BAB 2
LANDASAN TEORI

2.1 *Intelligent Tutoring System*

Intelligent Tutoring System (ITS) adalah sistem pembelajaran berbantuan komputer yang menyajikan dukungan berdasarkan kondisi pengguna. ITS berbeda dari aplikasi latihan statis karena menyimpan representasi kemampuan pengguna, memilih tindakan pembelajaran, dan memberikan umpan balik yang disesuaikan. Dalam pendidikan pemrograman, ITS dapat membantu pengguna yang mengalami kesulitan berbeda pada sintaks, logika, penelusuran kode, dan perbaikan kesalahan. Sharma dan Harkishan membahas desain ITS khusus untuk pembelajaran pemrograman [3]. Zhong dan Zhan menekankan peran umpan balik informatif dalam pengembangan ITS untuk pendidikan pemrograman [4]. SSDTutor memberikan contoh ITS yang menggunakan umpan balik untuk mendukung pembelajaran pengembangan perangkat lunak aman [20].

Secara konseptual, ITS terdiri atas empat komponen utama. *Domain model* menyimpan pengetahuan yang dipelajari, seperti topik, soal, jawaban, penjelasan, dan *hint*. *Student model* menyimpan kondisi pengguna berdasarkan riwayat interaksi. *Tutoring model* menentukan strategi seperti pemilihan soal, pemberian bantuan, dan rekomendasi. *Interface model* menyediakan media interaksi antara pengguna dan sistem. Keempat komponen tersebut harus terhubung agar sistem tidak hanya menampilkan konten, tetapi juga dapat mengambil keputusan pembelajaran. Zhong dan Zhan menunjukkan bahwa kualitas umpan balik merupakan bagian penting dari ITS pemrograman [4]. Lai dan Lin menganalisis perbedaan perilaku serta hasil belajar berdasarkan tingkat pengetahuan awal pengguna [10]. Fodouop Kouam membahas dukungan ITS terhadap peserta dengan pengalaman pemrograman yang beragam [9].

Pada penelitian ini, *domain model* diwujudkan melalui bank 5.500 soal Java; *student model* menyimpan *mastery*, jumlah jawaban, penggunaan *hint*, dan riwayat terbaru; *tutoring model* diimplementasikan pada *adaptive engine*; sedangkan *interface model* dibangun menggunakan Streamlit.

2.2 Pembelajaran Java untuk Siswa SMA/SMK

Target pengguna penelitian adalah siswa SMA/SMK pada jenjang pendidikan menengah atas. Rentang usia yang umum untuk jenjang ini adalah 15–18 tahun [2]. Capaian Pembelajaran Informatika Fase E menuntut peserta didik menerapkan konsep pemrograman prosedural dan mengembangkan program terstruktur [1]. Aplikasi penelitian ditempatkan sebagai media latihan tambahan setelah peserta didik memperoleh pengantar algoritma, variabel, percabangan, dan perulangan.

Java dipilih karena merupakan bahasa berorientasi objek, bertipe kuat, dan memiliki struktur kelas yang eksplisit sesuai spesifikasi bahasanya [7]. Karakter tersebut memungkinkan materi disusun dari konsep dasar menuju kelas dan objek. Java juga digunakan pada AP Computer Science A untuk siswa sekolah menengah, sehingga tersedia contoh bahwa Java dapat dipakai dalam pendidikan tingkat tersebut [8]. Di sisi lain, Java memiliki sintaks tekstual yang menuntut ketelitian. Tinjauan sistematis mengenai penilaian dan umpan balik pemrograman menunjukkan pentingnya umpan balik yang tepat waktu dan dapat ditindaklanjuti [21], sedangkan sistem pembangkit *hint* untuk latihan Java menunjukkan bahwa petunjuk dapat dirancang untuk membantu pengguna tanpa langsung memberikan solusi lengkap [22]. Oleh karena itu, aplikasi menyediakan soal bertingkat, *hint*, dan penjelasan agar kompleksitas tidak diberikan sekaligus.

2.3 Pembelajaran Adaptif Hibrida

Pembelajaran adaptif adalah penyesuaian konten atau strategi berdasarkan data pengguna. Adaptasi dapat bersifat berbasis aturan, berbasis model prediksi, atau gabungan keduanya. Dalam kerangka AI untuk pendidikan, keputusan sistem perlu tetap diarahkan pada kebutuhan pembelajaran dan hubungan antara peserta didik, pengajar, serta teknologi [18]. Villegas-Ch dkk. menunjukkan pentingnya personalisasi dan umpan balik dalam ITS adaptif [19]. Lai dan Lin menunjukkan bahwa tingkat pengetahuan awal berkaitan dengan perilaku serta hasil belajar pengguna ITS [10]. Yang dkk. memberikan contoh tutor pemrograman interaktif yang memanfaatkan percakapan dan umpan balik berbasis AI [11]. Sistem pada penelitian ini menggunakan pendekatan hibrida karena dua mekanisme bekerja pada tingkat yang berbeda.

Pertama, pemilihan soal di dalam sesi menggunakan prediksi peluang

keberhasilan dari *Random Forest* yang digabungkan dengan kondisi *student model*. Kandidat dipilih dari soal yang sesuai dengan topik atau mode campuran, tingkat kesulitan sesi, dan belum pernah dikerjakan. Kedua, perubahan tingkat kesulitan setelah sesi menggunakan aturan pedagogis. Jika pengguna menjawab minimal 17 dari 20 soal dengan benar, tingkat kesulitan dinaikkan satu level. Jika jawaban benar kurang dari 12, tingkat kesulitan diturunkan satu level. Hasil 12–16 jawaban benar mempertahankan level.

Pendekatan tersebut tetap termasuk adaptif karena keluaran sistem berbeda berdasarkan riwayat masing-masing pengguna. Namun, penting untuk menyatakan bahwa *Random Forest* tidak langsung menghasilkan kelas kesulitan 1–5. Model menghasilkan peluang keberhasilan, sedangkan aturan dan filter aplikasi mengubah peluang tersebut menjadi tindakan pemilihan soal.

2.4 *Student Model*

Student model adalah representasi kondisi belajar pengguna. Atribut yang digunakan dalam aplikasi meliputi jumlah percobaan, jumlah benar dan salah, penggunaan *hint*, *mastery* per topik, riwayat 20 jawaban terakhir, *current streak*, soal yang telah dijawab, dan topik yang direkomendasikan. Konsep ini berkaitan dengan *knowledge tracing*, yaitu estimasi kondisi pengetahuan berdasarkan urutan interaksi belajar. Dai dkk. meninjau perkembangan teknologi *knowledge tracing* untuk memperkirakan kondisi pengetahuan dari interaksi belajar [23]. Shen dkk. mengelompokkan model, variasi, dan penerapan *knowledge tracing* yang memanfaatkan urutan riwayat pengguna [24]. Pada tugas pemrograman, Code-DKT menggunakan representasi kode untuk pemodelan pengetahuan [25]. Zhu dkk. memperkenalkan dataset dan model untuk *programming knowledge tracing* [26]. Duan dkk. menggunakan informasi kasus uji untuk memodelkan pengetahuan pada tugas pemrograman terbuka [27].

Nilai *mastery* berada pada rentang 0,02–0,98. Setelah jawaban benar, nilai bertambah secara bertahap; kenaikan lebih besar diberikan jika pengguna tidak memakai *hint*. Setelah jawaban salah, nilai berkurang sesuai penalti yang juga mempertimbangkan jumlah *hint*. Pembaruan ini digunakan oleh sistem langsung, sedangkan model *Random Forest* memanfaatkan fitur riwayat yang dibentuk dari dataset pelatihan.

$$M' = \min(0{,}98,\ M + g(1{,}05 - M)) \tag{2.1}$$

Pada Rumus 2.1, M adalah *mastery* lama dan g adalah faktor kenaikan. Implementasi menggunakan $g = 0{,}095$ tanpa *hint* dan $g = 0{,}045$ ketika *hint* digunakan.

$$M' = \max(0{,}02,\ M - (0{,}075 + 0{,}025h)M) \tag{2.2}$$

Pada Rumus 2.2, h adalah jumlah *hint* yang digunakan pada soal tersebut.

2.5 *Random Forest*

Random Forest adalah metode ensemble yang menggabungkan banyak pohon keputusan. Setiap pohon dibangun dari sampel bootstrap data latih dan hanya mempertimbangkan subset acak fitur ketika menentukan pemisahan node. Variasi antarpohon mengurangi ketergantungan pada satu struktur pohon, sedangkan agregasi prediksi meningkatkan kestabilan model. Pan dan Dai menerapkan Random Forest untuk prediksi kinerja peserta didik pada data pendidikan [16]. Survei Batool dkk. menunjukkan bahwa hasil model prediksi akademik perlu dibaca bersama fitur dan konteks data yang digunakan [15]. Tinjauan Daza dkk. juga memperlihatkan bahwa tidak ada satu teknik data mining yang selalu unggul untuk seluruh konteks prediksi akademik [14].

2.5.1 Node, Cabang, dan Daun pada Pohon Keputusan

Satu pohon keputusan terdiri atas node akar, node internal, cabang, dan daun. Node internal memeriksa kondisi suatu fitur, misalnya apakah rata-rata percobaan lima soal terakhir lebih kecil dari ambang tertentu. Cabang merepresentasikan hasil kondisi, sedangkan daun menyimpan distribusi kelas. Pada klasifikasi biner penelitian ini, kelas 1 adalah berhasil dan kelas 0 adalah kesulitan.

Pemilihan pemisahan node pada pohon klasifikasi umumnya meminimalkan impuritas. Salah satu ukuran yang digunakan adalah impuritas Gini.

$$G = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.3}$$

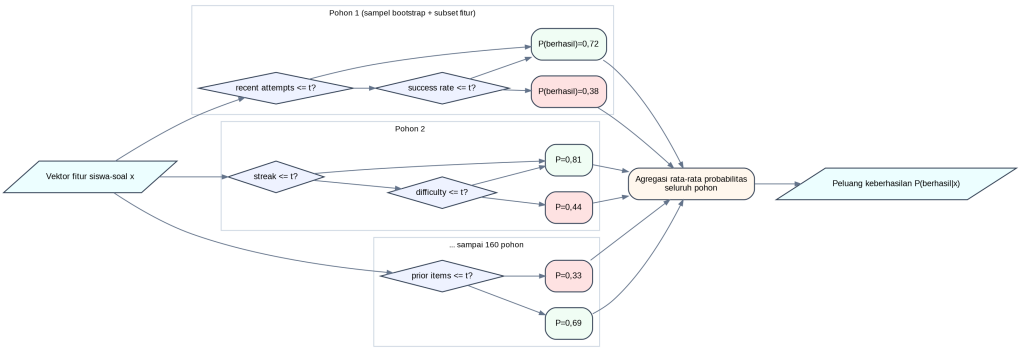
Nilai p_k adalah proporsi data kelas ke-k pada suatu node. Pemisahan dipilih apabila menghasilkan penurunan impuritas yang besar pada node anak.

2.5.2 Agregasi Banyak Pohon

Model penelitian menggunakan 160 pohon dengan kedalaman maksimum 12 dan minimal 4 sampel pada daun. Setiap pohon dapat menghasilkan probabilitas berbeda. Probabilitas akhir diperoleh dari rata-rata probabilitas seluruh pohon.

$$P(y = 1 \mid x) = \frac{1}{T} \sum_{t=1}^{T} P_t(y = 1 \mid x) \tag{2.4}$$

T adalah jumlah pohon dan P_t adalah probabilitas dari pohon ke-t. Hubungan antara node, pohon, dan agregasi ditunjukkan pada Gambar 2.1.



Gambar 2.1. Ilustrasi node dan agregasi pohon pada Random Forest

2.5.3 Kelebihan dan Keterbatasan

Random Forest sesuai untuk data tabular dengan hubungan nonlinier dan interaksi antarfaktor. Model juga menyediakan nilai kepentingan fitur yang dapat digunakan sebagai penjelasan global. Keterbatasannya adalah keputusan satu prediksi lebih sulit dibaca dibandingkan satu pohon, model dapat bias terhadap pola dataset sumber, dan probabilitasnya tetap perlu dievaluasi melalui kalibrasi serta metrik yang sesuai. Penelitian prediksi dini dalam pembelajaran pemrograman juga menunjukkan bahwa performa model perlu dibaca bersama ketersediaan fitur, waktu prediksi, dan konteks penggunaan [17].

2.6 Dataset CodeWorkout dan CSEDM

Dataset yang digunakan adalah CodeWorkout Fall 2019 dari *2nd CSEDM Data Challenge 2021*. Edwards dkk. meninjau data CSEDM dan karakteristik dataset publik untuk pembelajaran pemrograman pengantar [13]. Dataset yang digunakan dalam penelitian ini dipublikasikan oleh penyelenggara *2nd CSEDM Data Challenge 2021* [12]. Berkas dataset yang digunakan terdiri atas `MainTable.csv`, `early.csv`, `late.csv`, dan `DatasetMetadata.csv`.

`MainTable.csv` memuat 360.176 event dari 506 identitas anonim, 5 assignment, dan 50 problem. Rentang waktu pada data adalah 23 September 2019 sampai 5 Desember 2019. `early.csv` berisi 14.317 pasangan siswa-soal pada 30 problem awal, sedangkan `late.csv` berisi 9.386 pasangan pada 20 problem berikutnya. Setelah digabungkan, tersedia 23.703 baris dari 494 siswa yang memiliki label.

Kolom `Attempts` menunjukkan jumlah percobaan pada soal, `CorrectEventually` menunjukkan apakah soal akhirnya diselesaikan, dan `Label` adalah target berhasil atau kesulitan. Ketiga kolom tersebut tidak memiliki makna sebagai tingkat kesulitan soal 1–5. Dalam model aman, `Attempts` dan `CorrectEventually` pada baris yang sedang diprediksi tidak digunakan karena keduanya merangkum hasil soal saat ini dan dapat menyebabkan *data leakage*.

2.7 Fitur Model dan Pencegahan *Data Leakage*

Model menggunakan 12 fitur yang seluruhnya dapat dibentuk sebelum hasil soal saat ini diketahui. Fitur ditunjukkan pada Tabel 2.1.

Tabel 2.1. Fitur yang digunakan oleh model Random Forest

Fitur	**Makna**
phase_code	Fase early atau late.
assignment_order_norm	Urutan assignment yang dinormalisasi.
problem_order_norm	Urutan problem yang dinormalisasi.
estimated_difficulty	Proksi kesulitan berdasarkan urutan kurikuler, bukan hasil baris saat ini.
student_prior_items	Jumlah soal yang telah dikerjakan sebelumnya.

Tabel 2.1. Fitur yang digunakan oleh model Random Forest (lanjutan)

Fitur	**Makna**
student_prior_success_rate	Proporsi keberhasilan pada seluruh soal sebelumnya.
student_prior_mean_attempts	Rata-rata percobaan pada soal sebelumnya.
student_prior_incorrect_rate	Proporsi kegagalan sebelumnya.
student_recent_success_rate	Keberhasilan pada lima soal terakhir.
student_recent_mean_attempts	Rata-rata percobaan pada lima soal terakhir.
student_current_streak	Urutan keberhasilan atau kegagalan sebelum soal saat ini.
student_prior_compile_success_rate	Riwayat keberhasilan kompilasi; pada integrasi aplikasi diisi nilai netral 0,5.

Pemisahan data dilakukan berdasarkan `SubjectID` menggunakan *GroupShuffleSplit*. Strategi ini mencegah satu siswa muncul pada data latih dan validasi sekaligus. Sebanyak 395 siswa atau 19.018 baris digunakan untuk pelatihan, sedangkan 99 siswa atau 4.685 baris digunakan untuk validasi.

2.8 Integrasi Model ke Pemilihan Soal

Bank soal aplikasi tidak sama dengan soal CodeWorkout. Oleh karena itu, integrasi dilakukan melalui pemetaan fitur, bukan melalui pencocokan identitas soal. Tingkat kesulitan soal aplikasi dipetakan ke estimated_difficulty; nomor soal dinormalisasi menjadi urutan problem; dan riwayat pengguna aplikasi dipetakan ke fitur keberhasilan, percobaan, serta *streak*.

Untuk mengurangi risiko perbedaan domain, probabilitas model historis digabungkan dengan probabilitas fallback berbasis *mastery*. Implementasi menggunakan bobot 0,62 untuk model dan 0,38 untuk fallback.

$$P_{app} = 0{,}62P_{RF} + 0{,}38P_{mastery} \tag{2.5}$$

Probabilitas tersebut bukan skor akhir tunggal. Sistem memprioritaskan kandidat yang berada pada rentang peluang berhasil 0,65–0,80, sesuai tingkat kesulitan sesi, belum pernah dikerjakan, dan berkaitan dengan topik yang

memerlukan latihan. Rentang tersebut dimaksudkan untuk menghindari soal yang terlalu mudah maupun terlalu sulit.

2.9 *Explainable Artificial Intelligence*

Explainable Artificial Intelligence (XAI) digunakan untuk membantu pembaca memahami faktor yang berpengaruh pada prediksi. Dalam pendidikan, penjelasan perlu disesuaikan dengan pemangku kepentingan karena kebutuhan siswa, guru, peneliti, dan pengembang tidak selalu sama [28]. Penelitian ini menggunakan dua tingkat penjelasan. Penjelasan global menggunakan kepentingan fitur dari *Random Forest* untuk menunjukkan fitur yang paling sering berkontribusi pada penurunan impuritas. Penjelasan lokal dapat dikembangkan menggunakan metode atribusi fitur untuk menerangkan satu prediksi tertentu.

Kepentingan fitur tidak membuktikan hubungan kausal. Nilai tersebut hanya menjelaskan pola yang dipelajari model dari dataset. Oleh karena itu, XAI digunakan untuk audit dan komunikasi model, bukan untuk menyatakan bahwa suatu perilaku pasti menyebabkan keberhasilan siswa.

2.10 Metrik Evaluasi Model

Evaluasi model tidak hanya menggunakan akurasi karena distribusi label tidak seimbang. Dari 23.703 baris, 17.575 berlabel berhasil dan 6.128 berlabel kesulitan. Metrik yang digunakan adalah sebagai berikut.

1. *Confusion matrix* menunjukkan jumlah prediksi benar dan salah untuk setiap kelas.
2. *Precision* menunjukkan proporsi prediksi kelas positif yang benar.
3. *Recall* menunjukkan proporsi kelas positif yang berhasil ditemukan.
4. *F1-score* adalah rata-rata harmonik *precision* dan *recall*.
5. AUC mengukur kemampuan model mengurutkan contoh positif lebih tinggi daripada negatif pada berbagai ambang.
6. *Brier score* mengukur kesalahan kuadrat probabilitas; nilai lebih kecil menunjukkan probabilitas yang lebih baik.

7. *Macro F1* memberi bobot setara pada kedua kelas dan membantu membaca kinerja pada kelas kesulitan yang lebih sedikit.

2.11 Pengujian Aplikasi dan Respons Pengguna

Pengujian aplikasi terdiri atas pengujian fungsional dan pengujian mekanisme adaptif. Pengujian fungsional menggunakan pendekatan *black-box* untuk memastikan input menghasilkan keluaran yang diharapkan. Pengujian mekanisme adaptif memeriksa filter kandidat, larangan pengulangan soal, penggunaan model atau fallback, serta perubahan level setelah sesi.

Respons pengguna dinilai dengan skala Likert 1–5. Data evaluasi penelitian ini mencakup 40 responden dan ditempatkan sebagai evaluasi formatif-deskriptif, bukan survei inferensial populasi. Rekomendasi metodologis untuk studi usability kuantitatif menyebutkan bahwa sekitar 40 peserta sering digunakan untuk memperoleh estimasi awal yang lebih stabil pada banyak skenario pengukuran, tetapi jumlah tersebut tetap tidak otomatis menjamin keterwakilan populasi [29]. Oleh karena itu, teknik pemilihan responden, kesesuaian karakteristik dengan target pengguna, dan batas generalisasi tetap dinyatakan secara eksplisit.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.6}$$

$$P = \frac{\bar{x}}{5} \times 100\% \tag{2.7}$$

2.12 Metode Pengembangan *Waterfall*

Metode *Waterfall* membagi pengembangan ke dalam tahap kebutuhan, perancangan, implementasi, integrasi dan pengujian, serta pemeliharaan. Tahapan disusun berurutan agar keluaran satu tahap menjadi dasar tahap berikutnya. Perbandingan model siklus hidup pengembangan sistem menunjukkan bahwa Waterfall sesuai ketika kebutuhan dan dokumentasi tahapan perlu dinyatakan secara terstruktur, walaupun perubahan tetap perlu dikelola ketika ditemukan ketidaksesuaian pada tahap berikutnya [30]. Pada penelitian ini, metode digunakan secara terstruktur tetapi koreksi teknis tetap dapat mengembalikan pekerjaan ke

tahap desain apabila ditemukan ketidaksesuaian.

2.13 Teknologi Implementasi

Aplikasi dibangun dengan Python dan Streamlit. Data konten disimpan dalam JSON, progres lokal menggunakan SQLite, dan Supabase disediakan sebagai opsi penyimpanan daring. Pelatihan dan pemuatan model menggunakan pandas, NumPy, *scikit-learn*, dan joblib. Visualisasi menggunakan Plotly dan Matplotlib. Pemilihan teknologi tersebut mendukung pengolahan data, pemodelan tabular, serta pengembangan prototipe web dalam satu ekosistem.