi menjadi sejumlah K bagian dan eksperimen akan diulang sebanyak K pula. Metode validasi ini cocok digunakan untuk kasus data dengan jumlah sampel terbatas. Pada *K-fold cross validation*, data (K) dibagi ke dalam n bagian dengan jumlah yang sama (K1, K2, K3, ... , Kn). Kemudian, data K ke-n akan digunakan sebagai data uji, dan sisanya sebagai data latih. Proses ini diulangi sebanyak K bagian, dan didapatkan hasil akurasi klasifikasi dari hasil rata-rata dari setiap data *training* dan data *testing*. *K-folds* yang biasa digunakan sebanyak 3, 5, 10 dan 20.

Sebagai contoh, misalkan terdapat data yang berjumlah 120, jika digunakan nilai K=5, maka data tersebut akan dibagi menjadi 5 bagian dengan isi masing-masing sebanyak 24 data. Data pada partisi K ke-n akan digunakan sebagai data uji, sedangkan sisanya akan digunakan sebagai data latih. Untuk lebih jelasnya dapat di lihat pada ilustrasi Tabel 2.2.

Tabel 2.2 Ilustrasi Metode K-Folds Cross Validation

Eksperimen ke-	K1	K2	K3	K4	K5
1	Test	Train	Train	Train	Train
2	Train	Test	Train	Train	Train
3	Train	Train	Test	Train	Train
4	Train	Train	Train	Test	Train
5	Train	Train	Train	Train	Test

2.11 Unified Modelling Language

Dalam membangun aplikasi berorientasi objek, salah satu model perancangan yang digunakan adalah *Unified Modelling Language* (UML). UML adalah sebuah standar bahasa berdasarkan grafik/gambar yang berguna untuk memvisualisasi, menspesifikasikan, membangun, dan mendokumentasikan sebuah pengembangan sistem perangkat lunak *software* berbasis OO *(Object-Oriented*) [24]. Dalam UML terdapat berbagai macam diagram yang digunakan untuk memodelkan aplikasi berorientasi objek, antara lain: *Use Case, Activity Diagram, Sequence Diagram, dan Class Diagram*. UML digunakan agar rancangan sistem lebih mudah dipahami oleh banyak pihak yang terlibat dalam proses pengembangannya.

2.11.1 Use Case Diagram

Use case diagram digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri dari diagram untuk *use case* dan *actor*, di mana *actor* merupakan representasi dari orang yang akan mengoperasikan orang yang berinteraksi dengan sistem, namun bisa juga sebuah sistem yang lain [24].

2.11.2 Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas yang ada pada sistem yang dirancang, mulai dari awal dari masing-masing alir berawal, *decision* atau pilihan yang mungkin terjadi, hingga bagaimana aliran tersebut berakhir. Keunggulan dari *activity diagram* adalah lebih mudah dimengerti jika dibandingkan skenario [24].

2.11.3 Class Diagram

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas [24]. *Class diagram* berfungsi untuk memvisualisasikan struktur kelas-kelas dari suatu sistem. Suatu Class memiliki tiga bagian pokok, yaitu nama (dan *stereotype*), atribut, dan metoda yang dapat dilakukannya. Suatu *class* dapat memiliki hubungan antara satu sama lain melalui berbagai cara, antara lain: *associated* (terhubung satu sama lain), *dependent* (satu *class* bergantung atau menggunakan class yang lain), *specialized* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit). Di dalam sebuah sistem biasanya terdapat beberapa *class diagram*.

2.11.4 Sequence Diagram

Sequence diagram menggambarkan interaksi objek dalam sistem yang disusun berdasarkan urutan waktu. *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan sesuai dengan waktu terjadinya dalam pesan yang terurut [24].

2.12 Python

Python adalah salah satu bahasa pemrograman komputer yang populer dan sering digunakan terutama dalam melakukan perhitungan. Bahasa pemrograman ini dapat dijalankan hampir di semua platform, seperti *Linux, Windows,* dan *Machintos*. Pada bahasa pemrograman *Python*, deklarasi suatu variabel dapat dilakukan secara langsung tanpa menyebutkan tipe datanya.

Bahasa pemrograman *Python* diciptakan oleh Guido van Rossum, dan diperkenalkan pertama kali pada Centrum Wiskunde & Informatica (CWI) di Belanda pada awal tahun 1990-an. Bahasa pemrograman yang terinspirasi dari bahasa pemrograman ABC ini pertama kali dikembangkan pada awal tahun 1990 oleh Guido van Rossum di Stiching Mathematisch Centrum. Pada tahun 1995, pengembangan bahasa pemrograman *Python* dilakukan di Corporation for National Research Initiatives. Saat ini *Python Software Foundation* bertugas sebagai pengembang *Python* [24].


UNIVERSITAS KOMPUTER INDONESIA
UNIKOM