BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Penelitian terdahulu merupakan teori dari berbagai penelitian sebelumnya yang dijadikan acuan penelitian dan data pendukung penelitian. Adapun beberapa penelitian yang memiliki topik serupa tentang Sistem Cerdas Prediksi Jumlah Kelas Program Studi Teknik Informatika Menggunakan Algoritma Regresi Linear Berganda adalah sebagai berikut :

1. (Syahputra, Syahril, & Sobirin, 2020) dengan judul "Prediksi Jumlah Murid Baru Dengan Menggunakan Metode Regresi Linear Berganda". Dalam penelitian ini dirancang dan dibangun sebuah aplikasi berbasis Website untuk meningkatkan kinerja pada SMA Plus Taruna Akterlis Medan. Adapun variable yang digunakan untuk menentukan prediksi jumlah murid baru pada penelitian ini yakni Jumlah Yang Mengikuti Ujian Masuk, dan Promosi.
2. (Afkarina, Widodo, & Furqon, 2019) dengan judul "Implementasi Regresi Linier Berganda Untuk Prediksi Jumlah Peminat Mata Kuliah Pilihan". Penelitian ini dibuat dilatarbelakangi dengan banyaknya mata kuliah pilihan dengan masing masing keminatan yang membuat mahasiswa mengalami kesulitan dalam mengetahui jumah peminat yang ada. Oleh karena itu dibutuhkan sebuah sistem untuk prediksi jumlah peminat mata kuliah pilihan, dalam hal ini menggunakan algoritma regresi linier berganda.
3. (Wulandari, Sarja, & Saryanti, 2015) dengan judul "Prediksi Jumlah Pelanggan dan Persediaan Barang Menggunakan Metode Regresi Linier Berganda pada Bali Orchid". Penelitian ini dibuat untuk tujuan memprediksi jumlah pelanggan SPA agar dapat digunakan sebagai dasar pengambilan keputusan dimasa yang akan dating hari. Keputusan di sini dimaksudkan untuk menyediakan data kebutuhan dan keinginan pelanggan, karena barang-barang tersebut nantinya akan mempengaruhi minat pelanggan untuk memilihnya.

Berdasarkan penelitian-penelitian tersebut dapat ditarik kesimpulan bahwa peran dari Sistem Cerdas Prediksi Jumlah Kelas Program Studi Teknik Informatika Menggunakan Algoritma Regresi Linear Berganda sangat penting untuk membantu jurusan Teknik Informatika dalam menentukan kebutuhan jumlah kelas pada suatu mata kuliah.

2.2. Kecerdasan Buatan

Artificial Intelligence atau Kecerdasan Buatan adalah sebuah sistem computer yang mampu melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia. Proses yang terjadi dalam Artificial Intelligence mencakup learning, reasoning dan self-correction. Proses ini mirip dengan manusia yang melakukan analisis sebelum memberikan keputusan, (Lubis, 2021).

2.3. Regresi Linear Berganda

2.3.1 Model Regresi Linear Berganda

Regresi linear berganda merupakan model regresi yang melibatkan lebih dari satu variabel independen. Analisis regresi linear berganda dilakukan untuk mengetahui arah dan seberapa besar pengaruh variabel independen terhadap variabel dependen, (Ghozali, 2018) Ini adalah metode pokok di dalam ilmu statistik. Gunanya adalah untuk mengekspresikan kelas sebagai kombinasi linear dari atribut, dengan bobot yang telah di tentukan.

2.3.2 Operasi Hitung Regresi Linear Berganda

Regresi Linear Berganda memiliki rumus sebagai berikut :

$Y = a + b1x1 + b2X2 + \ldots + bnXn$

Di mana Y adalah kelas; X1, X2, ..., Xn adalah nilai atribut; dan a, b1, ..., bn adalah bobot. Bobot dihitung dari data sampel.

Dimana:

Y = Variabel tidak bebas (nilai yang diprediksikan)

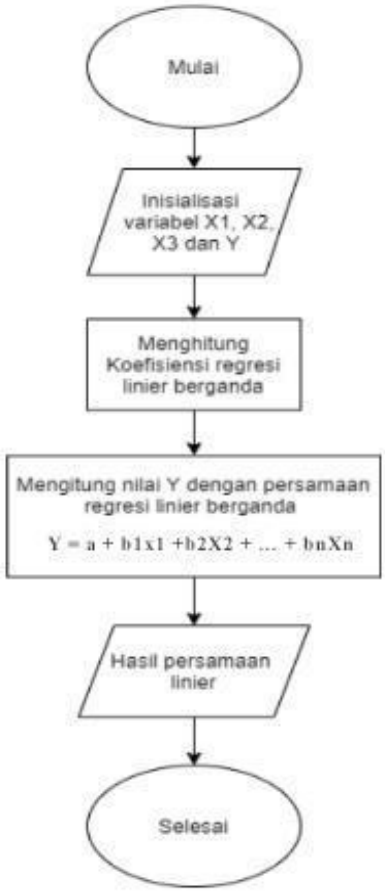
X= Variabel bebas

a = Konstanta (nilai Y apabila X1, X2... Xn = 0)

b = Koefisien regresi (nilai peningkatan atau penurunan)

2.3.3 Tahapan Implementasi Regresi Linear Berganda

Berikut ini (gambar 2.1) merupakan penggambaran secara grafik dari langkah-langkah prosedur dari satu program kerja secara keselurhan menggunakan metode Regresi Linear Berganda mulai dari awal sampai akhir prosesnya (Syahputra, Syahril, & Sobirin, 2020).



Gambar 2. 1 *Flowchart* Regresi Linear Berganda

Untuk mendapatkan hasil dari operasi hitung regresi linear berganda, diperlukan beberapa tahap seperti pada gambar 2.1, diantaranya: Inisalisasi variabel X_1, X_2,

X_3 dan Y. Kemudian menghitung koefiesinsi linear berganda menggunakan rumus Y = a + b1x1 +b2X2 + … + bnXn, dan selanjutnya akan didapatkan hasil persamaan linier.

2.4. Pemrograman Berbasis Website

2.4.1 *Pretext HyperProcessor* (PHP)

PHP adalah bahasa pemrograman yang umum digunakan untuk pemrosesan web. Tujuan PHP adalah membuat halaman web terlihat dinamis. Kode PHP dimasukkan ke dalam HTML. Saat ini sudah ada beberapa framework yang menggunakan bahasa pemrograman PHP seperti *CodeIgniter, Laravel, Yii Framework,* dll.

2.4.2 MySQL

MySQL adalah sistem manajemen basis data 'DBMS'. berarti sistem pemrosesan basis data. MySQL adalah alat manajemen sistem basis data open source. Pengelolaan sistem dasar seperti menambah, mengedit, menghapus membutuhkan sistem penanganan basis data. Kelebihan yang dimiliki MySQL antara lain:

1. Sistem Manajemen Basis Data Relasional (RDBMS) berarti bahwa MySQL adalah basis data relasional yang dapat menyimpan data dalam tabel terpisah alih-alih menyimpannya dalam satu ruang penyimpanan besar. Ini menambah kecepatan dan fleksibilitas.
2. Open source, merupakan aplikasi yang dapat digunakan oleh semua orang. Siapa pun dapat mengunduh perangkat lunak MySQL tanpa membayar.
3. Cross-platform, adalah perangkat lunak yang dapat digunakan oleh berbagai platform.

2.4.3 WebServer

Webserver adalah penyedia layanan yang bertindak sebagai alat untuk menghubungkan dan mentransfer data seperti *HyperText Markup Language (HTML)*, *Active Server Pages (ASP)*, *Pretext HyperText Processor (PHP)*, *Javascript*, dll.

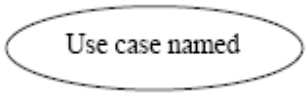
2.5. *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. *Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen - komponen yang diperlukan dalam sistem *software*. Diagram *Unified Modelling Language* (UML) (Suendri, 2018) antara lain sebagai berikut:

1. *Use Case* Diagram, *Use case* menggambarkan *external view* dari sistem yang akan kita buat modelnya. Model *use case* dapat dijabarkan dalam diagram *use case*, tetapi perlu diingat, diagram tidak indetik dengan model karena model lebih luas dari diagram. *Use case* harus mampu mengambarkan urutan aktor yang menghasilkan nilai terukur (Widodo, 2011)

Tabel 2.1 Simbol-simbol *UseCase*

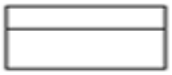
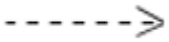
(Sumber : Widodo, 2011**)**

SIMBOL	NAMA	KETERANGAN	
	**Actor**	Actor adalah pengguna sistem. Actor tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan input atau memberikan *output*, maka aplikasi tersebut juga bisa dianggap sebagai actor.	
	***Use Case***		*Use case* digambarkan sebagai lingkaran elips dengan nama use case dituliskan didalam elips tersebut.
	Association		Asosiasi digunakan untuk menghubungkan actor dengan *use case.* Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara Actor dengan *Use Case.*

2. *Class Diagram*, Kelas sebagai suatu set objek yang memiliki atribut dan perilaku yang sama, kelas kadang disebut kelas objek (Jeffery L. Whitten, 2004). *Class* memiliki tiga area pokok yaitu :
    1. Nama, kelas harus mempunyai sebuah nama.
    2. Atribut, adalah kelengkapan yang melekat pada kelas. Nilai dari sekelas hanya bisa diproses sebatas atribut yang dimiliki.
    3. Operasi, adalah proses yang dapat dilakukan oleh sebuah kelas, baik pada kelas itu sendiri ataupun kepada kelas lainnya.

Tabel 2.2 Simbol *Class Diagram*

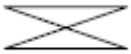
(Sumber : Whitten, 2004**)**

GAMBAR	NAMA	KETERANGAN
	Generalization	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	***N-Ary Association***	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	***Class***	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	***Collaboration***	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	***Realization***	Operasi yang benar-benar dilakukan oleh suatu objek.
	***Dependency***	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

3. *Activity Diagram*, Diagram actifitas menunjukkan aktifitas sistem dalam bentuk kumpulan aksi-aksi, bagaimana masing-masing aksi tersebut dimulai, keputusan yang mungkin terjadi hingga berakhirnya aksi. *Activity diagram* juga dapat menggambarkan proses lebih dari satu aksi salam waktu bersamaan. "Diagram *activity* adalah aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktifitas" (Haviluddin, 2011).

Tabel 2.3 Simbol *Activity Diagram*

(Sumber : Haviluddin, 2011**)**

SIMBOL	KETERANGAN
	Titik Awal
	Titik Akhir
	*Activity*
	Pilihan Untuk mengambil Keputusan
	*Fork*; Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan peralel menjadi satu.
	*Rake*; Menunjukkan adanya dekomposisi
	Tanda Waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

2.6 *Data Flow Diagram (DFD)*

DFD adalah diagram yang menggambarkan proses aliran data input/output dari sebuah sistem informasi yang dibangun. *DFD* memiliki komponen-komponen diantaranya :

1. User/Terminator

   Kesatuan diluar sistem (*external entity*) yang memberikan input ke sistem atau menerima output dari sistem berupa orang, organisasi, atau sistem lain.

2. Process

   Aktivitas yang mengolah input menjadi output.

3. Data *Flow*

   Aliran data pada sistem (antar proses, antara terminator & proses, serta antara proses & data store).

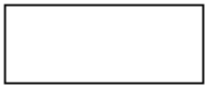
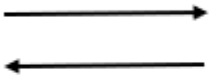
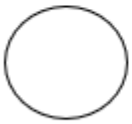
4. Data *Store*

   Penyimpanan data pada database, biasanya berupa tabel.

Adapun simbol-simbol *DFD* disajikan pada gambar 2.4.

Gane/Sarson	Yourdon/De Marco	Keterangan
Entitas Eksternal	Entitas Eksternal	Entitas eksternal dapat berupa orang/unit terkait yang berinterkasi dengan sistem tetapi di luar sistem.
Proses	Proses	Orang/unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diidentifikasikan.
Aliran Data	Aliran Data	Aliran data dengan arah khusus dari sumber ke tujuan
Data Store	Data Store	Penyimpanan data atau tempat data dilihat oleh proses.

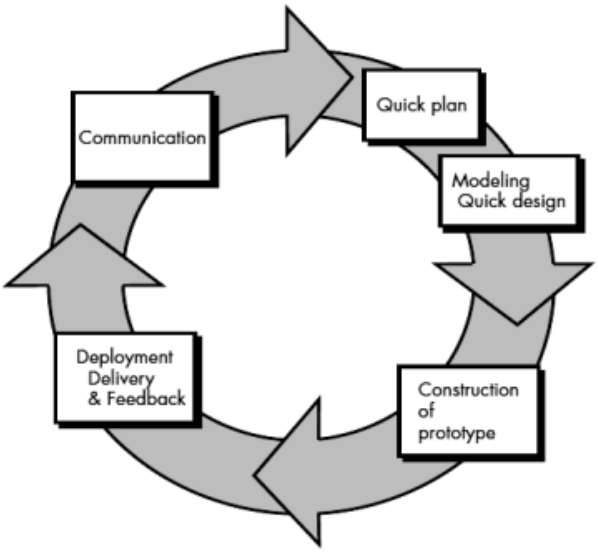
Tabel 2.4 Simbol *DFD* dan Keterangan

Simbol	Keterangan Fungsi
	*External entity*/Entitas luar. Simbol ini menunjukkan orang, organisasi, atau sistem yang berada di luar sistem tetapi berinteraksi dengan sistem.
	*Data Flow* diberi simbol panah. Simbol ini menunjukkan satu data tunggal atau kumpulan logis suatu data, selalu diawali atau diakhiri pada suatu proses.
	Proses adalah aktivitas atau fungsi yang dilakukan untuk alasan bisnis yang spesifik, bisa berupa manual maupun terkomputerisasi.
	*Data Store* adalah kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam *data store*.

Tabel 2.5 Fungsi Simbol *ERD*

2.7. Metode Pengembangan Sistem *Prototype*

Metode *prototype* merupakan sebuah metode pengembangan system yang terdiri dari beberapa tahapan disajikan pada gambar 2.2 (Roger, 2012)



Gambar 2.2 Metode *Prototype*

Perancangan sistem menurut metode ini dimulai dari tahapan komunikasi (c*ommunication*), perencanaan secara cepat (*quick plan*), pemodelan perancangan secara cepat (*modeling quick design)*, pembentukan prototipe (*construction of prototype*) dan penyerahan sistem/perangkat lunak kepada pengguna, pengiriman dan umpan balik (*deployment delivery & feedback*).

1.Komunikasi (*Communcation*)

Tahapan komunikasi dilakukan antara tim pengembang perangkat lunak dan pengguna. Tujuan dari tahapan ini yakni agar tim pengembang dapat mengetahui kebutuhan pengguna secara keseluruhan. Tahapan komunikasi dapat dilakukan dengan cara mengajukan pertanyaan seputar sistem kepada pengguna atau wawancara.

2.Perancangan secara cepat (*Quick plan*)

Pada tahapan ini dilakukan diskusi hasil komunikasi atau wawancara yang dilakukan dengan calon pengguna. Tujuannya yakni agar semua anggota tim memiliki gambaran yang sama mengenai perangkat lunak yang akan dibangun.

3.Permodelan perancangan secara cepat (*Modeling quick design*)

Tahapan ini merupakan kelanjutan dari hasil diskusi antara tim pengambang dan pengguna. Pada tahapan ini anggota tim telah memiliki gambaran yang sama mengenai perangkat lunak yang akan dibuat sehingga tim pengembang dapat mulai membuat desain rancangan perangkat lunak. Perncangan yang dibuat berfokus pada semua aspek perangkat lunak yang akan terlihat oleh pengguna seperti desain antarmuka perangkat lunak, *flowchart, DFD,* dll.

4.Pembentukan prototype (*Construction of prototype*)

Tahapan ini merupakan tahapan implementasi dari *prototype* yang telah dibuat. Pada tahapan ini para pengembang perangkat lunak melakukan pengkodean untuk membangung perangkat lunak sesuai dengan rancangan dengan bahasa pemrograman tertentu.

5.Penyerahan, pengiriman dan umpan balik (*Deployment delivery & feedback*)

Pada tahapan ini tim pengembang menyerahkan perangkat lunak kepada calon pengguna untuk dilakukan pengujian dan evaluasi. Setelah melakukan pengujian

maka selanjutnya calon pengguna akan memberikan umpan balik kepada tim pengembang. Umpan balik yang dimakud yakni masukan mengenai fitur-fitur yang masih kurang sesuai. Tujuannya yakni untuk memperhalus spesifikasi kebutuhan sehingga terwujud sebuah perangkat lunak yang sesuai dengan kebutuhan pengguna.

2.8 Pengujian sistem

Pengujian sistem adalah pengujian program perangkat lunak yang lengkap dan terintegrasi. Pengujian perangkat lunak dapat dibedakan menjadi dua yaitu *Black Box Testing* dan *White Box Testing*.

2.8.1 *Black box testing*

Black Box Testing atau yang sering dikenal dengan sebutan pengujian fungsional merupakan metode pengujian Perangkat Lunak yang digunakan untuk menguji perangkat lunak tanpa mengetahui struktur internal kode atau Program. *Blackbox testing* dilakukan hanya dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari aplikasi yang sedang dikembangkan. Jadi dapat dianalogikan seperti sedang melihat suatu kotak hitam dimana hanya bisa melihat penampilan pada luarnya saja, tanpa mengetahui apa yang ada dibalik kotak hitam tersebut. Pengujian *black box testing* ini mengevaluasi hanya pada tampilan luarnya saja (*interface*), fungsionalnya, dan tidak melihat atau mengetahui apa yang sesungguhnya terjadi di dalam proses detailnya.

2.8.2 Kesalahan yang akan ditemukan menggunakan *Black Box Testing*

1. Kesalahan Fungsi

Kesalahan ini merupakan hal yang seringkali terjadi dalam perangkat lunak. Hal ini diakibatkan karena kesalahan seorang programmer dalam menuliskan kode program atau kesalahan dalam menerapkan algoritma.

2. Kesalahan *Interface*

Hal ini terkait dengan fungsionalitas dari elemen-elemen interface yang terdapat pada tiap aplikasi seperti halnya input pada form, tombol, label notifikasi, dll.

3. Kesalahan Kinerja

Kesalahan ini biasanya ditandai dengan sulitnya suatu halaman aplikasi untuk dimuat. Halaman aplikasi yang sulit dimuat tersebut tentunya memiliki kesalahan pada kode programnya.