

## BAB III

### LANDASAN TEORI

#### 3.1 Prediksi

Prediksi merupakan proses memperkirakan yang terjadi dimasa datang berdasarkan data dimasa lalu dan mengurangi ketidakpastian terhadap apa yang akan terjadi dimasa mendatang, dengan meminimumkan kesalahan dalam prediksi menggunakan *mean square error* dan *mean absolute percentage error*. Adapun tahapan-tahapan melakukan prediksi yaitu [9] :

1. Tentukan masalah yang dapat dianalisa serta melakukan pengumpulan data yang akan dibutuhkan dalam proses analisis.
2. Siapkan data untuk diproses dengan benar
3. Tetapkan metode untuk prediksi yang sesuai dengan data yang telah disiapkan
4. Lakukan penerapan dalam metode yang sudah ditetapkan dan lakukan prediksi pada data untuk dimasa yang akan datang.

Melakukan evaluasi hasil prediksi.

Maka dari itu penelitian melakukan prediksi dengan memperkirakan jumlah dari sebuah produk pada periode yang akan datang berdasarkan data masa lalu dan dapat dilakukan analisa menggunakan metode *statistika* atau *least square*. Hasilnya dapat membantu dalam mengambil keputusan untuk mengetahui berapa jumlah produk yang akan dibeli pada berikutnya.

### 3.2 Data Mining

Data Mining merupakan proses ekstraksi atau penggalian data yang besar berguna dari *database* serta digunakan untuk membuat sebuah keputusan yang sangat penting. Proses *data mining* memanfaatkan metode *statistika*, matematika sampai dengan memanfaatkan teknologi *Artificial Intelligence* (AI) [10]. *Data Mining* merupakan metode untuk menemukan informasi yang tersembunyi dalam database dan bagian dari proses *Knowledge Discovery in Databases* (KDD) untuk menemukan informasi dan pola yang berguna di dalam data.

Secara umum, *Data Mining* dibagi menjadi dua kategori yaitu [11] :

1. *Prediktif*, proses untuk menemukan karakteristik dari data dalam satu basis data dengan teknik data miningnya adalah *clustering*, *asosiation*, dan *sequential mining*.
2. *Deskriptif*, proses untuk menemukan pola dari data yang menggunakan beberapa variabel lain di masa depan. Teknik data miningnya adalah *klasifikasi*, *estimasi* dan prediksi. *Data Mining* yang sebagai proses penyaring atau “menambang” pengetahuan dari sejumlah data yang besar.

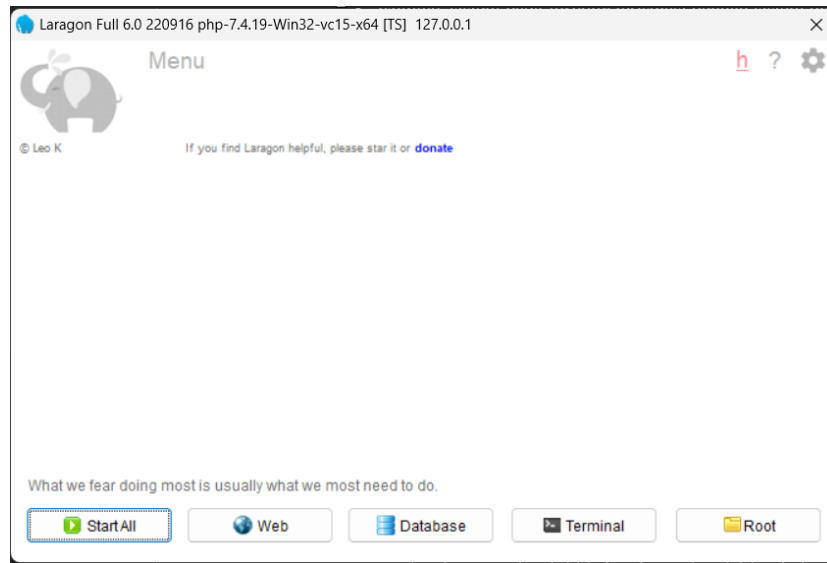
Proses *Knowledge Discovery in Databases* (KDD) dapat digambarkan sebagai berikut :

1. *Data Selection* (seleksi data)
2. *Pre-processing/cleaning* (pembersihan data)
3. *Transformation* (pengubah data)
4. *Data Mining* (Penambangan Data)
5. *Intepretasi/Evaluasi*

### 3.3 Laragon

Laragon merupakan perangkat lunak bersifat *open source* yang mendukung banyak sekali sistem operasi. Tugas *laragon* sebagai *server virtual* atau sering disebut sebagai *localhost* yang menggunakan domain sesuai dengan keinginan atau bisa disebut dengan *pretty url's*, *laragon* juga sebagai pengganti *XAMPP* sangat baik untuk pengelolaan aplikasi yang berbasis *website*. Kelebihan dari *laragon* yaitu [12] :

1. *Pretty URLs*, project diakses dengan menggunakan *app.test* tanpa harus menggunakan *localhost/app*
2. *Portable*, project yang dapat dipindahkan dengan mudahnya tanpa dengan merusak sistem.
3. *Isolated*, sistem pada *laragon* terisolasi dengan sistem operasi maka apa yang pengguna lakukan pada aplikasi tidak mempengaruhi komputer lokal pengguna
4. *Easy Operation*, aplikasi yang otomatis dengan memiliki banyak yang konfigurasi sehingga sangat mudah untuk digunakan
5. *Modern dan Powerful*, aplikasi yang memiliki arsitektur modern sehingga dapat memudahkan digunakan saat membangun web yang modern. Adapun gambar *Laragon* dapat dilihat pada gambar 3.1 berikut :



Gambar 3.1 Laragon

### 3.4 Database MySQL

Database MySQL merupakan *database* yang didukung dengan bahasa pemrograman *script* untuk internet (*PHP* dan *Perl*) atau bisa disebut DBMS (*database management system*) yang *multithread*, *multi-user*. Database bersifat *open source* yang berhubungan dengan *laragon* bertujuan untuk pembuatan kolom serta tabel yang saling berkaitan (memiliki relasi). Mengelola *website* seperti *username*, *password*, tema, *script* dan sebagainya, *PostgreSQL*, *MariaDB*, *OracleDB* dan *MongoDB*. Jadi *MySQL* adalah suatu manajemen sistem yang bagus untuk penghubung antara aplikasi dan *database server*. Adapun cara kerja *database MySQL* sebagai berikut [13].

1. *Client* dapat memberikan sebuah perintah dengan instruksi spesifik dengan bahasa pemrogramannya yaitu *SQL*
2. *Server* akan menjalankan sebuah perintah dan diterima serta menampilkan sebuah informasi di layar *client*.

### 3.5 CodeIgniter

*CodeIgniter* merupakan *framework* PHP untuk membuat sebuah *website* serta dilengkapi yang banyak dan *helper* didalamnya. *CodeIgniter* bersifat *open source* yang mempunyai banyak versi dan khusus untuk *application* juga *web development* berbasis *Model – View - Controller* (MVC), maka sebagai pola desain dan memisahkan kode-kode aplikasi yang dibuat dengan data dinamis yang melibatkan pengkodean lebih singkat [14] :

1. *Model*, adalah bagian hubungan dan pengelolaan langsung ke database.
2. *View*, adalah bagian yang menampilkan tampilan dari sistem.
3. *Controller*, adalah bagian yang menghubungkan antara model dan view pada tiap proses *request database*.

Penggunaan konsep MVC ini akan membuat *website* akan memiliki bagian terpisah. Proses pembuatan *website* dapat dibuat lebih cepat dibandingkan dengan membuat susunannya dari awal. *CodeIgniter* yang saya gunakan dalam pembuatan aplikasi prediksi yaitu *codeigniter* versi 3.

### 3.6 Visual Studio Code(VSCode)

*Visual Studio Code (VSCode)* merupakan perangkat lunak untuk membuat banyak aplikasi dan memudahkan dalam penulisan code yang mendukung beberapa jenis pemrograman, seperti C++, C#, Java, Python, PHP, GO. Visual Studio Code ini bersifat *open source* bagi kalangan programmer sangat terkenal pada *survey Stack Overflow*, dan telah terintegrasi ke Github. Selain itu, fitur lainnya adalah kemampuan untuk menambahkan *plugin*, dimana dapat menambahkan *plugin* untuk menambah fungsionalitas yang tidak disertakan kedalam *Visual Studio Code* [15].

### 3.7 Metode Least Square

Metode Least Square merupakan metode yang berupa data deret berkala atau *time series* yang dibutuhkan data penjualan dimasa lalu akan melakukan prediksi penjualan dimasa mendatang [16]. Pada penelitian ini bahwa X nantinya akan dibuat melalui perhitungan waktu yaitu total pengeluaran. Metode *least square* dibagi menjadi dua kasus yaitu kasus data genap dan kasus data ganjil, garis *linear time series* ini dapat dirumuskan dilihat dari persamaan 1 sebagai berikut:

$$y = a + bx \dots \dots \dots [1]$$

Keterangan:

$y$  : Jumlah penjualan

$a$  dan  $b$  : Koefesien

$x$  : Variabel waktu (hari, bulan, atau tahun)

$N$  : Banyaknya data

Data genap, yaitu skor nilai  $x$  nya: ..., -5, -3, -1, 1, 3, 5, ... Data ganjil, yaitu skor nilai  $x$  nya : ..., -3, -2, -1, 0, 1, 2, 3, ...

Untuk mencari nilai  $a$  dan  $b$  dari persamaan trend dengan caramenggunakan rumus persamaan 2 & 3 sebagai berikut:

$$a = \frac{\sum y}{n} \dots \dots \dots [2]$$

$$b = \frac{\sum xy}{\sum x^2} \dots \dots \dots [3]$$

### 3.8 Postman

Postman digunakan dalam pengujian permintaan API. *Postman* akan mengirim permintaan API ke *server web* dan menerima *respons*, untuk pengujian

tidak ada tambahan atau pengaturan dalam kerangka kerja saat mengirim dan menerima permintaan dari *postman*. *Continuous Integration (CI)* dan *Development Pipeline* menggunakan *postman* untuk pengujian atau *testing*, tidak perlu menulis kode ke jaringan klien HTTP apapun. Dengan gantinya, *test suites* atau disebut dengan *collections* yang membiarkan *postman* berinteraksi dengan API yang memiliki berbagai *tools* jenis permintaan HTTP seperti GET, POST, PUT, PATCH dan mengonversi menjadi kode untuk bahasa seperti *JavaScript* dan *Python*. Untuk pengujian *postman* dan membuat *workspace* baru dengan menggunakan method GET, kemudian menambahkan Alamat *url* dari *device* dan melakukan GET data [19]. Adapun gambar *Postman* pada gambar 3.2 sebagai berikut :



Gambar 3.2 *Postman*





### 3.9 Anaconda Navigator

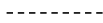
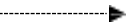
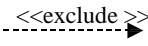
Anaconda Navigator merupakan aplikasi mengelola paket conda, yang tanpa menggunakan perintah baris perintah (CLI) dari bahasa pemrograman *Python* dan *R* yang bersifat gratis dan *open source* untuk kebutuhan *scientific computing (data science)*. Dan merupakan antarmuka pengguna *grafis desktop (GUI)* yang termasuk dalam distribusi *Anaconda* yang pengguna meluncurkan aplikasi dan mengelola

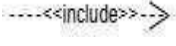




Tabel 3.1 Simbol *Use Case Diagram*

Simbol	Nama	Keterangan
	<p><i>Actor</i></p>	<p><i>Actor</i> merupakan proses atau sistem yang berinteraksi dengan sistem, yang memiliki peran untuk menciptakan <i>Use Case</i>. <i>Actor</i> juga hanya menerima dan memberikan informasi dari sistem kesistem atau keduanya bisa terjadi secara bersamaan.</p>
	<p><i>Use case</i></p>	<p><i>Abstraksi</i> merupakan interaksi antara sistem dan <i>actor</i> yang disediakan sebagai unit-unit dan bertugas untuk tukar pesan antar unit dengan <i>actor</i>. Dan biasanya dikatakan dengan menggunakan kata kerja di awal nama <i>use case</i></p>
	<p>Relasi</p>	<p>Menggambarkan hubungan <i>relasi</i> antara aktor dan <i>use case</i></p>
	<p>Batasan Sistem</p>	<p>Memperlihatkan gambaran dari batasan sistem yang ada pada diagram <i>use case</i></p>


	<p><i>Dependency</i></p>	<p>Hubungan semantik antara dua hal di mana satu (<i>independen</i>) perubahan dapat mempengaruhi lain (<i>independen</i>) lain.</p>
	<p>Generalization</p>	<p>Antara hubungan dua buah <i>use case</i> dimana yang satu berfungsi umum dari yang lainnya, dan bila <i>actor</i> berinteraksi secara pasif dengan sistem. Contohnya adalah jika arah panah mengarah pada <i>use case</i> yang menjadi <i>generalization</i>.</p>
	<p><i>Exclude</i></p>	<p>Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> ditambahkan dapat berdiri sendiri walaupun tanpa <i>use case</i> tambahan. Biasanya juga <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> ditambahkan. Misalnya arah panah yang mengarah pada <i>use case</i> yang ditambahkan, biasanya <i>use case</i> yang telah</p>



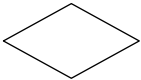
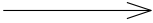

		menjadi <i>exclude</i> merupakan jenis yang sama dengan <i>use case</i> induknya.
	<i>Include</i>	Di dalamnya <i>use case</i> lain ( <i>required</i> ) atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi program.

### 3.10.2 Activity Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem pada proses bisnis yang ada di menu perangkat lunak dan menggambarkan aktivitas dari sistem bukan yang dilakukan aktor. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi, *activity* dapat direalisasikan oleh satu *use case* atau lebih. Dengan fungsi memperlihatkan urutan aktifitas proses pada sistem, membantu memahami proses secara keseluruhan, berdasarkan *use case*, dan menggambarkan proses dan urutan aktivitas sebuah bisnis. Adapun keterangan untuk Simbol *Activity Diagram* dapat dilihat pada tabel 3.3 berikut :

Tabel 3.3 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	<i>Start State</i>	Menunjukkan dimulainya suatu <i>workflow</i>




	<i>End State</i>	Menggambarkan akhir dari pada sebuah <i>activity diagram</i>
	<i>Activities</i>	Menggambarkan sebuah pekerjaan atau tugas dalam <i>workflow</i>
	<i>Decision</i>	Suatu titik pada <i>activity diagram</i> yang mengindikasikan kondisi dimana ada kemungkinan perbedaan transisi.
	<i>State Transition</i>	Menunjukkan suatu aktivitas berikutnya yang setelah aktivitas sebelumnya
	<i>Note</i>	Simbol yang memberikan batasan dan komentar yang terkait dengan suatu <i>element</i> atau kelompok <i>element</i> tertentu.

### 3.10.3 Sequence Diagram

Sequence Diagram secara khusus menjabarkan *use case*, atau bisa disebut dengan diagram yang menggambarkan tentang kolaborasi dinamis antara jumlah *object*. Selain itu *sequence diagram* akan menampilkan pesan atau perintah yang dikirim beserta waktu pelaksanaannya, dengan bertujuan untuk mengetahui urutan kejadian yang menghasilkan output yang diinginkan. Adapun keterangan dari

Simbol *Sequence Diagram* dapat dilihat pada Tabel 3.4 berikut :



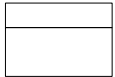



Tabel 3.4 Simbol *Sequence Diagram*

Simbol	Nama	Keterangan
	<i>Life Line</i>	Objek <i>entity</i> , <i>interface</i> yang saling berinteraksi.
	<i>Message</i>	Spesifikasi dari komunikasi antar objek yang berisikan informasi tentang aktivitas yang sedang berlangsung.
	<i>Message</i>	Spesifikasi untuk komunikasi antar objek yang berisi informasi tentang aktivitas yang sedang berlangsung

### 3.10.4 Class Diagram

Class Diagram adalah proses dari pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini. *Forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya, merubah kode program menjadi model. *Class Diagram* memiliki fitur yaitu atribut dan operasi. Atribut (*attribute*) dan operasi (*operation*) menggambarkan perilaku suatu kelas serta perluasannya yaitu seperti *stereotypes*, *tagged values*, dan batasan (*constraints*) merupakan fitur-fitur sebuah kelas. Adapun keterangan dari Simbol *Class Diagram* dapat dilihat pada Tabel 3.5 berikut :

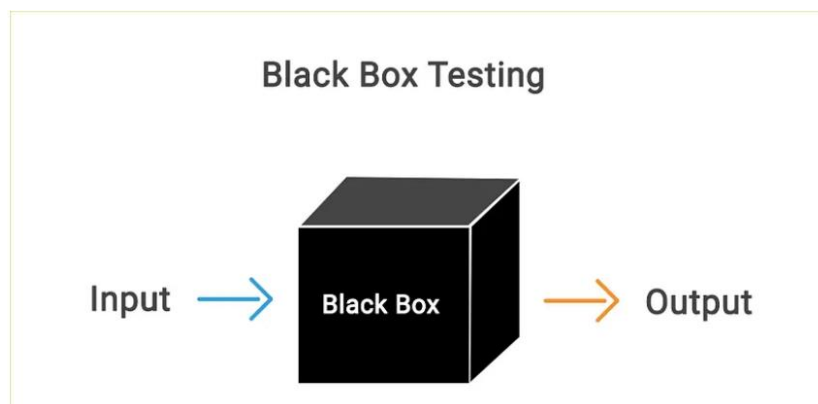
Tabel 3.5 Simbol *Class Diagram*

Simbol	Nama	Keterangan
	<i>Generalization</i>	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
	<i>Nary Association</i>	Upaya untuk menghindari <i>asosiasi</i> dengan lebih dari 2 objek.
	<i>Class</i>	Himpunan dari obek-objek yang berbagi atribut serta operasi yang sama.
	<i>Collaboration</i>	<i>Deskripsi</i> dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
	<i>Dependency</i>	Suatu hubungan <i>semantic</i> antara dua <i>things</i> dimana perubahan pada suatu <i>things (independent)</i> mungkin mempengaruhi <i>semantic</i>

		<i>things (independent) lain.</i>
—————	<i>Association</i>	Penghubung objek satu dengan objek lainnya

### 3.10.5 Pengujian Sistem

Pengujian sistem ini adalah pengujian yang dilakukan dengan mengamati hasil *input* dan *output* dari perangkat lunak tanpa mengetahui *source code* dari sistem tersebut. Pengujian yang dilakukan diakhir pembuatan sistem untuk mengetahui apakah sistem berfungsi dengan baik atau tidak. Pada gambar 3.4 merupakan hasil *pengujian* aplikasi prediksi jumlah penjualan produk menggunakan metode *Black Box*.



Gambar 3.4 *Black Box Testing*