BAB 2
LANDASAN TEORI

Bab ini menguraikan konsep-konsep dasar yang menjadi landasan penelitian. Pembahasan mencakup definisi *engagement*, prinsip *machine learning* untuk klasifikasi, deskripsi algoritma yang digunakan, metrik evaluasi, dan metode *feature importance*.

2.1 Engagement

Dalam studi *game* digital, *engagement* merujuk pada sejauh mana pemain menunjukkan keterlibatan aktif dengan suatu permainan. Indikator umum meliputi frekuensi bermain, durasi per sesi, dan konsistensi interaksi [6]. Semakin tinggi nilai-nilai ini, semakin besar kemungkinan pemain memiliki *engagement* yang kuat. Penelitian ini mengadopsi pembagian tiga tingkatan *low*, *medium*, *high* yang umum digunakan dalam literatur [6]. Pengelompokan ini memungkinkan analisis yang lebih terstruktur terhadap perbedaan perilaku antar kelompok. Memahami *engagement* bukan sekadar kepentingan akademis. Bagi pengembang, informasi ini dapat menjadi dasar untuk merancang fitur yang lebih menarik, menyesuaikan tingkat kesulitan, atau mengidentifikasi pemain yang berisiko berhenti bermain [6]. Dengan kata lain, *engagement* adalah jembatan antara data perilaku dan strategi pengembangan produk.

2.2 Data Preparation

Sebelum data digunakan untuk proses pelatihan model, diperlukan tahap *data preparation* agar data berada dalam format yang sesuai untuk diproses oleh algoritma *machine learning*. Tahap ini bertujuan untuk meningkatkan kualitas data serta memastikan setiap algoritma dapat mempelajari pola dari data secara optimal.

2.2.1 Label Encoding

Label Encoding merupakan salah satu teknik *feature engineering* atau prapemrosesan data yang digunakan untuk mengubah data kategorikal menjadi bentuk numerik agar dapat diproses oleh algoritma *machine learning*. Hal ini dilakukan karena sebagian besar algoritma *machine learning* dirancang untuk

mengolah data dalam bentuk numerik sehingga data yang masih berupa teks atau label perlu ditransformasikan terlebih dahulu sebelum proses pelatihan model dilakukan [7]. Proses *Label Encoding* dilakukan dengan mengidentifikasi seluruh kategori unik pada suatu atribut, kemudian setiap kategori dipetakan ke sebuah nilai bilangan bulat yang unik [7]. Contoh Label encoding tersebut dapat dilihat pada Tabel 2.1.

Tabel 2.1. Contoh Hasil Proses Label Encoding

Kategori	**Nilai Encoding**
merah	0
biru	1
hijau	2

Berdasarkan Tabel 2.1, setiap kategori diberikan nilai numerik yang berbeda sebagai representasi data kategorikal seperti merah menjadi 0, biru menjadi 1, dan hijau menjadi 2. Proses ini memungkinkan data yang semula berbentuk teks diolah oleh algoritma *machine learning* tanpa mengubah informasi yang terkandung pada masing-masing kategori.

2.2.2 Train & Test Split

Train-test split merupakan metode yang digunakan untuk membagi dataset menjadi data latih (*training set*) dan data uji (*testing set*). Data latih digunakan untuk membangun model *machine learning*, sedangkan data uji digunakan untuk mengevaluasi kemampuan model dalam melakukan prediksi pada data yang belum pernah dilihat sebelumnya. Rasio pembagian untuk *training set* dan *testing set* juga perlu dipertimbangkan karena dapat memengaruhi kinerja model. Ukuran *training set* yang lebih besar memungkinkan model untuk belajar secara lebih efektif dari data, sementara *testing set* yang banyak memastikan evaluasi yang adil terhadap efektivitas model. Penentuan rasio pembagian data juga bergantung pada apa yang ingin dicapai oleh model *Machine Learning* yang digunakan [8].

2.2.3 Variance Inflation Factor (VIF)

Variance Inflation Factor (VIF) merupakan metode yang digunakan untuk mendeteksi adanya multikolinearitas antar variabel independen dalam suatu dataset.

Multikolinearitas terjadi ketika dua atau lebih variabel independen memiliki hubungan yang kuat sehingga dapat memengaruhi kestabilan estimasi parameter pada model regresi. Oleh karena itu, VIF digunakan untuk mengevaluasi tingkat multikolinearitas antar variabel independen sebelum proses pelatihan model. Pada penelitian ini, analisis VIF dilakukan untuk memastikan bahwa variabel yang digunakan tidak memiliki tingkat multikolinearitas yang tinggi, khususnya karena salah satu algoritma yang digunakan adalah *Logistic Regression*, sehingga hasil pemodelan dapat diinterpretasikan dengan lebih baik [9].

Nilai VIF dihitung berdasarkan koefisien determinasi (R^2) yang diperoleh dari hasil regresi suatu variabel independen terhadap variabel independen lainnya. Persamaan VIF ditunjukkan pada Persamaan berikut.

$$VIF = \frac{1}{1 - R^2} \tag{2.1}$$

Semakin besar nilai VIF, semakin tinggi tingkat multikolinearitas pada variabel tersebut. Secara umum, nilai VIF kurang dari 5 menunjukkan bahwa multikolinearitas relatif rendah, sedangkan nilai VIF di atas 5 mengindikasikan adanya multikolinearitas yang perlu diperhatikan. Pada beberapa penelitian, batas yang lebih longgar yaitu VIF sebesar 10 juga masih digunakan sebagai indikator multikolinearitas yang tinggi.

2.3 Machine Learning

Machine learning adalah cabang dari kecerdasan buatan yang berfokus pada pengembangan algoritma yang dapat belajar dari data [10]. Berbeda dengan pemrograman konvensional, pendekatan ini tidak memerlukan aturan yang ditentukan secara eksplisit, melainkan mempelajari pola dari data untuk menghasilkan prediksi atau keputusan. Selain itu, *machine learning* hadir dalam berbagai bentuk. Salah satu paradigma utama dalam *machine learning* adalah *supervised learning*, yaitu proses pelatihan model menggunakan data yang telah memiliki label [11].

Salah satu penerapan *supervised learning* adalah klasifikasi (*classification*). Berbeda dengan *regression tasks* dimana model mengeluarkan hasil berupa kontinu, *classification* memprediksi atau mengelompokkan suatu data ke dalam kelas tertentu berdasarkan pola yang dipelajari dari data berlabel [12]. Dalam penelitian ini, *machine learning* digunakan untuk mengklasifikasikan pemain ke dalam kategori *Low Engagement*, *Medium Engagement*, dan *High Engagement*. Selain

menghasilkan prediksi, penelitian ini juga menganalisis faktor-faktor yang paling berpengaruh terhadap hasil klasifikasi untuk memahami karakteristik perilaku pemain [13].

2.3.1 Decision Tree

Decision Tree merupakan salah satu algoritma *supervised learning* yang banyak digunakan untuk menyelesaikan permasalahan klasifikasi maupun regresi. Algoritma ini membentuk model prediksi dalam struktur menyerupai pohon (*tree*) yang terdiri atas *root node*, *internal node*, cabang (*branch*), dan *leaf node*. Setiap jalur dari *root node* hingga *leaf node* merepresentasikan serangkaian aturan keputusan yang digunakan untuk menentukan kelas suatu data. Struktur tersebut membuat *Decision Tree* mudah dipahami dan diinterpretasikan dibandingkan banyak algoritma klasifikasi lainnya [14].

Proses pembentukan *Decision Tree* dimulai dengan menempatkan seluruh data latih pada *root node*. Selanjutnya, algoritma memilih atribut yang paling sesuai untuk memisahkan data sehingga setiap cabang memiliki karakteristik yang semakin homogen atau memiliki label kelas yang semakin seragam. Proses pemisahan (*splitting*) dilakukan secara berulang pada setiap node hingga memenuhi kriteria penghentian tertentu, misalnya seluruh data pada suatu node berasal dari kelas yang sama atau pohon telah mencapai batas yang ditentukan. Setelah proses pembentukan selesai, model yang dihasilkan dapat digunakan untuk mengklasifikasikan data baru berdasarkan aturan keputusan yang terbentuk [14].

Dalam proses pembentukan *Decision Tree*, pemilihan atribut terbaik dilakukan menggunakan kriteria *splitting* untuk menghasilkan pemisahan data yang optimal. Beberapa kriteria yang umum digunakan adalah *Gini Index*, *Entropy*, dan *Log Loss*. Kriteria tersebut digunakan untuk mengukur kualitas pemisahan data sehingga algoritma dapat memilih atribut yang menghasilkan node paling homogen [15].

Salah satu kriteria yang umum digunakan adalah *Gini Index*, yaitu ukuran yang digunakan untuk menghitung tingkat ketidakmurnian (*impurity*) suatu node. Nilai *Gini Index* yang semakin kecil menunjukkan bahwa data pada node tersebut semakin homogen atau didominasi oleh satu kelas. Persamaan *Gini Index* ditunjukkan pada Persamaan 2.2.

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{2.2}$$

- D = himpunan data pada suatu node,
- p_i = proporsi data pada kelas ke-i,
- c = jumlah kelas.

Semakin kecil nilai *Gini Index*, maka node tersebut semakin homogen sehingga menghasilkan pemisahan data yang lebih baik. Algoritma akan memilih atribut yang menghasilkan nilai *Gini Index* paling kecil atau penurunan impuritas terbesar sebagai dasar pembentukan node berikutnya [15].

Selain *Gini Index*, *Entropy* juga banyak digunakan sebagai kriteria dalam proses *splitting*. *Entropy* mengukur tingkat ketidakpastian (*uncertainty*) atau ketidakteraturan data pada suatu node berdasarkan konsep teori informasi. Semakin kecil nilai *Entropy*, maka data pada node tersebut semakin homogen. Persamaan *Entropy* ditunjukkan pada Persamaan 2.3.

$$Entropy(D) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{2.3}$$

- D = himpunan data pada suatu node,
- p_i = proporsi data pada kelas ke-i,
- c = jumlah kelas.

Berdasarkan nilai *Entropy*, dihitung nilai *Information Gain* untuk menentukan atribut yang menghasilkan pemisahan terbaik. Persamaan *Information Gain* ditunjukkan pada Persamaan 2.4.

$$\text{Information Gain}(D,A) = \text{Entropy}(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} \text{Entropy}(D_v) \tag{2.4}$$

- D = himpunan data sebelum dilakukan pemisahan,
- A = atribut yang dievaluasi,

- D_v = himpunan data hasil pemisahan berdasarkan nilai atribut A,
- $|D_v|$ = jumlah data pada subset D_v,
- $|D|$ = jumlah seluruh data pada node.

Atribut dengan nilai *Information Gain* tertinggi dipilih sebagai dasar pembentukan cabang berikutnya karena mampu menghasilkan penurunan tingkat ketidakpastian yang paling besar. Pada penelitian ini, algoritma *Decision Tree* menggunakan kriteria *Entropy* sehingga proses pemilihan atribut didasarkan pada nilai *Information Gain* yang dihasilkan pada setiap proses *splitting* [15].

Salah satu keunggulan *Decision Tree* adalah kemampuannya dalam menangani data numerik maupun kategorikal tanpa memerlukan asumsi mengenai distribusi data. Selain itu, struktur pohon yang dihasilkan mudah dipahami sehingga proses pengambilan keputusan dapat diinterpretasikan dengan baik. Namun demikian, *Decision Tree* memiliki kecenderungan mengalami *overfitting* apabila pohon yang terbentuk terlalu kompleks. Oleh karena itu, pengaturan parameter seperti *max_depth* dan *min_samples_split* diperlukan untuk mengendalikan kompleksitas model agar mampu melakukan generalisasi dengan lebih baik [14].

2.3.2 Random Forest

Random Forest merupakan algoritma *supervised learning* yang dikembangkan dari metode *Classification and Regression Tree* (CART) dengan menerapkan konsep *ensemble learning*, yaitu menggabungkan hasil prediksi dari beberapa *Decision Tree* untuk menghasilkan model klasifikasi yang lebih baik [16]. Berbeda dengan *Decision Tree* yang hanya membangun satu pohon keputusan, *Random Forest* membentuk banyak *Decision Tree* secara paralel menggunakan data latih dan subset fitur yang dipilih secara acak. Pendekatan tersebut memungkinkan setiap pohon mempelajari karakteristik data yang berbeda sehingga menghasilkan model yang lebih stabil dan memiliki kemampuan generalisasi yang lebih baik dibandingkan penggunaan satu *Decision Tree* [17].

Proses pembentukan *Random Forest* diawali dengan menerapkan teknik *Bootstrap Aggregating* (*Bagging*), yaitu membangun beberapa *Decision Tree* menggunakan sampel data hasil *bootstrap sampling*. Pada proses ini, data latih dipilih secara acak menggunakan metode *sampling with replacement*, sehingga suatu data dapat terpilih lebih dari satu kali, sedangkan sebagian data lainnya

mungkin tidak terpilih pada suatu sampel [16]. Setiap sampel kemudian digunakan untuk melatih satu *Decision Tree* secara independen. Penerapan teknik *Bagging* menghasilkan kumpulan *Decision Tree* yang beragam sehingga mampu mengurangi variansi model dan meningkatkan kestabilan hasil prediksi dibandingkan penggunaan satu *Decision Tree* [17]. Prediksi dari setiap *Decision Tree* tersebut tidak langsung digunakan sebagai hasil akhir, melainkan akan digabungkan pada tahap akhir menggunakan mekanisme *ensemble*.

Selain menggunakan teknik *Bagging*, *Random Forest* juga menerapkan proses pemilihan subset fitur secara acak pada setiap proses pemisahan node (*split*) [16]. Algoritma tidak mengevaluasi seluruh fitur yang tersedia, melainkan hanya mempertimbangkan sebagian fitur yang dipilih secara acak, kemudian memilih atribut terbaik dari subset tersebut sebagai dasar pembentukan node berikutnya. Dengan menggunakan subset fitur yang berbeda pada setiap *Decision Tree*, struktur pohon yang dihasilkan menjadi lebih beragam meskipun dilatih menggunakan dataset yang sama. Keragaman tersebut mengurangi korelasi antar *Decision Tree* sehingga meningkatkan efektivitas proses *ensemble* serta membantu mengurangi risiko *overfitting* [17].

Setelah seluruh *Decision Tree* selesai dibangun, masing-masing pohon memberikan prediksi terhadap data masukan. Seluruh hasil prediksi kemudian digabungkan menggunakan mekanisme *majority voting*, yaitu kelas yang memperoleh jumlah suara terbanyak akan dipilih sebagai hasil klasifikasi akhir [16]. Melalui mekanisme tersebut, kesalahan prediksi yang dihasilkan oleh sebagian *Decision Tree* dapat dikompensasi oleh prediksi dari pohon lainnya sehingga menghasilkan keputusan yang lebih konsisten dibandingkan hanya mengandalkan satu *Decision Tree*.

Berdasarkan karakteristik tersebut, *Random Forest* banyak digunakan dalam berbagai permasalahan klasifikasi karena mampu menghasilkan prediksi yang akurat, memiliki kemampuan generalisasi yang baik, serta efektif dalam menangani hubungan non-linear antar fitur. Selain itu, algoritma ini lebih tahan terhadap *overfitting* dibandingkan metode yang hanya menggunakan satu *Decision Tree*. Meskipun demikian, proses pelatihan *Random Forest* memerlukan sumber daya komputasi yang lebih besar karena harus membangun banyak *Decision Tree* sebelum menghasilkan prediksi akhir [16].

2.3.3 Logistic Regression

Logistic Regression merupakan salah satu algoritma *supervised learning* yang digunakan untuk menyelesaikan permasalahan klasifikasi. Algoritma ini memprediksi probabilitas suatu data termasuk ke dalam kelas tertentu berdasarkan hubungan antara variabel masukan dan variabel keluaran. *Logistic Regression* banyak digunakan pada berbagai bidang karena memiliki struktur model yang sederhana, mudah diinterpretasikan, serta mampu memberikan hasil prediksi dalam bentuk probabilitas [18, 19]. Dalam penelitian ini, *Logistic Regression* digunakan sebagai salah satu algoritma klasifikasi untuk membandingkan performanya dengan algoritma lain berdasarkan metrik evaluasi yang digunakan.

Pada *Logistic Regression*, hubungan antara variabel masukan dengan probabilitas kelas dimodelkan menggunakan fungsi logistik atau fungsi *sigmoid*. Fungsi ini mengubah kombinasi linear dari seluruh fitur menjadi nilai probabilitas pada rentang 0 hingga 1. Selama proses pelatihan, algoritma akan mengestimasi parameter model sehingga probabilitas yang dihasilkan mendekati label sebenarnya pada data pelatihan [18]. Melalui fungsi tersebut, model dapat merepresentasikan hubungan antara variabel masukan dan peluang suatu data untuk termasuk ke dalam kelas tertentu secara matematis.

$$P(Y = 1|X) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{2.6}$$

Pada rumus 2.5, nilai $P(Y|X)$ menunjukkan probabilitas suatu data termasuk ke dalam suatu kelas berdasarkan fitur yang dimiliki. Sementara itu, z merupakan kombinasi linear dari seluruh variabel masukan, di mana β_0 adalah konstanta, $\beta_1, \beta_2, \ldots, \beta_n$ merupakan koefisien model, sedangkan $X_1, X_2, \ldots, X_n$ merupakan variabel independen. Nilai koefisien tersebut diperoleh melalui proses pelatihan sehingga model dapat menghasilkan prediksi yang optimal [18]. Probabilitas yang dihasilkan kemudian dibandingkan dengan suatu nilai ambang (*threshold*) untuk menentukan kelas akhir, sehingga setiap data dapat diklasifikasikan ke dalam salah satu kelas yang tersedia.

2.3.4 Hyperparameter Tuning

Hyperparameter merupakan parameter yang ditentukan sebelum proses pelatihan model dimulai dan berperan dalam mengatur proses pembentukan model. Berbeda dengan parameter model yang dipelajari secara otomatis selama proses pelatihan, nilai *hyperparameter* harus ditentukan terlebih dahulu oleh pengguna. *Hyperparameter* juga bergantung pada algoritma yang digunakan karena algoritma yang berbeda juga dapat memiliki *hyperparameter* yang berbeda dan unik untuk masing-masing. Pemilihan nilai *hyperparameter* yang kurang sesuai dapat menyebabkan performa model tidak optimal, sehingga diperlukan proses *hyperparameter tuning* untuk memperoleh konfigurasi yang memberikan hasil terbaik [20].

Salah satu metode yang umum digunakan untuk melakukan *hyperparameter tuning* adalah *GridSearchCV*. Metode ini bekerja dengan menguji seluruh kombinasi *hyperparameter* yang telah ditentukan sebelumnya. Setiap kombinasi kemudian dievaluasi menggunakan *cross-validation* sehingga performa masing-masing konfigurasi dapat dibandingkan secara objektif. Kombinasi yang menghasilkan performa terbaik selanjutnya dipilih sebagai konfigurasi *hyperparameter* optimal untuk digunakan dalam proses pelatihan model [20].

Pada penelitian ini, *GridSearchCV* digunakan untuk memperoleh kombinasi *hyperparameter* terbaik pada algoritma *Logistic Regression*, *Decision Tree*, dan *Random Forest*. Dengan menggunakan proses pencarian secara sistematis dan evaluasi melalui *cross-validation*, setiap algoritma dilatih menggunakan konfigurasi hyperparameter yang telah dioptimalkan sehingga hasil perbandingan performa antar model menjadi lebih objektif. Selain itu, *GridSearchCV* dilakukan pada model secara individual atau masing-masing dan tidak bersamaan untuk menghindari terjadinya konflik antar *hyperparameter*.

2.4 Evaluation Metrics

Evaluasi model klasifikasi dilakukan untuk mengukur kemampuan model dalam menghasilkan prediksi yang tepat terhadap data uji. Pada penelitian ini, performa model dievaluasi menggunakan empat metrik, yaitu *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Keempat metrik tersebut memberikan perspektif yang berbeda dalam menilai kualitas hasil klasifikasi sehingga dapat digunakan untuk membandingkan performa antar algoritma secara lebih komprehensif [21].

Accuracy mengukur proporsi keseluruhan prediksi yang berhasil diklasifikasikan dengan benar terhadap jumlah seluruh data. Metrik ini memberikan gambaran umum mengenai performa model klasifikasi [21].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Precision mengukur proporsi data yang diprediksi sebagai positif dan benar-benar termasuk ke dalam kelas positif. Nilai *Precision* yang tinggi menunjukkan bahwa model mampu meminimalkan kesalahan prediksi positif (*False Positive*) [21].

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

Recall mengukur kemampuan model dalam mengidentifikasi seluruh data yang benar-benar termasuk ke dalam kelas positif. Nilai *Recall* yang tinggi menunjukkan bahwa model mampu mendeteksi sebagian besar data positif yang tersedia [21].

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

F1-Score merupakan rata-rata harmonik antara *Precision* dan *Recall*. Metrik ini digunakan untuk memberikan keseimbangan antara kemampuan model dalam menghasilkan prediksi positif yang benar dan kemampuannya dalam mendeteksi seluruh data positif [21].

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.10}$$

2.5 One-Way ANOVA

Analysis of Variance (ANOVA) merupakan metode statistik yang digunakan untuk membandingkan rata-rata dari tiga kelompok atau lebih guna mengetahui apakah terdapat perbedaan yang signifikan di antara kelompok tersebut [22]. ANOVA merupakan pengembangan dari *t-test*, sehingga memungkinkan pengujian perbedaan rata-rata beberapa kelompok secara simultan tanpa harus melakukan pengujian secara berulang.

Salah satu jenis ANOVA yang umum digunakan adalah *One-Way ANOVA*,

yaitu ANOVA yang hanya memiliki satu variabel independen atau faktor. Metode ini digunakan untuk menguji apakah terdapat perbedaan rata-rata antar kelompok berdasarkan satu faktor tertentu [22].

Pada penelitian ini, *One-Way ANOVA* digunakan untuk menganalisis apakah terdapat perbedaan yang signifikan pada nilai rata-rata fitur-fitur terpilih di antara tiga kategori *player engagement*, yaitu *Low*, *Medium*, dan *High*. Hasil pengujian ini digunakan untuk mendukung analisis karakteristik masing-masing tingkat *engagement* pemain.

2.6 Penelitian Terdahulu

Sebelum melakukan penelitian ini, beberapa penelitian terdahulu dilakukan terkait topik *engagement* pemain dan juga *machine learning*. Penelitian-penelitian tersebut dapat dilihat pada tabel 2.2.

Tabel 2.2. Penelitian Terdahulu

No	Peneliti	Judul	Metode	Temuan
1	Theodorakopoulos & Theodoropoulou (2024)	Consumer Behavior Analytics	Big Data Analytics	Analitik data efektif untuk memahami perilaku konsumen dan *engagement*.
2	Romero-Mendez et al. (2023)	Deep Learning for Player *Engagement*	Deep Learning	Penyesuaian kesulitan meningkatkan *engagement* pemain.
3	Peters et al. (2024)	Context-aware User *Engagement* Prediction	Context-aware Machine Learning	Model meningkatkan akurasi prediksi *engagement*.

Bersambung ke halaman berikutnya

Tabel 2.2 – lanjutan

No	Peneliti	Judul	Metode	Temuan
4	Rismayanti (2024)	Online Gaming Behaviour Prediction	Machine learning Classification	*Machine learning* efektif memprediksi perilaku pemain.

Berdasarkan Tabel 2.2, dapat dilihat bahwa penelitian mengenai *engagement* telah dilakukan dengan berbagai pendekatan. Penelitian oleh Theodorakopoulos & Theodoropoulou (2024) menunjukkan bahwa analitik data dapat dimanfaatkan untuk memahami perilaku pengguna dan *engagement*. Sementara itu, Romero-Mendez et al. (2023) menerapkan pendekatan *deep learning* untuk meningkatkan *engagement* pemain melalui penyesuaian tingkat kesulitan permainan. Penelitian Peters et al. (2024) juga menunjukkan bahwa model *machine learning* berbasis konteks mampu meningkatkan akurasi prediksi *engagement*, sedangkan Rismayanti (2024) membuktikan bahwa *machine learning* efektif digunakan untuk memprediksi perilaku pemain *game online*.

Secara umum, penelitian-penelitian tersebut menunjukkan bahwa *machine learning* memiliki potensi yang baik dalam memahami perilaku pemain maupun memprediksi tingkat *engagement*. Namun, penelitian yang secara khusus membandingkan beberapa algoritma klasifikasi sekaligus menganalisis *feature importance* dan karakteristik perilaku pemain masih belum banyak ditemukan. Oleh karena itu, penelitian ini dilakukan untuk membandingkan performa beberapa algoritma klasifikasi serta mengidentifikasi faktor-faktor yang paling berpengaruh terhadap tingkat *engagement* pemain.