BAB II
LANDASAN TEORI

2.1 Pengertian Data

Data adalah fakta yang dapat dicatat dan memiliki arti. Data adalah sesuatu yang mewakilkan objek dan peristiwa yang memiliki arti dan sangat penting bagi pemakai (*user*). Objek dapat berupa gambar, suara, huruf, angka, bahasa, ataupun simbol-simbol lainnya yang bisa digunakan sebagai bahan untuk melihat lingkungan, obyek, kejadian ataupun suatu konsep. Oleh karena itu, suatu data belum dapat berbicara banyak sebelum diolah lebih lanjut. Syarat dari sebuah data dianggap baik dan berguna adalah sebagai berikut: (Elmasri dan Navathe, 2004)

1. Data harus objektif, artinya data itu menggambarkan seperti apa adanya.
2. Data harus mewakili.
3. Data harus mempunyai kesalahan baku (*standar error*) yang kecil (apabila data merupakan suatu perkiraan). Kesalahan baku merupakan simpangan baku suatu perkiraan dan digunakan untuk mengukur tingkat ketelitian. Makin kecil kesalahan baku suatu perkiraan, makin telitilah perkiraan tersebut.
4. Data harus tepat waktu, syarat tepat waktu penting sekali jika data tersebut akan digunakan untuk mengontrol pelaksanaan dan perencanaan sehingga persoalan yang terjadi dapat diketahui untuk segera diatasi, dikoreksi dan dipecahkan.
5. Data harus mempunyai hubungan dengan persoalan yang akan dipecahkan.

Proses pengolahan data terbagi menjadi tiga tahapan, yang disebut dengan siklus pengolahan data (*Data Processing Cycle*). Siklus tersebut yaitu: (Pralystia, 2009)

1. Pada tahapan input dilakukan proses pemasukan data ke dalam komputer lewat media input.
2. Pada tahapan processing dimana dilakukannya proses pengolahan data yang sudah dimasukkan pada tahapan input. Proses ini dilakukan oleh alat

pemroses yang dapat berupa proses perhitungan, perbandingan, pengendalian, atau pencarian di storage.

3. Pada tahapan output yaitu dilakukan proses menghasilkan keluaran dari hasil pengolahan data ke alat output yaitu berupa informasi.

2.2 Data Mining

Pada sub bab ini akan dibahas mengenai pengertian data mining, tugas data mining dan tahapan data mining.

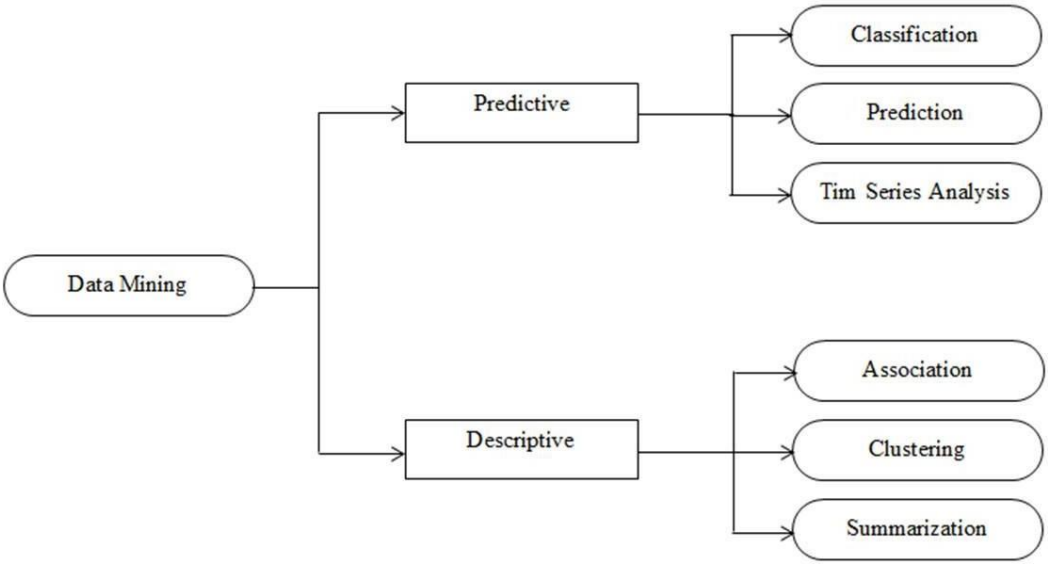
2.2.1 Pengertian Data Mining

Data mining adalah proses yang menggunakan satu atau lebih teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstraks pengetahuan secara otomatis. Data mining merupakan serangkaian proses yang bertujuan untuk menemukan secara manual nilai tambah dari sekumpulan data berupa pengetahuan yang sebelumnya tidak diketahui (Wibowo & Rohman, 2022).

Data mining memproses pengumpulan, pengolahan, dan analisis data untuk menemukan pola dan hubungan yang tersembunyi dalam data. Proses ini melibatkan teknik-teknik dari pembelajaran mesin, statistik, dan database sistem. Beberapa tujuan dari data mining asdalah untuk mengekstrak informasi penting dari data, memprediksi perilaku pelanggan, meningkatkan efisiensi bisnis, dan sebagainya. Data mining dapat digunakan di berbagai bidang, seperti bisnis, perpustakaan akademik, dan penelitian sosial. Beberapa teknik data mining yang umum digunakan adalah *machine learning*, *statistical analysis*, dan *data management task*. Data mining dapat membantu organisasi dalam membuat keputusan yang lebih baik dan efektif, serta meningkatkan kinerja bisnis. Namun, data mining juga memiliki beberapa tantangan, seperti pemilihan atribut yang tepat dan keamanan data.

2.2.2 Tugas Data Mining

Tugas data mining secara garis besar dibagi menjadi dua kategori utama yaitu tugas prediksi dan tugas deskripsi. Berikut ini adalah analisisnya:



Gambar 2.1 Tugas data mining

Tujuan tugas prediksi adalah memprediksi nilai dari atribut tertentu berdasarkan nilai dari atribut lainnya. Atribut yang diprediksi dikenal sebagai target atau *dependent variable*, sedangkan atribut yang digunakan untuk membuat prediksi disebut penjelas atau *independent variable*. Metode yang termasuk *predictive* data mining yaitu: (Tan, Steinbach, Karpatne, 2018)

1. Klasifikasi: pembagian data ke dalam beberapa kelompok atau kelas yang telah ditentukan sebelumnya.
2. Regresi: memerakan data ke suatu *prediction variable*.
3. *Time Series Analysis*: pengamatan perubahan nilai atribut dari waktu ke waktu.

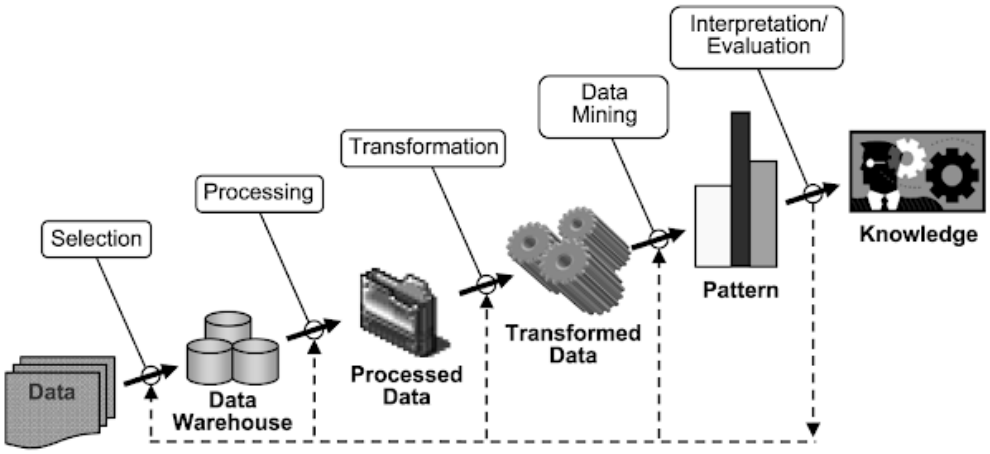
Sedangkan tujuan tugas deskripsi adalah memperoleh pola untuk menyimpulkan hubungan di dalam data. Tugas deskriptif merupakan tugas data mining yang sering dibutuhkan pada teknik postprocessing untuk melakukan validasi dan menjelaskan hasil proses data mining. Metode yang termasuk *descriptive* data mining yaitu: (Tan, Steinbach, Karpatne, 2018)

1. Clustering: mengelompokan beberapa objek yang serupa ke dalam sebuah cluster, dan yang tidak serupa ke cluster yang lain.

2. *Associationrules*: identifikasi hubungan antara data yang satu dengan yang lainnya.
3. *Summarization*: pemetaan data ke dalam subset dengan deskripsi sederhana.
4. *Sequence Discovery*: identifikasi pola sekuensial dalam data.

2.2.3 Tahapan Data Mining

Data mining sering disebut dengan *Knowledge Discovery in Database* (KDD). KDD adalah kegiatan yang meliputi pengumpulan, pemakaian data historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar. Tahapan dari proses *Knowledge Discovery in Database* (KDD) adalah:



Gambar 2.2 Tahapan *Knowledge Discovery in Database* (KDD)

1. Data Selection

   Pemilihan data dari sekumpulan data operasional yang perlu dilakukan sebelum tahap penggalian informasi dalam KDD dimulai.

2. Preprocessing

   Sebelum proses data mining dapat dilaksanakan, perlu dilakukan proses cleaning dengan tujuan untuk membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data. Dilakukan juga proses *enrichment*, yaitu proses "memperkaya" data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan untuk KDD.

3. Transformation

   Proses coding pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses data mining. Proses coding dalam KDD merupakan proses kreatif dan sangat tergantung pada jenis atau pola informası yang akan dicari dalam database

4. Data mining

   Proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu

5. Interpretation Evaluation

   Pola informasi yang dihasilkan dari proses data mining perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Tahap ini mencakup pemeriksaan apakah pola yang ditemukan bertentangan dengan fakta atau hipotesa yang ada sebelumnya atau tidak.

2.3 *Machine Learning*

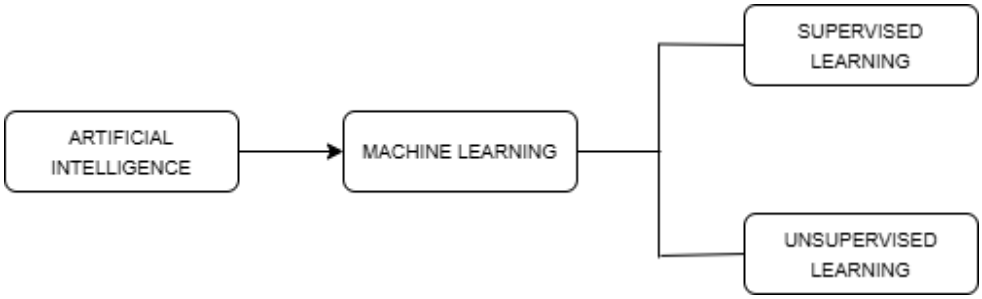
Sub bab ini akan dibahas mengenai pengertian dan metode *machine learning*.

2.3.1 Pengertian *Machine Learning*

Machine learning adalah bagian dari kecerdasan buatan (*Artificial Intelligence*) yang memungkinkan komputer berperilaku seperti manusia dengan membuat sistem yang memiliki kemampuan belajar secara otomatis tanpa diprogram. *Machine learning* menggunakan teknik yang bekerja pada data besar sebagai input untuk dianalisis sehingga menemukan pola tertentu untuk melakukan pembelajaran sehingga sistem dapat melakukan analisis dengan benar (Huang et al., 2004).

2.3.2 Metode *Machine Learning*

Metode merupakan prosedur, teknik, atau langkah untuk melakukan sesuatu untuk mencapai tujuan tertentu. Berdasarkan teknik pembelajarannya, *machine learning* dibedakan menjadi beberapa metode yang di antaranya yaitu *supervised learning* dan *unsupervised learning* (Retnoningsih & Pramudita, 2020).



Gambar 2.3 Bagan *Artificial Intelligent* & *Machine Learning*

1. *Supervised learning* merupakan metode yang menggunakan dataset (data training) yang telah dilabeli (*labeled data*) sehingga sistem mampu mengidentifikasi label input dengan fitur yang dimiliki untuk selanjutnya dilakukan prediksi maupun klasifikasi. Metode ini dapat memberikan target terhadap *output* yang dilakukan dengan membandingkan pengalaman belajar di masa lalu.
2. *Unsupervised learning* adalah metode yang menarik kesimpulan berdasarkan dataset dari input data *labeled response*. Metode ini dapat membantu menemukan struktur atau pola tersembunyi pada data yang tidak memiliki label.

2.4 Klasifikasi

Metode klasifikasi adalah jenis analisis data yang dapat membantu orang memprediksi label kelas sampel harus diklasifikasikan. Berbagai macam teknik klasifikasi telah diusulkan dalam bidang-bidang seperti pembelajaran mesin, sistem pakar dan statistik. Sejak beberapa tahun yang lalu peneliti telah mengembangkan repositori software untuk lebih dalam megenal data, dari dua puluh metode klasifikasi. *Naïve Bayes* merupakan salah satu metode klasifikasi terbaik (Sidik et al., 2020).

2.5 *Naïve Bayes*

Naive Bayes merupakan suatu algoritma yang dikemukakan oleh *Thomas Bayes* seorang ilmuan Inggris yang dapat memprediksi peluang masa depan berdasarkan pengalaman sebelumnya sehingga dikenal sebagai *Teorema Bayes* dan

merupakan algoritma yang terdapat pada teknik klasifikasi probabilitas dan statistik (Fadlan & Ningsih, 2018). *Naïve Bayes* adalah tata cara pengklasifikasian probabilistik simpel. Model ini hendak menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi serta campuran nilai dari dataset yang diberikan. Berikut persamaan dari *Teorema Bayes*:

$$P(H|X) = \frac{P(X|H).P(H)}{P(X)} \quad (1)$$

Keterangan :

X	=	data class yang belum diketahui
H	=	hipotesis data
P(H|X)	=	probabilitas hipotesis H berdasar X
P(X|H)	=	probabilitas hipotesis X berdasar H
P(H)	=	probabilitas hipotesis H
P(X)	=	probabilitas X

Untuk menjelaskan teorema *Naïve Bayes*, perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang dianalisis tersebut. Karena itu, teorema bayes di atas disesuaikan sebagai berikut:

$$P(C|F_1 \dots F_n) = \frac{P(C|H)\ P(F_1 \dots F_n|C)}{P(F_1 \dots F_n)}$$

Dimana variable C merepresentasikan kelas, sementara variable $F_1 \dots F_n$, merepresentasikan karakteristik untuk klasifikasi. Rumus tersebut menjelaskan bahwa peluang masuknya sampel karakteristik tertentu dalam kelas C (*Posterior*) adalah peluang munculnya kelas C (prior), dikali dengan peluang kemunculan karakteristik - karakteristik sampel pada kelas C (*likelihood*), dibagi dengan peluang kemunculan karakteristik- karakteristik sampel secara keseluruhan (*evidence*). Sehingga rumus ini dapat ditulis secara sederhana sebagai berikut:

$$Posterior = \frac{Prior \times likelihood}{evidence}$$

Nilai *evidence* selalu tetap untuk setiap kelas pada satu sampel. Nilai dari posterior tersebut nantinya akan dibandingkan dengan nilai-nilai *posterior* kelas lainnya untuk menentukan ke kelas apa suatu sampel akan diklasifikasikan. Penjabaran lebih lanjut rumus Bayes tersebut dilakukan dengan menjabarkan ($C|F_1, \dots F_n$) menggunakan aturan perkalian sebagai berikut:

$$\begin{aligned} P(C|F_1, \dots, F_n) &= P(C)\, P(F_1, \dots, F_n|C) \\ &= P(C)\, P(F_1|C)\, P(F_2, \dots, F_n|C, F_1) \\ &= P(C)\, P(F_1|C)\, P(F_2|C, F_1) P(F_3, \dots, F_n|C, F_1, F_2) \\ &= P(C)\, P(F_1|C)\, P(F_2|C, F_1)\, P(F_3|C, F_1, F_2) \\ &\quad P(F_4, \dots, F_n|C, F_1, F_2, F_3) \\ &= P(C)\, P(F_1|C)\, P(F_2|C, F_1)\, P(F_3|C, F_1, F_2) \dots \\ &\quad P(F_n|C, F_1, F_2, F_3, \dots, F_{n-1}) \end{aligned}$$

Dapat dilihat bahwa hasil penjabaran tersebut menyebabkan semakin banyak dan semakin kompleksnya factor-faktor syarat yang mempengaruhi nilai probabilitas, yang hampir mustahil untuk dianalisa satu persatu. Akibatnya, perhitungan tersebut menjadi sulit untuk dilakukan. Disinilah digunakan asumsi independensi yang sangat tinggi (*naif*), bahwa masing-masing *prior* ($F_1, F_2 \dots F_n$) saling bebas (*independen*) satu sama lain. Dengan asumsi tersebut, maka berlaku suatu persamaan sebagai berikut:

$$P(P_i|F_j) = \frac{P(F_i \cap F_j)}{P(F_j)} = \frac{P(F_i)P(F_j)}{P(F_j)} = P(F_i)$$

Untuk $i \neq j$, sehingga

$$P(F_i|C, F_i) = P(F_i|C)$$

Dari persamaan ini dapat disimpulkan bahwa asumsi independensi naif tersebut membuat syarat peluang menjadi sederhana, sehingga perhitungan menjadi mungkin untuk dilakukan. Selanjutnya, penjabaran P($C|F_1, \dots, F_n$) dapat

disederhanakan menjadi:

$$P(C|F_1, \dots, F_n) = P(C)\, P(F_1|C)\;\; P(F_2|C)\;\; P(F_3|C)\;\; P(F_4|C) \dots \;\; P(F_n|C)$$
$$= P(C) \prod_{i=1}^{n} \; P(F_i|C)$$

Persamaan ini merupakan model dari teorema *Naïve Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi.

2.6 Ketepatan Klasifikasi

Naïve Bayes pada proses klasifikasi diperlukan adanya evaluasi kinerja dari sistem klasifikasi. Evaluasi kinerja dari model yang sudah dibangun oleh algoritma klasifikasi *Naïve Bayes* dapat dilakukan dengan menjumlahkan data testing yang di prediksi benar (akurasi) atau salah (*error rate*) oleh model tersebut. Umumnya, evaluasi kinerja klasifikasi ini menggunakan matriks konfusi yang berupa tabel pencatat hasil kerja klasifikasi. Isi dari tabel ini adalah jumlah data uji yang terklasifikasi benar dan salah. Tabel matriks konfusi dapat dilihat di Tabel 2.1. (Hidayatulloh et al., 2023)

Tabel 2.1 Matriks Konfusi

Predicted Class	Actual Class	
	TRUE	*FALSE*
TRUE	*TP*	*FP*
FALSE	*FN*	*TN*

Berdasarkan tabel matriks konfusi di atas:

1. *True Positives* (TP) merupakan jumlah dari record data positif yang tepat diklasifikasikan sebagai nilai positif.
2. *False Positives* (FP) merupakan jumlah dari record data negatif yang diklasifikasikan sebagai nilai positif.
3. *False Negatives* (FN) merupakan jumlah dari record data positif yang diklasifikasikan sebagai nilai negative.
4. *True Negatives* (TN) merupakan jumlah dari record data negatif yang diklasifikasikan tepat sebagai nilai negative.

Berikut ini beberapa ukuran yang digunakan untuk mengukur ketepatan klasifikasi:

1. *Apparent Error Rate* (APER)

   Apparent Error Rate (APER) merupakan prosedur evaluasi untuk mengetahui kesalahan oleh fungsi klasifikasi. Suatu metode dikatakan memiliki tingkat akurasi yang baik apabila memiliki nilai APER yang kecil dan tingkat akurasi yang tinggi. Berikut adalah rumus menghitung APER :

$$APER = \frac{FP + FN}{TP + FP + TN + FN} \times 100\% \qquad (2)$$

2. Akurasi

   Nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Akurasi menunjukkan tingkat akurasi model dalam mengklasifikasikan data dengan benar.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (3)$$

2.7 Rapid Miner

RapidMiner merupakan perangakat lunak yang bersifat terbuka (*open source*). RapidMiner adalah sebuah solusi analisis prediksi, data mining, dan text mining yang menggunakan macam-macam teknik prediksi dan deskriptif untuk membuat keputusan yang paling baik bagi penggunanya (Prismas et al., 2017). RapidMiner menggunakan berbagai teknik deskriptif dan prediksi dalam memberikan wawasan kepada pengguna sehingga dapat membuat keputusan yang paling baik. RapidMiner memiliki kurang lebih 500 operator data mining, termasuk operator untuk input, output, data preprocessing dan visualisasi. RapidMiner merupakan software yang berdiri sendiri untuk analisis data dan sebagai mesin data mining yang dapat diintegrasikan pada produknya sendiri. Ditulis dengan menggunakan bahasa java sehingga dapat bekerja di semua sistem operasi.

Sebelumnya RapidMiner bernama YALE (*Yet Another Learning Environment*), dimana versi awalnya mulai dikembangkan pada tahun 2001 oleh RalfKlinkenberg, Ingo Mierswa, dan Simon Fischer di Artificial Intelligence Unit

dari University of Dortmund. RapidMiner didistribusikan di bawah lisensi AGPL (*GNU Affero General Public License*) versi 3. Hingga saat ini telah ribuan aplikasi yang dikembangkan mengunakan RapidMiner di lebih dari 40 negara. RapidMiner sebagai software open source untuk data mining tidak perlu diragukan lagi karena software ini sudah terkemuka di dunia. RapidMiner menempati peringkat pertama sebagai Software data mining pada polling oleh *KDnuggets*, sebuah portal data-mining pada 2010-2011.

RapidMiner menyediakan GUI (*Graphic User Interface*) untuk merancang sebuah pipeline analitis. GUI ini akan menghasilkan file XML (*Extensible Markup Language*) yang mendefenisikan proses analitis keingginan pengguna untuk diterpkan ke data. File ini kemudian dibaca oleh RapidMiner untuk menjalankan analis secara otomatis.

RapidMiner memiliki beberapa sifat sebagai berikut: (Sudarsono et al., 2021)

1. Ditulis dengan bahasa pemrograman Java sehingga dapat dijalankan di berbagai sistem operasi.
2. Proses penemuan pengetahuan dimodelkan sebagai operator trees
3. Representasi XML internal untuk memastikan format standar pertukaran data.
4. Bahasa scripting memungkinkan untuk eksperimen skala besar dan otomatisasi eksperimen.
5. Konsep multi-layer untuk menjamin tampilan data yang efisien dan menjamin penanganan data.
6. Memiliki GUI, command line mode, dan Java API yang dapat dipanggil dari program lain.

Beberapa Fitur dari RapidMiner, antara lain:

1. Banyaknya algoritma data mining, seperti decision tree dan *self-organization map*.
2. Bentuk grafis yang canggih, seperti tumpang tindih diagram histogram, tree chart dan 3D *scatter plots*.
3. Banyaknya variasi plugin, seperti text plugin untuk melakukan analisis teks.

4. Menyediakan prosedur data mining dan machine learning termasuk: ETL (*extraction, transformation, loading*), data preprocessing, visualisasi, modelling dan evaluasi.
5. Proses data mining tersusun atas operator-operator yang nestable, dideskripsikan dengan XML, dan dibuat dengan GUI.
6. Mengintegrasikan proyek data mining Weka dan statistika R.

2.8 PHP

Hypertext Preprocessor (PHP) yaitu bahasa pemograman web server-side yang bersifat *open source*. PHP merupakan script yang terintegrasi dengan HTML dan berada pada server. PHP adalah script yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Semua script PHP dieksekusi pada server di mana script tersebut dijalankan. Kode PHP diawali dengan <? php diakhiri dengan ?>. Pasangan kedua kode inilah yang berfungsi sebagai tag kode PHP. Berdasarkan tag inilah, pihak server dapat memahami kode PHP dan kemudian memprosesnya. Hasilnya dikirim ke browser. (Lavarino, 2016)

Penemu bahasa pemrograman ini adalah Rasmus Lerdorf, yang bermula dari keinginan sederhana Lerdorf untuk mempunyai alat bantu dalam memonitor pengunjung yang melihat situs web pribadinya. Inilah sebabnya pada awal pengembangannya, PHP merupakan singkatannya dari *Personal Home Page tools*, sebelum akhirnya menjadi *Hypertect Preprocessor*.

2.9 *Framework* Codeigniter

CodeIgniter adalah sebuah *framework* PHP yang digunakan untuk pengembangan aplikasi web. Sebagai aplikasi *open source*, Codeigniter menyediakan sebuah *framework* untuk membangun website dinamis menggunakan PHP. Dengan menggunakan Codeigniter proses pengembangan aplikasi web dapat cepat diselesaikan. Seorang *developer web* tidak perlu lagi menghabiskan waktu untuk menulis kode perintah program dari awal karena Codeigniter sudah menyediakan berbagai macam *library* dan fungsi-fungsi umum yang dibutuhkan

oleh *developer* (Prismas et al., 2017).

Sebuah *Framework* adalah kerangka konseptual yang berisi kumpulan fungsi, prosedur, dan kelas yang siap digunakan untuk memecahkan suatu masalah dalam pemrograman. *Framework* terdiri dari kumpulan perintah atau fungsi dasar yang membentuk aturan-aturan tertentu dan saling berinteraksi satu sama lain sehingga dalam pembuatan aplikasi website, harus mengikuti aturan dari framework tersebut. Dengan menggunakan *framework* (dalam hal ini *framework* PHP), tidak perlu memikirkan kode perintah/fungsi dasar dari aplikasi website, seperti bagaimana mengambil data dari database untuk ditampilkan. Aspek-aspek pendukung lainnya seperti koneksi database, validasi form, GUL, dan keamanan, telah disediakan oleh *framework* sehingga jumlah baris kode yang di buat jauh lebih sedikit. Jadi, keuntungan yang dapat diperoleh dari penggunaan framework adalah:

1. Waktu pembuatan aplikasi website jauh lebih singkat.
2. Kode aplikasi website menjadi lebih mudah dibaca, karena sedikit dan sifatnya pokok. Detailnya adalah kode dari framework dan ini mungkin tidak perlu dipikirkan.
3. Website menjadi lebih mudah diperbaiki, karena tidak perlu fokus ke semua komponen kode website, terutama kode sistem *framework*.
4. Tidak perlu lagi membuat kode penunjang aplikasi website seperti koneksi database, validasi form, GUI, dan keamanan.

Framework ini memiliki struktur organisasi data *source code* yang menggunakan metode MVC, yaitu:

1. Model

   Model merupakan sebuah objek yang berperan dalam sistem untuk melakukan manipulasi data, komputasi, atau menjalankan algoritma tertentu. Semua logika bisnis biasanya diimplementasikan di dalam model. Model adalah bagian dari kode program yang menangani query atau interaksi dengan database. Isi dari model biasanya berupa fungsi-fungsi yang langsung berhubungan dengan database untuk memanipulasi data seperti memasukkan data, pembaruan data, hapus data, dan sebagainya, namun tidak dapat berhubungan langsung dengan bagian view.

2. View

   View yaitu sebuah komponen yang menampilkan hasil yang dihasilkan oleh Controller dengan bantuan model. Pada aplikasi web, bagian ini biasanya berupa file HTML. View tidak memiliki akses langsung terhadap model. View adalah bagian kode program yang mengatur tampilan website. Pada aplikasi web, bagian View biasanya berupa file template HTML, yang diatur oleh controller. Bagian ini tidak memiliki akses langsung terhadap bagian model namun berhubungan langsung dengan controller. View berfungsi untuk menerima dan merepresentasikan data kepada pengguna. Jadi, bisa dikatakan bahwa View merupakan halaman web.

3. Controller

   Bagian ini menjadi penghubung antara bagian model dengan view. Controller berperan sebagai jembatan antara keduanya dengan menyimpan perintah-perintah yang berfungsi untuk memproses data dan mengirimkannya ke halaman web. Controller terdiri dari potongan-potongan kode yang menerima permintaan dari pengguna, kemudian menentukan tindakan apa yang akan diproses oleh sistem.

2.10 AJAX

AJAX adalah singkatan dari *Asynchronous JavaScript and XML*. AJAX merupakan sebuah fitur yang digunakan untuk membantu pembangunan webiste. Sesuai dengan namanya, AJAX mengombinasikan JavaScript dan XML (*eXtensible Markup Language*) yang bekerja bersama. Tujuan utama AJAX adalah untuk membangun sebuah website yang lebih responsif dan cepat tanggap, dengan melakukan pertukaran data dalam ukuran lebih kecil dengan server secara *asynchronous*. Pertukaran data dalam ukuran lebih kecil ini berarti bahwa saat sebuah website melakukan perubahan data, hanya sebagian halaman yang akan ditampilkan, sehingga tidak memaksa browser untuk kemudian melakukan loading seluruh halaman website. Teknik ini dirancang untuk mempercepat dan mempermudah kegiatan browsing. (Tahir, 2018)

Adapun fungsi AJAX dapat dijelaskan sebagai berikut: (Hariadi, 2010)

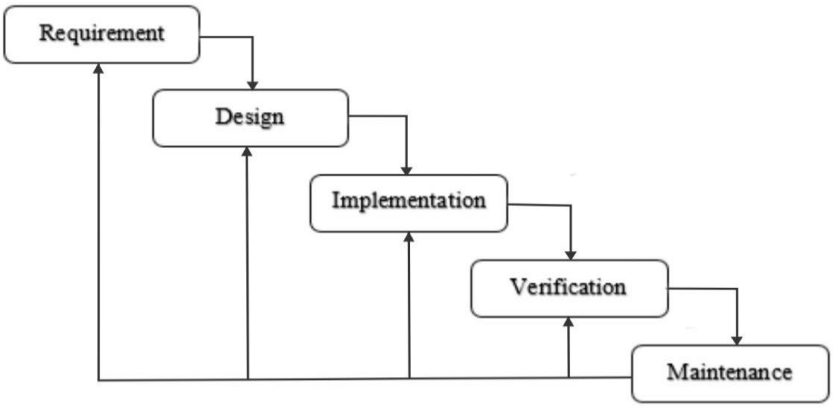
1. Aplikasi internet memiliki berbagai keuntungan dibandingkan aplikasi desktop. Aplikasi internet dapat menjangkau pengguna secara luas, mudah untuk instalasi dan support, serta mudah untuk dikembangkan. Namun, aplikasi internet tidak selalu se-kaya fitur dan *se-user-friendly* aplikasi desktop tradisional. Dengan penggunaan AJAX, aplikasi internet dapat dibuat lebih kaya fitur dan lebih *user-friendly* layaknya aplikasi desktop.
2. Mendapatkan atau mengirimkan informasi dari/ke sebuah database atau sebuah file di server menggunakan Javascript tradisional, diperlukan pembuatan form HTML. Pengguna harus mengklik tombol "Submit" untuk mengirimkan atau mendapatkan informasi, menunggu respon dari server, kemudian sebuah halaman baru akan dimuat dengan hasilnya. Dikarenakan server mengembalikan halaman baru setiap kali pengguna mengirimkan input, aplikasi web tradisional dapat berjalan lambat dan tidak *user-friendly*.
3. Dengan AJAX, Javascript akan berkomunikasi secara langsung dengan server, melalui objek JavaScript *XMLHttpRequest*.
4. Dengan objek *XMLHttpRequest*, halaman web dapat membuat permintaan ke, dan mendapatkan sebuah respons dari server web secara langsung.

2.11 Metode Pengembangan Sistem

Metode pengembangan sistem adalah serangkaian aktivitas, metode, praktik terbaik, dan alat yang digunakan untuk mengembangkan dan merawat sistem informasi atau perangkat lunak. Beberapa metode pengembangan sistem yang umum digunakan antara lain:

2.11.1 *Waterfall*

Metode *Waterfall* adalah suatu proses pengembangan perangkat lunak berurutan, di mana kemajuan dipandang sebagai terus mengalir ke bawah (seperti air terjun) melewati fase-fase perencanaan, pemodelan, implementasi (konstruksi), dan pengujian. Dalam pengembangannya metode *waterfall* memiliki beberapa tahapan yang runtut: *requirement* (analisi kebutuhan), desain sistem (*system design*), *coding & testing*, penerapan program, pemeliharaan. (Sasmito, 2017)



Gambar 2.4 Metode *Waterfall*

1. *Requirement* (analisis kebutuhan)

   Dalam langkah ini merupakan analisa terhadap kebutuhan sistem.Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study literatur*. Seseorang sistem analisis akan menggali informasi sebanyak-banyaknya dari user sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh user tersebut. Tahapan ini akan menghasilkan *document user reguipment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan sistem analisis untuk menerjemahkan ke dalam bahasa pemrograman.

2. *Design System*

   Proses design akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat koding. Proses ini terfokus pada: struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) procedural. Tahapan ini akan mengasilkan dokumen yang disebut software reguipment. Dokumen inilah yang akan digunakan programmer untuk melakukan aktivitas pembuatan sistemnya.

3. *Coding & Testing* (penulisan sinkode program/*implemention*)

   Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh programmer yang akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan

secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan testing terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

4. *Integration & testing* (penerapan/pengujian program)

   Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, design dan pengkodean maka sistem yang sudah jadikan digunakan oleh *user*.

5. *Operation & maintenance* (pemeliharaan)

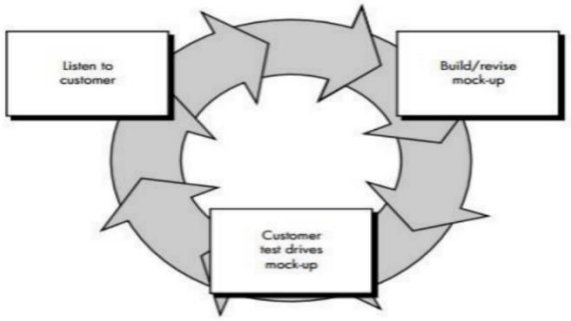
   Perangkat lunak yang susah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (*periperal* atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

2.11.2 *Prototyping*

Sebuah *prototype* adalah versi awal dari sistem perangkat lunak yang digunakan untuk mendemonstrasikan konsep-konsep, percobaan rancangan, dan menemukan lebih banyak masalah dan solusi yang memungkinkan. Sistem *prototype* memperbolehkan pengguna untuk mengetahui bagaimana sistem berjalan dengan baik. Penggunaan metode *prototyping* di dalam penelitian ini bertujuan agar peneliti mendapatkan gambaran aplikasi yang akan dibangun melalui tahap pembangunan aplikasi *prototype* terlebih dahulu yang akan dievaluasi oleh *user*. Aplikasi *prototype* yang telah dievaluasi oleh user selanjutnya akan dijadikan acuan untuk membuat aplikasi yang dijadikan produk akhir sebagai output dari penelitian ini. (Prasetyo et al., 2015)

Metode *prototyping* dimulai dengan mendengarkan kebutuhan dan masukan dari pengguna. Pengembang dan pengguna bertemu dan bersama-sama menentukan tujuan keseluruhan untuk perangkat lunak dan mengidentifikasi apapun persyaratan yang diperlukan. Lalu pengembang membuat sebuah gambaran tentang aplikasi

yang selanjutnya dapat dipresentasikan kepada pelanggan. Gambaran tersebut berfokus pada representasi aspek-aspek aplikasi yang akan terlihat oleh pelanggan/pengguna.



Gambar 2.5 Metode *Prototyping*

Beberapa keunggulan dalam menggunakan metode *prototyping*:

1. Pengembang sistem dan pengguna saling berkomunikasi khususnya dalam hal penyamaan persepsi terhadap pemodelan sistem yang akan menjadi dasar pengembangan sistem operasionalnya,
2. Pelanggan/pengguna ikut terlibat secara aktif dan berpartisipasi dalam menentukan model sistem dan sistem operasionalnya sehingga pelanggan/pengguna akan puas karena sistem yang dibuat sesuai dengan keinginan dan harapannya,
3. Sistem yang dibangun memiliki kualitas yang diinginkan karena sesuai dengan kebutuhan yang ada.

2.11.3 *Rapid Aplication Development* (RAD)

Sebuah proses perkembangan perangkat lunak *sekuensial linier* yang menekankan siklus perkembangan dalam waktu yang singkat. RAD menggunakan metode *iteratif* (berulang) dalam mengembangkan sistem dimana *working model* (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan pengguna dan selanjutnya disingkirkan. Dalam pengembangan sistem informasi normal, memerlukan waktu minimal 180 hari, namun dengan menggunakan metode RAD, sistem dapat diselesaikan dalam waktu 30-90 hari.

Model RAD memiliki 3 tahapan sebagai berikut:

1. *Requirements Planning*

   Secara gars besar dalam fase rencana kebutuhan (*Requirement Planning*):

   User dan *analyst* melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan informasi untuk mencapai tujuan. Tahap rencana kebutuhan juga dilakukan pengumpulan dan analisis alat, bahan, sumber daya, biaya dan data. Pada tahap ini merupakan hal terpenting yaitu adanya keterlibatan dari kedua belah pihak.

2. RAD *Design Workshop*

   Secara garis besar pada tahap ini keaktifan *user* yang terlibat menentukan untuk mencapai tujuan karena pada proses ini melakukan proses ini melakukan proses desain dan melakukan perbaikan apabila masih terdapat ketidaksesuaian desain antara *user* dan analis. Seorang *user* dapat langsung memberikan komentar apabila terdapat ketidaksesuaian pada desain, merancang sistem dengan mengacu pada dokumentasi kebutuhan *user* yang sudah dibuat pada tahap sebelumnya. Keluaran dari tahapan ini adalah spesifikasi *software* yang meliputi organisasi sistem secara umum, struktur data dan yang lain.

3. *Implementation*

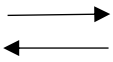
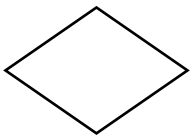
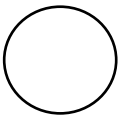
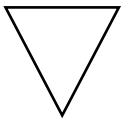
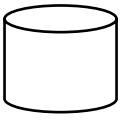
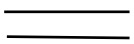
   Tahap *implementation* merupakan tugas dari programmer meneruskan dalam bentuk coding melalui tinjauan pemrograman berdasarkan rancangan sistem yang telah dibuat oleh desainer sistem. Tinjauan pemrograman yang dipakai tergantung dari permintaan *user*.

2.12 *Flowchart*

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah suatu masalah. *Flowchart* biasa disebut sebagai gambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program (Rini, 2009).

Berikut adalah simbol-simbol yang terdapat pada *flowchart*:

Tabel 2.2 Simbol *Flowchart*

Simbol	Nama	Keterangan
	Terminator	Menunjukan awal dan akhir dari suatu proses
	Arah aliran	Menunjukan arah aliran dokumen antar bagian yang terkait pada suatu sistem, baik dari sistem atau keluar sistem
	Input/output	Proses input/output data, parameter, informasi
	Proses	Menunjukan proses yang dilakukan secara komputasi
	Decision	Menunjukan perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	*Connector*	Menunjukan aliran dokumen yang terputus atau terpisah pada *flowchart* yang sama
	*Off-page Connector*	Menunjukan aliran dokumen yang saling antar berhubungan pada *flowchart* yang berbeda
	Pengarsipan	Menunjukan simpanan data non komputer/informasi berupa file pada proses manual. (Dokumen yang disimpan pada lemari arsip, map dll)
	Penyimpanan magnetic	Media penyimpanan yang dilakukan untuk proses terkomputerisasi
	Arsip digital	Menunjukan simpanan data terkomputerisasi

2.13 *Unifed Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka UML lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. (Dharwiyanti, 2003)

Unified Modelling Language adalah sesuatu bahasa pemodelan untuk sistem atau perangkat lunak yang digunakan untuk menyederhanakan permasalahan-permasalahan yang kompleks sehingga lebih mudah dipelajari dan dipahami. Dalam kerangka pengembangan sistem, UML sebagai bahasa pemodelan berguna sebagai sarana analisis, pemahaman, visualisasi, komunikasi, dokumentasi, dan juga bermanfaat dalam pengujian terhadap aplikasi yang telah selesai dikembangkan. UML memiliki beberapa diagram, salah satunya yaitu:

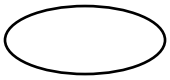
2.13.1 *Use Case Diagram*

Use case diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan perangkat lunak yang akan dibuat. *Use case diagram* juga digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah perangkat lunak dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*, yaitu:

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah orang, tetapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol pada use case diagram:

Tabel 2.3 Simbol *Use Case Diagram*

Simbol	Nama	Keterangan
	*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit saling bertukar antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frame nama *use case*
	Aktor	Proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu merupakan orang
	Asosiasi	Komunikasi antara aktor dengan use case atau use case memiliki interaksi dengan aktor
	*Extends*	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi
	Generalisasi	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya. Arah panah yang mengarah pada use case menjadi generalisasinya (umum)
	*Include*	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya

2.14 Metode Pengujian Sistem

Metode pengujian adalah metode atau teknik untuk menguji perangkat lunak. Metode pengujian mengacu pada perancangan data uji yang akan dieksekusi pada perangkat lunak yang dikembangkan. Suatu metode pengujian diharapkan memiliki mekanisme menemukan data uji yang dapat menguji perangkat lunak secara lengkap dan mencapai probabilitas tinggi untuk menemukan kesalahan. Perangkat lunak dapat diuji dengan dua cara yaitu:

2.14.1 Pengujian *Black Box*

Pengujian *black box* digunakan untuk merepresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam kotak hitam, item-item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar. Pada *black box testing*, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Pada *black box testing*, dicobakan beragam masukan dan memeriksa keluaran yang dihasilkan. Teknik *black box testing* juga dapat digunakan untuk pengujian berbasis skenario di mana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan usecase dan informasi analisis yang lain (Wibisono & Baskoro, 2002). Jika keluaran program dan seluruh fasilitas program berjalan dan tidak terjadi kesalahan pada saat dioperasikan, data keluaran telah sesuai dengan yang diharapkan, maka program dianggap baik. (Sriwahyuni, 2011)

2.14.2 Pengujian *White Box*

Pengujian *white box* adalah teknik pengujian yang melibatkan struktur logis dan implementasi suatu sistem. Pengujian *white box* mengharuskan penguji memiliki pengetahuan yang cukup tentang struktur logika dan implementasi sistem yang diuji. Pengujian ini dilakukan dengan menganalisis dan memeriksa kode program secara cermat untuk menemukan kesalahan pada sistem. Keuntungan dari pengujian *white box* adalah dapat dilakukan dengan lebih detail dan menyeluruh sehingga meminimalisir kesalahan. Selain itu, pengujian ini juga bisa dilakukan pada tahap awal tanpa harus menunggu tampilan pengguna (*User Interface*) siap. Namun, pengujian *white box* juga mempunyai kelemahan yaitu pengujian yang dilakukan sangat kompleks sehingga memerlukan lebih banyak waktu dan sumber daya. (Mohd. Ehmer & Farmeena, 2012)