

BAB II

LANDASAN TEORI

2.1 PENGERTIAN PREDIKSI

Prediksi adalah suatu proses memperkirakan secara sistematis tentang sesuatu yang paling mungkin terjadi di masa depan berdasarkan informasi masa lalu dan sekarang yang dimiliki, agar kesalahannya (selisih antara sesuatu yang terjadi dengan hasil perkiraan) dapat diperkecil. Prediksi tidak harus memberikan jawaban secara pasti kejadian yang akan terjadi, melainkan berusaha untuk mencari jawaban sedekat mungkin yang akan terjadi.

Berdasarkan teknik yang digunakan untuk memprediksi maka prediksi dapat dibagi menjadi dua bagian yaitu prediksi kualitatif dan prediksi kuantitatif.

1. Prediksi Kualitatif

Prediksi kualitatif didasarkan atas data kualitatif pada masa lalu. Metode kualitatif digunakan jika data masa lalu dari variabel yang akan diprediksi tidak ada, tidak cukup atau kurang dipercaya. Hasil prediksi yang dibuat sangat tergantung pada individu yang menyusunnya. Hal ini penting karena hasil prediksi tersebut ditentukan berdasarkan pemikiran yang bersifat *judgement* atau opini, pengetahuan dan pengalaman dari penyusunnya. Oleh karena itu metode kualitatif ini disebut juga *judgemental, subjective, intuitive*.

2. Prediksi Kuantitatif

Prediksi kuantitatif didasarkan atas data kuantitatif pada masa lalu. Hasil prediksi yang dibuat sangat tergantung pada metode yang dipergunakan dalam prediksi tersebut. Dengan metode yang berbeda akan diperoleh hasil prediksi yang berbeda. Hal yang perlu diperhatikan dari penggunaan metode tersebut adalah baik tidaknya metode yang digunakan dan sangat ditentukan dari penyimpangan antara hasil prediksi dengan kenyataan yang terjadi. Metode yang baik adalah metode yang memberikan nilai-nilai perbedaan atau penyimpangan yang

mungkin. Prediksi kuantitatif hanya dapat digunakan apabila terdapat tiga kondisi sebagai berikut :

- a. Adanya informasi tentang keadaan yang lain.
- b. Informasi tersebut dapat dikuantifikasikan dalam bentuk data.
- c. Dapat diasumsikan bahwa pola yang lalu akan berkelanjutan pada masa yang akan datang

(Herdianto, 2013)

2.2 KLASIFIKASI

2.2.1 KONSEP KLASIFIKASI

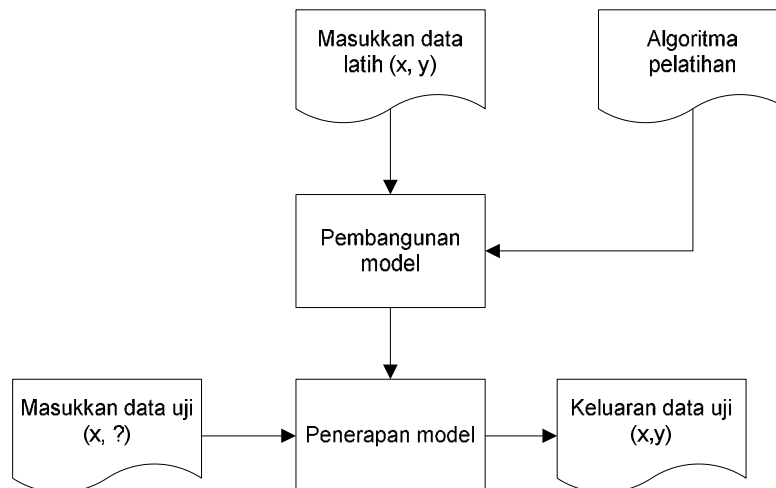
Klasifikasi merupakan suatu pekerjaan menilai objek data untuk memasukkannya ke dalam kelas tertentu dari sejumlah kelas yang tersedia. Dalam klasifikasi ada dua pekerjaan utama yang dilakukan yaitu : pertama, pembangunan model sebagai *prototype* untuk disimpan sebagai memori dan kedua, penggunaan model tersebut untuk melakukan pengenalan/ klasifikasi/ prediksi pada suatu objek data lain agar diketahui di kelas mana objek data tersebut dalam model yang mudah disimpan.

Contoh aplikasi yang sering ditemui adalah pengklasifikasian jenis hewan, yang mempunyai sejumlah atribut. Dengan atribut tersebut, jika ada hewan baru, kelas hewannya bisa langsung diketahui. Contoh lain adalah bagaimana melakukan diagnosis penyakit kulit kanker melanoma (Amaliyah et al, 2011), yaitu dengan melakukan pembangunan model berdasarkan data latih yang ada, kemudian menggunakan model tersebut untuk mengidentifikasi penyakit pasien baru sehingga diketahui pasien tersebut menderita kanker atau tidak.

(Prasetyo, 2012)

2.2.2 MODEL KLASIFIKASI

Model dalam klasifikasi mempunyai arti yang sama dengan kotak hitam, dimana ada suatu model yang menerima masukan, kemudian mampu melakukan pemikiran terhadap masukan tersebut dan memberikan jawaban sebagai keluaran dari hasil pemikirannya.



Gambar 2. 1 Proses Klasifikasi (Prasetyo, 2012)

Pada gambar di atas disediakan sejumlah data latih (x, y) untuk digunakan sebagai data pembangunan model. Model tersebut kemudian dipakai untuk memprediksi kelas dari data uji (x, y) sehingga diketahui kelas y yang sesungguhnya.

Model yang sudah dibangun pada saat pelatihan kemudian dapat digunakan untuk memprediksi label kelas baru yang belum diketahui. Dalam pembangunan model selama proses pelatihan tersebut diperlukan suatu algoritma untuk membangunnya, yang disebut algoritma pelatihan. Ada banyak algoritma pelatihan yang sudah dikembangkan oleh para peneliti seperti K-Nearest Neighbor, Artificial Neural Network, Support Vector Machine dan sebagainya. Setiap algoritma mempunyai kelebihan dan kekurangan, tetapi semua algoritma berprinsip sama, yaitu melakukan suatu pelatihan sehingga di akhir pelatihan, model dapat memetakan (memprediksi) setiap vektor masukan ke label kelas keluaran dengan benar. (Prasetyo, 2012)

2.3 KLASIFIKASI NAIVE BAYES

Naive Bayes adalah sebuah *classifier* probabilistik berdasarkan *Bayes Rule of conditional probability*. Naive Bayes menggunakan probabilitas untuk mengklasifikasikan *instance* baru. Cara kerja Naive Bayes sendiri adalah dengan mencari angka peluang terbesar dari kemungkinan klasifikasi, dengan melihat

frekuensi tiap klasifikasi pada data training. Klasifikasi Bayesian didasarkan pada teorema Bayes, yang diambil dari nama ahli matematika sekaligus menteri Prebysterian Inggris yang bernama Thomas Bayes (1702-1761). Terdapat dua peluang dalam teorema ini, yaitu Posterior dan Prior. $P(H|X)$ merupakan probabilitas Posterior dari H dikondisikan dalam X. Sedangkan, $P(H)$ merupakan probabilitas Prior dari H. Probabilitas Posterior merupakan probabilitas yang didasarkan pada informasi-informasi yang ada, sedangkan probabilitas Prior merupakan probabilitas yang independen. Secara sama, $P(X|H)$ adalah probabilitas Posterior dari X dikondisikan dalam H. Sedangkan $P(X)$ adalah probabilitas Prior dari X. Bentuk umum dari teorema bayes adalah sebagai berikut.

Misalnya, $P(A) = 0,99$ artinya probabilitas bahwa kejadian A akan terjadi 99%, probabilitas A tidak akan terjadi $(100-99)\% = 1\%$. Didalam praktiknya nilai probailitas bisa berdasarkan pertimbangan tenaga ahli dalam bidangnya. Kejadian (event) disebutkan suatu sub-set dan sub-set terdiri dari elemen-elemen, maka apabila A suatu kejadian.

$$P(A) = \frac{\text{banyak elemen subset yang membentuk } A}{\text{seluruh elemen dalam set}}$$

Didalam probabilitas terdapat probabilitas prior dan probabilitas posterior. Probabilitas prior adalah informasi awal probabilitas dengan sudah adanya pengalaman sebelumnya namun belum ada hasil nyata, Probabilitas posterior adalah probabilitas setelah memperhatikan hasil penelitian sampel.

Ide dasar dari naive bayes adalah menangani masalah yang bersifat hipotesis yakni mendesain suatu klasifikasi untuk memisahkan objek. Misalkan terdapat dua jenis objek dengan kemungkinan kemunculan acak, selanjutnya ingin diprediksi objek apa yang akan terjadi selanjutnya. Persamaan dari teorema bayes adalah :

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

Keterangan :

X = kemunculan karakteristik secara keseluruhan

H = kemunculan karakteristik pada kelas

$P(H)$ = probabilitas hipotesis H (prior probabilitas)

$P(X)$ = probabilitas dari X

$P(H|X)$ = probabilitas X berdasarkan kondisi H (posterior probabilitas)

$P(X|H)$ = probabilitas H berdasarkan kondisi X

Naive bayes merupakan penyederhanaan dari teorema Bayes. Bentuk dari Naive bayes adalah sebagai berikut :

$$P(H|X) = P(X|H) \times P(X)$$

Klasifikasi naive bayes bekerja berdasarkan teori probabilitas yang memandang semua fitur dari data sebagai bukti dalam probabilitas. Hal ini memberikan karakteristik naive bayes sebagai berikut :

1. Metode Naive Bayes bekerja teguh (*robust*) terhadap data-data yang terisolasi yang biasanya merupakan data dengan karakteristik berbeda (*outliner*). Naive bayes juga bisa menangani nilai atribut yang salah dengan mengabaikan data latih selama proses pembangunan model dan prediksi.
 2. Tangguh menghadapi atribut yang tidak relevan.
 3. Atribut yang mempunyai korelasi bisa mendegradasi kinerja klasifikasi naive bayes karena asumsi independensi atribut tersebut sudah tidak ada.
- (Septari, 2014)

2.4 ASUMSI KLASIFIKASI NAIVE BAYES

Ketepatan waktu keberangkatan penerbangan yang akan datang salah satunya dipengaruhi oleh informasi di masa lalu. Pada bulan Januari merupakan periode *peak-season* dimana banyak orang bepergian menggunakan angkutan udara sehingga besar kemungkinan adanya penundaan pada keberangkatan penerbangan. Naive bayes akan mengklasifikasikan penerbangan yang akan datang termasuk tepat waktu atau tidak tepat waktu berdasarkan data histori penerbangan.

2.5 PENGERTIAN ON TIME PERFORMANCE

On time performance adalah ukuran kemampuan layanan transportasi untuk dapat tepat waktu. Hampir semua sistem transportasi memiliki jadwal, yang menggambarkan kapan kendaraan harus sampai pada jadwal pemberhentian. *On time performance* pada penerbangan dimonitor secara ketat. Sejumlah situs *website* memiliki laporan ketepatan waktu maskapai penerbangan yang sering dioperasikan oleh departemen pemerintah. OTP merupakan indikator ketepatan waktu penerbangan yang dimiliki oleh setiap maskapai penerbangan. Berdasarkan penelitian yang dilakukan oleh Booz Allen Hamilton pada tahun 2016 yang berjudul “Punctuality: How Airlines can Improve On Time Performance” bahwa pada umumnya industri penerbangan menerapkan aturan 15 menit toleransi waktu keberangkatan pesawat sebagai tolak ukur ketepatan waktu penerbangan, dan menurut Australian Business Traveler, maskapai penerbangan dikatakan memiliki performa yang baik jika OTP mencapai nilai 90% (Wikipedia, On-time performance, 2016).

Beberapa definisi yang menyangkut *on time performance* :

1. Estimated Time Departure (ETD)
Merupakan waktu perkiraan keberangkatan pesawat dari suatu bandara.
2. Actual Time Departure (ATD)
Merupakan waktu sebenarnya dari keberangkatan pesawat dari suatu bandara.
3. Estimated Time Arrival (ETA)
Merupakan waktu perkiraan kedatangan pesawat di suatu bandara.
4. Actual Time Arrival (ATA)
Merupakan waktu sebenarnya dari kedatangan pesawat di suatu bandara.
5. Flight Time
Merupakan waktu total pesawat mulai dari mesin dinyalakan dari bandara keberangkatan sampai tiba di bandara kedatangan.
6. Total Departure

Merupakan jumlah keberangkatan pesawat yang direncanakan oleh maskapai penerbangan.

Adapun rumus untuk menentukan *on time performance* adalah :

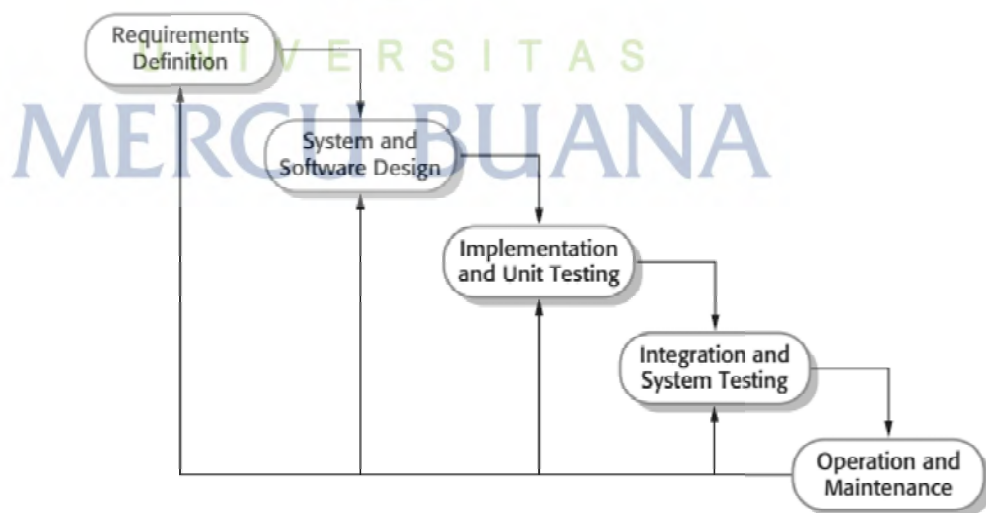
$$\text{On Time Performance} = \frac{\text{Services on time}}{\text{Total services}} \times 100\%$$

(Wikipedia, On-time performance, 2016)

2.6 TEORI PENGEMBANGAN APLIKASI

2.6.1 MODEL WATERFALL

Model pengembangan perangkat lunak yang pertama kali diperkenalkan oleh Royce pada tahun 1970 ini berasal dari hasil adaptasi pengembangan perangkat keras, karena pada waktu itu belum terdapat metodologi pengembangan perangkat lunak yang lain. Adanya aliran dari satu tahap ke tahap lainnya, model ini disebut sebagai model air terjun atau *waterfall* model. Model *waterfall* adalah pengembangan berbasis rencana dimana pada prinsipnya semua aktivitas harus terencana dan terjadwal sebelum memulai suatu pekerjaan.



Gambar 2. 2 Diagram Model Waterfall (Sommerville, 2011)

Tahapan-tahapan model *waterfall* adalah sebagai berikut :

1. Requirement analysis and definition

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2. System and software design

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain sistem disiapkan. Desain sistem membantu dalam menentukan perangkat keras dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

3. Implementation and unit testing

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut unit, yang terintegrasi dalam tahap selanjutnya. Setiap unit dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai unit testing.

4. Integration and system testing

Seluruh unit yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing unit. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

5. Operation and maintenance

Tahap akhir dalam model *waterfall*. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

Model air terjun sangat cocok digunakan jika kebutuhan pelanggan sudah sangat dipahami dan kemungkinan terjadinya perubahan kebutuhan selama pengembangan perangkat lunak kecil. Hal positif dari model air

terjun adalah struktur tahap pengembangan sistem yang jelas, dokumentasi dihasilkan di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan (tidak ada tumpang tindih pelaksanaan tahap).

(Sommerville, 2011)

Kenyataan yang terjadi sangat jarang model air terjun dapat dilakukan sesuai alurnya karena sebab berikut :

1. Perubahan spesifikasi perangkat lunak terjadi di tengah alur pengembangan.
2. Sangat sulit bagi pelanggan untuk mendefinisikan semua spesifikasi di awal alur pengembangan.
3. Pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan di akhir alur pengembangan.

Dengan berbagai kelemahan yang dimiliki model air terjun tapi model ini telah menjadi dasar dari model-model yang lain dalam melakukan perbaikan model pengembangan perangkat lunak.

(A.S & Shalahuddin, 2015)

2.6.2 UNIFIED MODELING LANGUAGE (UML)

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. *Unified Modeling Language* atau UML muncul karena adanya kebutuhan pemodelan visual untuk mespesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Berikut adalah jenis-jenis diagram UML :

1. Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use*

case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan apa saja yang berhak menggunakan fungsi-fungsi itu.

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel 2. 1 Simbol-Simbol Use Case (A.S & Shalahuddin, 2015)

Simbol	Deskripsi
	<p>Use case</p> <p>Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i></p>
	<p>Aktor / <i>actor</i></p> <p>Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri</p>
	<p>Asosiasi / <i>association</i></p> <p>Komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor</p>
	<p>Ekstensi / <i>extend</i></p> <p>Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau <i>tanpa use case</i> tambahan itu</p>

Simbol	Deskripsi
	Generalisasi / <i>generalization</i> Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
	Menggunakan / <i>include</i> / <i>uses</i> Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini.

Berikut adalah contoh dari use case diagram :



Gambar 2. 3 Contoh Use Case Diagram (Permana, 2012)




2. Activity Diagram

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas

menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2. 2 Simbol-Simbol Activity Diagram (A.S & Shalahuddin, 2015)

Simbol	Deskripsi
	Status awal Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas Aktivitas yang dilakukan sistem, aktivitasnya biasanya diawali dengan kata kerja
	Percabangan / <i>decision</i> Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
	Penggabungan / <i>join</i> Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
	Status akhir Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	<i>Swimlane</i> Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Berikut adalah contoh activity diagram :



Gambar 2. 4 Contoh Activity Diagram (Permana, 2012)

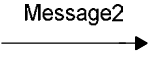
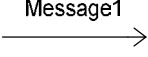


3. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case.

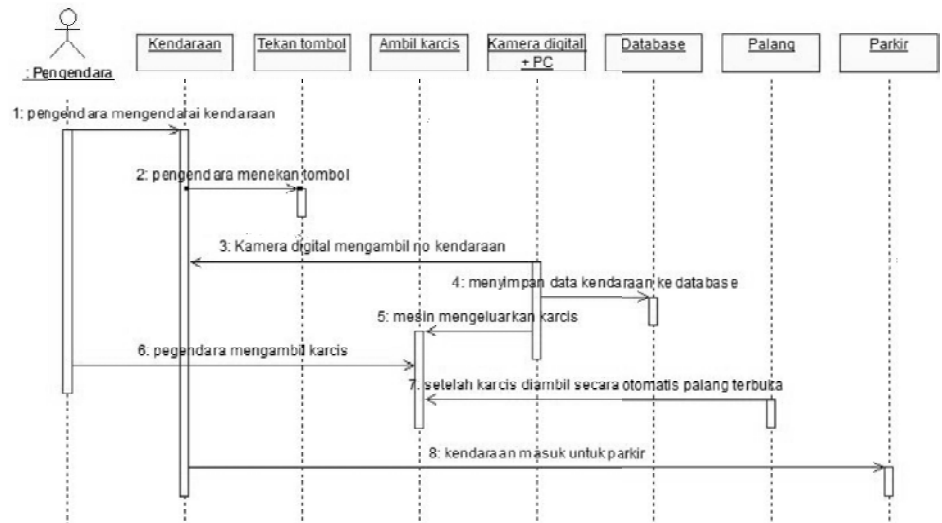
Berikut adalah simbol-simbol yang ada pada diagram sekuen :

Tabel 2. 3 Simbol-Simbol Sequence Diagram (A.S & Shalahuddin, 2015)

Simbol	Deskripsi
	<i>Object lifeline</i> Menyatakan objek yang berinteraksi peserta kehidupan dari suatu objek
	Waktu aktif Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya
	Pesan tipe <i>create</i> Menyatakan suatu objek membuat objek yang lain, arah panah

	mengarah pada objek yang dibuat
	Pesan tipe <i>call</i> Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
	Pesan tipe <i>send</i> Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
	Pesan tipe keluaran Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
	Pesan tipe <i>destroy</i> Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada <i>create</i> maka ada <i>destroy</i>

Berikut adalah contoh dari sequence diagram :



Gambar 2. 5 Contoh Sequence Diagram (Permana, 2012)

4. Class Diagram

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi

- Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
- Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

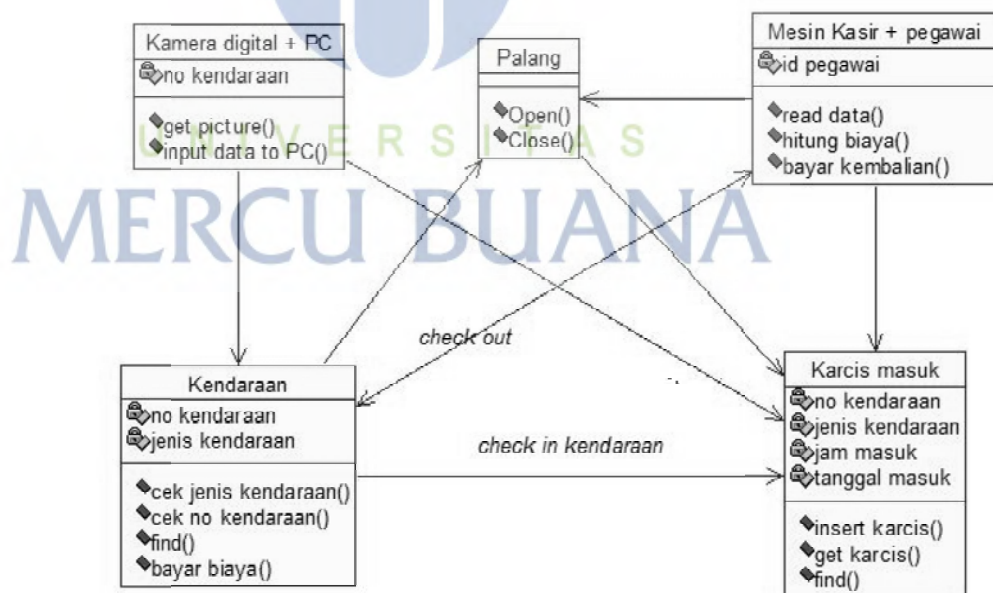
Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel 2. 4 Simbol-Simbol Class Diagram (A.S & Shalahuddin, 2015)

Simbol	Deskripsi
	Kelas Kelas pada struktur sistem
	Antarmuka / <i>interface</i> Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek
	Asosiasi / <i>association</i> Relasi antarkelas dengan makna umum, asosiasi biasanya juga

Simbol	Deskripsi
	disertai dengan <i>multiplicity</i>
	Asosiasi berarah / <i>directed association</i> Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
	Generalisasi Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
	Agregasi / <i>aggregation</i> Relasi antarkelas dengan makna semua-bagian (<i>whole-part</i>)

Berikut adalah contoh dari class diagram :



Gambar 2. 6 Contoh Class Diagram (Permana, 2012)

(A.S & Shalahuddin, 2015)

2.6.3 CONCEPTUAL DATA MODEL (CDM)

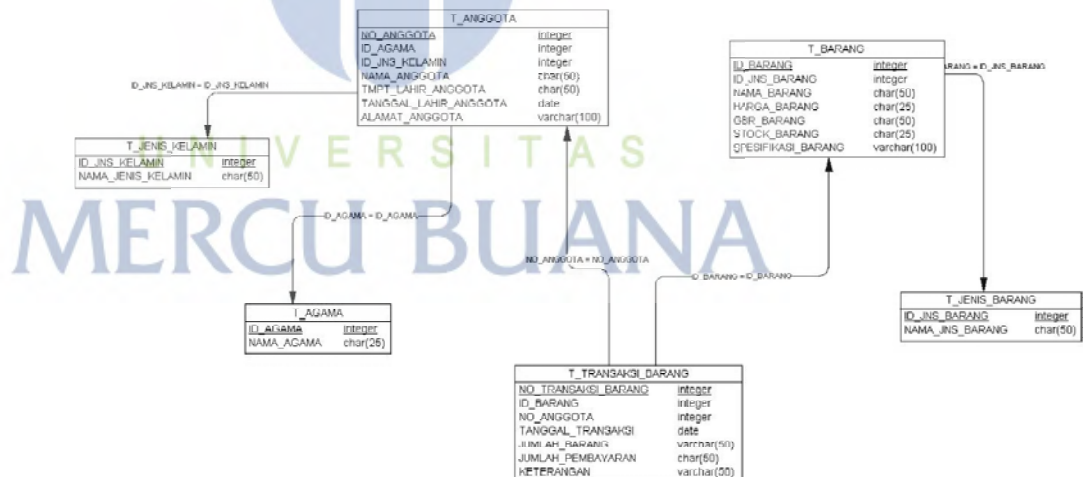
Conceptual Data Model (CDM) atau model konsep data merupakan konsep yang berkaitan dengan pandangan pemakai terhadap data yang disimpan dalam basis data. CDM dibuat sudah dalam bentuk tabel-tabel tanpa tipe data yang menggambarkan relasi antar tabel untuk keperluan implementasi ke basis data.

Berikut adalah simbol-simbol yang ada pada CDM :

Tabel 2. 5 Simbol-Simbol Conceptual Data Model (A.S & Shalahuddin, 2015)

Simbol	Deskripsi
	Entitas / tabel Entitas atau tabel yang menyimpan data dalam basis data
1...*	Relasi Relasi antar tabel yang terdiri atas nama relasi dan multiplicity

Berikut adalah contoh dari conceptual data model :



Gambar 2. 7 Contoh Conceptual Data Model (Permana, 2012)

(A.S & Shalahuddin, 2015)

2.7 TEORI ALAT PENGEMBANGAN APLIKASI

2.7.1 BASIS DATA

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Database Management System* (DBMS).

Kebutuhan basis data dalam sistem informasi meliputi :

1. Memasukkan, menyimpan, dan mengambil data.
2. Membuat laporan berdasarkan data yang telah disimpan.

Tujuan dari dibuatnya tabel-tabel disini adalah untuk menyimpan data ke dalam tabel-tabel agar mudah diakses. Oleh karena itu, untuk merancang tabel-tabel yang akan dibuat maka dibutuhkan pola pikir penyimpanan data nantinya jika dalam bentuk baris-baris data dimana setiap baris terdiri dari beberapa kolom.

(A.S & Shalahuddin, 2015)

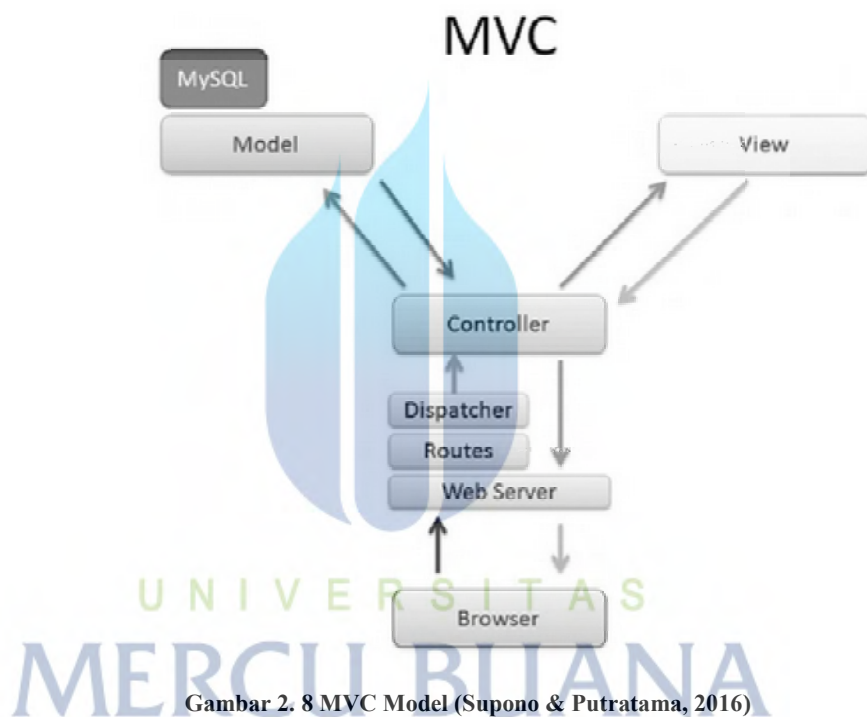
2.7.2 CODEIGNITER PHP FRAMEWORK

Codeigniter adalah aplikasi *open source* berupa *framework* dengan model MVC (*Model, View, Controller*) untuk membangun *website* dinamis dengan menggunakan PHP. Codeigniter memudahkan developer atau pengembang *web* untuk membuat aplikasi *web* dengan cepat dan mudah dibandingkan dengan membuat dari awal.

MVC merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *web*. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi :

1. *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web* bagian ini biasanya berupa *file template* HTML, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian model.

2. Model, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*select, insert, update, delete*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
3. *Controller*, merupakan bagian yang mengatur hubungan antara bagian model dengan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.



Gambar 2. 8 MVC Model (Supono & Putratama, 2016)

(Supono & Putratama, 2016)

2.7.3 MYSQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (DBMS) yang *multithread*, dan *multi-user*. MySQL adalah implementasi dari sistem manajemen basis data relasional (RDBMS). MySQL dibuat oleh TcX dan telah dipercaya mengelola sistem dengan 40 buah *database* berisi 10.000 tabel dan 500 di antaranya memiliki 7 juta baris. Pada saat ini MySQL merupakan *database server* yang sangat terkenal di dunia, semua itu tak lain karena bahasa dasar yang digunakan

untuk mengakses *database* yaitu SQL. SQL (*Structured Query Language*) pertama kali diterapkan pada sebuah proyek riset pada laboratorium riset San Jose, IBM yang bernama sistem R. Kemudian SQL juga dikembangkan oleh Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses pengaksesan *database* lebih *user-friendly* dibandingkan dengan yang lain, misalnya dBase atau Clipper karena mereka masih menggunakan perintah-perintah pemrograman murni.

(Supono & Putratama, 2016)

2.7.4 WAMP

WAMP Server adalah sebuah paket server yang berada di localhost dan hanya diinstall di dalam sistem operasi windows. WAMP sendiri adalah singkatan dari Windows and the principal components of the package: Apache, MySQL and PHP.

(Fajar, 2016)

2.7.5 NOTEPAD++

Notepad++ adalah suatu *text editor* yang berjalan pada sistem operasi windows. Notepad++ menggunakan komponen-komponen Scintilla agar dapat menampilkan dan menyunting *text* dan berkas *source code* berbagai bahasa pemrograman. Notepad++ didistribusikan sebagai *free software* (gratis). Proyek ini dilayani oleh Sourceforge.net dengan telah diunduh lebih dari 27 juta kali dan dua kali memenangkan penghargaan *SourceForge Community Choice Award for Best Developer Tool*.

(Dwi, 2015)

2.8 TEORI PENGUJIAN

2.8.1 CONFUSION MATRIX

Confusion matrix adalah tabel untuk mengukur kinerja algoritma klasifikasi atau *classifier*. Pada *confusion matrix* terdapat beberapa istilah yang umum digunakan pada kasus klasifikasi *binary class*, yaitu :

1. True Positives (TP) : kasus dimana seseorang diprediksi sebagai penderita kanker dan kenyataannya orang itu adalah penderita kanker.
2. True Negatives (TN) : kasus dimana seseorang diprediksi tidak menderita kanker dan pada kenyataannya orang itu tidak menderita kanker.
3. FP (False Positives) : kasus dimana seseorang diprediksi sebagai penderita kanker tetapi ternyata orang itu tidak menderita kanker.
4. FN (False Negative) : kasus dimana seseorang diprediksi tidak menderita kanker tetapi ternyata orang itu menderita kanker.

Berikut adalah contoh confusion matrix untuk kasus di atas :

Tabel 2. 6 Confusion Matrix

N = 165	Prediksi TIDAK KANKER	Prediksi KANKER
Aktual TIDAK KANKER	TN = 50	FP = 10
Aktual KANKER	FN = 5	TP = 100

Dari nilai TP, TN, FP, dan FN maka dapat dihitung beberapa nilai lain yang dapat dijadikan nilai kinerja classifier. Nilai-nilai tersebut adalah :

Tabel 2. 7 Perhitungan Confusion Matrix

Nama	Rumus	Perhitungan
Accuracy Persentase classifier benar melakukan prediksi	$(TP + TN) / N$	$(100 + 50) / 165 = 0,91$
Error Rate Persentase classifier melakukan kesalahan prediksi	$(FP + FN) / N$	$(10 + 5) / 165 = 0,09$
TP Rate Persentase data positif yang diprediksi sebagai positif	$TP / (TP + FN)$	$100 / (100 + 5) = 0,95$

Nama	Rumus	Perhitungan
FP Rate Persentase data negatif diprediksi sebagai positif	$FP / (TN + FP)$	$10 / (10 + 5) = 0,17$
Specificity Persentase data negatif diprediksi sebagai negatif	$TN / (TN + FP)$	$50 / (50 + 10) = 0,83$
Precision Persentase prediksi data sebagai positif yang benar	$TP / (FP + TP)$	$100 / (10 + 100) = 0,91$
Prevalence Persentase jumlah instance positif pada data	Actual Positive / N	$105 / 165 = 0,64$

(Markham, 2014)

2.8.2 WHITE-BOX TESTING

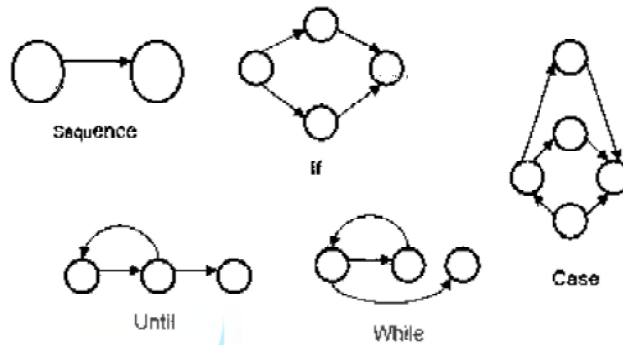
Pengujian *white-box*, yang kadang-kadang disebut pengujian *glass-box*, adalah metode desain *test case* yang menggunakan struktur kontrol desain prosedural untuk memperoleh *test case*. Dengan menggunakan metode pengujian *white-box*, perekayasa sistem dapat melakukan *test case*: (1) memberikan jaminan bahwa semua jalur independen pada suatu modul telah digunakan paling tidak satu kali; (2) menggunakan semua keputusan logis pada sisi *true* dan *false*; (3) mengeksekusi semua *loop* pada batasan mereka dan pada batas operasional mereka; (4) menggunakan struktur data internal untuk menjamin validitasnya.

Pengujian basis path adalah teknik pengujian *white-box* yang diusulkan pertama kali oleh Tom McCabe. Metode basis *path* ini memungkinkan desainer *test case* mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan basis set dari jalur eksekusi. Test case yang dilakukan untuk menggunakan basis set tersebut dijamin untuk menggunakan setiap statemen di dalam program paling tidak sekali selama pengujian.

(Pressman, 2012)

2.8.2.1 NOTASI DIAGRAM ALIR

Grafik alir menggambarkan aliran kontrol logika yang menggunakan notasi yang sudah ditentukan. Masing-masing gagasan terstruktur memiliki simbol grafik alir yang sesuai.



Gambar 2. 9 Notasi Diagram Alir (Pressman, 2012)

2.8.2.2 KOMPLEKSITAS SIKLOMATIS

Kompleksitas siklomatis adalah matriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila matriks ini digunakan dalam konteks metode pengujian basis *path*, maka nilai yang terhitung untuk kompleksitas siklomatis menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua statemen telah dieksekusi sedikitnya satu kali.

Kompleksitas siklomatis dapat dihitung menggunakan cara :

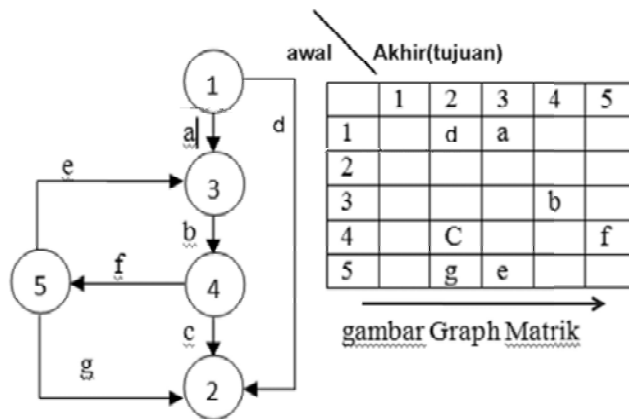
$$V(G) = E - N + 2$$

Dimana E adalah jumlah edge grafik alur dan N adalah jumlah simpul grafik alir.

2.8.2.3 MATRIKS GRAFIK

Matriks grafis adalah matriks bujur sangkar yang ukurannya sama dengan jumlah simpul pada grafik alir. Masing-masing baris

dan kolom sesuai dengan simpul yang diidentifikasi, dan entri matriks sesuai dengan *edge* di antara simpul.



Gambar 2. 10 Pressman (2012)

Kompleksitas siklomatis dengan menggunakan matriks koneksi dapat dihitung menggunakan cara :

$$V(G) = P + 1$$

Dimana P adalah total dari *node* yang terhubung.

(Pressman, 2012)

2.9 PENELITIAN TERDAHULU

1. Jurnal tahun 2014 “DATA MINING MENGGUNAKAN ALGORITMA NAIVE BAYES UNTUK KLASIFIKASI KELULUSAN MAHASISWA UNIVERSITAS DIAN NUSWANTORO” ditulis oleh Yuda Septian Nugroho menyimpulkan bahwa “*Nilai yang dihasilkan oleh algoritma naive bayes memiliki tingkat kekuatan yang cukup tinggi. Hal ini di buktikan dengan hasil perhitungan yang mencapai nilai 82.08%, Nilai 82.08% membuktikan bahwa model yang dibangun dapat digunakan untuk melakukan klasifikasi kelulusan mahasiswa. Nilai 82.08% bisa juga disebabkan oleh kurang kompleksan data yang mengakibatkan model dapat memprediksi dengan akurat.*”
2. Jurnal tahun 2014 “PERANCANGAN APLIKASI PREDIKSI KELULUSAN TEPAT WAKTU BAGI MAHASISWA BARU DENGAN TEKNIK DATA MINING (STUDI KASUS: DATA AKADEMIK MAHASISWA STMIK DIPANEGARA MAKASSAR)”

ditulis oleh Muhammad Syukri Mustafa dan I Wayan Simpen menyimpulkan bahwa *“Dengan menggunakan data mining, khususnya penerapan algoritma K-Nearest Neighbor (KNN), kita dapat mengetahui hubungan kedekatan antara kasus yang baru dengan kasus yang telah ada dalam suatu gudang data (data warehouse) sehingga dapat menjadi acuan untuk memprediksi kelulusan seorang mahasiswa baru apakah dapat menyelesaikan kuliahnya dengan tepat waktu atau tidak berdasarkan kedekatan data yang sudah ada. Dari hasil pengujian dengan menerapkan algoritma KNN dan menggunakan data sampel alumni tahun wisuda 2004-2010 untuk kasus lama dan data alumni tahun wisuda 2011 untuk kasus baru diperoleh tingkat akurasi sebesar 83,36%.”*

3. Jurnal tahun 2013 *“ANALISIS DELAY PENERBANGAN AKIBAT CUACA DI BANDARA AHMAD YANI SEMARANG DENGAN ALGORITMA C4.5”* ditulis oleh Mochamad Nur Sholikhin menyimpulkan bahwa *“Hasil menunjukkan bahwa algoritma C4.5 yang diterapkan pada data set Delay penerbangan di tahun 2013, data menghasilkan nilai akurasi confusion matrix sebesar 94.55% dan akurasi AOC 0.815 dalam selang waktu 0 detik. Dengan adanya penerapan Decision Tree C4.5 diharapkan mampu memberikan solusi bagi pihak bandara maupun maskapai penerbangan dalam membantu menentukan delay penerbangan akibat gangguan cuaca buruk.”*