

## **BAB II**

### **KAJIAN LITERATUR**

#### **2.1 Literasi Keuangan dan Pengelolaan Keuangan Pribadi**

Literasi keuangan adalah kemampuan seseorang dalam memahami dan menerapkan konsep keuangan dalam kehidupan sehari-hari, seperti membuat anggaran, mengelola utang, dan merencanakan tabungan [1]. Kemampuan ini penting karena secara langsung mempengaruhi kualitas keputusan finansial yang diambil seseorang.

Di Indonesia, tingkat literasi keuangan masih menjadi perhatian. Berdasarkan SNLIK 2022 yang dilakukan oleh OJK, indeks literasi keuangan nasional tercatat sebesar 49,68%, naik dari 38,03% pada tahun 2019 [1]. Meski ada peningkatan, angka ini masih menunjukkan bahwa lebih dari separuh masyarakat Indonesia belum memiliki pemahaman keuangan yang memadai. Kondisi ini berdampak pada rendahnya kemampuan masyarakat dalam merencanakan dan mengontrol pengeluaran bulanan mereka.

Pengelolaan keuangan pribadi mencakup beberapa aspek utama, di antaranya penganggaran, pengelolaan arus kas, dan perencanaan jangka panjang [1]. Dalam penelitian ini, aspek penganggaran dan pengelolaan pengeluaran menjadi fokus utama, karena aplikasi Smart-Budgeting dirancang untuk membantu pengguna dalam merencanakan dan memantau pengeluaran bulanan mereka secara lebih akurat.

Pengeluaran bulanan pengguna dipengaruhi oleh berbagai faktor seperti pendapatan, status kepemilikan pinjaman, tingkat pendidikan, dan wilayah tempat tinggal. Faktor-faktor ini selaras dengan kolom-kolom yang tersedia dalam *Personal Finance ML Dataset* yang digunakan dalam penelitian ini, sehingga dataset tersebut relevan untuk memodelkan perilaku pengeluaran nyata.

#### **2.2 Aplikasi Mobile Pengelolaan Keuangan Personal**

Perkembangan teknologi *mobile* mendorong munculnya berbagai aplikasi pengelolaan keuangan personal (*Personal Finance Management / PFM*) yang semakin canggih. Aplikasi jenis ini tidak lagi sekadar berfungsi sebagai catatan transaksi, melainkan sudah dilengkapi fitur seperti visualisasi tren pengeluaran, notifikasi anggaran, dan rekomendasi berbasis data .

Salah satu fitur yang semakin banyak dikembangkan adalah prediksi pengeluaran — yaitu kemampuan sistem untuk memperkirakan berapa besar pengeluaran pengguna di bulan

berikutnya berdasarkan data historis dan profil finansial mereka. Fitur inilah yang menjadi fokus utama penelitian ini dalam konteks aplikasi *Smart-Budgeting*.

Aplikasi *Smart-Budgeting* dirancang dengan arsitektur *client-server*, di mana aplikasi mobile berfungsi sebagai antarmuka pengguna yang berkomunikasi dengan *backend* melalui *RESTful API*. *Backend* bertugas menyimpan data pengguna dan menjalankan model machine learning untuk menghasilkan prediksi. Data profil yang tersimpan dalam sistem — seperti usia, pendapatan, status pinjaman, dan wilayah — secara langsung berhubungan dengan fitur-fitur dalam *Personal Finance ML Dataset* [9] yang digunakan dalam penelitian ini.

Akurasi prediksi dalam konteks ini sangat penting. Jika prediksi terlalu rendah, pengguna bisa mengalami defisit anggaran. Sebaliknya, jika terlalu tinggi, pengguna mungkin tidak terdorong untuk berhemat. Oleh karena itu, meminimalkan *MAE* dan *MAPE* secara bersamaan menjadi tujuan yang paling relevan untuk aplikasi ini.

### 2.3 *Machine Learning* untuk Prediksi Tabular

*Machine learning* adalah pendekatan dalam kecerdasan buatan di mana sistem belajar dari data untuk membuat prediksi atau keputusan, tanpa harus diprogram secara eksplisit untuk setiap kasus [11]. Dalam penelitian ini, *machine learning* digunakan untuk memprediksi pengeluaran bulanan pengguna berdasarkan profil demografis dan finansial mereka — yang merupakan masalah *supervised learning* dengan jenis tugas regresi. Model belajar memetakan fitur *input* (profil pengguna) ke nilai *output* (pengeluaran bulanan) dari data pelatihan yang sudah diketahui hasilnya.

Dataset yang digunakan berbentuk tabular — terstruktur dalam baris dan kolom seperti *spreadsheet* — yang merupakan format data paling umum di domain keuangan. Data tabular memiliki karakteristik tersendiri dibanding data citra atau teks, yaitu campuran antara fitur numerik dan kategoris, serta potensi hubungan *non-linear* antar fitur [3]. Karakteristik ini mempengaruhi pemilihan algoritma yang tepat.

Secara umum, algoritma yang digunakan untuk data tabular dapat dibagi menjadi beberapa kelompok. Algoritma linier seperti Regresi Linear sederhana dan cepat, namun tidak mampu menangkap hubungan *non-linear*. Algoritma berbasis pohon seperti *Random Forest* lebih fleksibel [3]. Metode *gradient boosting* seperti *XGBoost* saat ini menjadi pilihan utama karena akurasinya yang tinggi pada data tabular [7]. Sementara itu, jaringan saraf seperti KAN [4] mulai menunjukkan potensi sebagai alternatif dengan interpretabilitas

yang lebih baik, meski evaluasinya pada data keuangan masih terbatas. Dua algoritma terakhir inilah yang menjadi fokus perbandingan dalam penelitian ini.

## 2.4 *Extreme Gradient Boosting (XGBoost)*

### 2.4.1 *Gradient Boosting sebagai Fondasi*

*Gradient Boosting* [8] merupakan teknik *machine learning ensemble* yang membangun prediksi sebagai penjumlahan aditif dari sejumlah model basis (*base learners*), umumnya pohon keputusan *CART* (*Classification and Regression Trees*). Proses pembangunannya bersifat sekuensial dan iteratif: setiap model baru yang ditambahkan bertujuan mengoreksi residual *error* dari *ensemble* sebelumnya. Secara matematis, prediksi model pada iterasi ke- $t$  adalah:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \dots\dots\dots(2.1)$$

di mana  $f_t$  adalah pohon keputusan ke- $t$ . *Gradient Boosting* beroperasi dengan memandang masalah pembangunan ensemble sebagai optimasi numerik dalam ruang fungsi: setiap  $f_t$  dipilih sebagai fungsi yang paling mengurangi fungsi *loss*  $L$  secara keseluruhan, di mana arah penurunan ditentukan oleh *pseudo-residual* (*negatif gradien loss*):

$$r_i^{(t)} = - \frac{\partial l(y_i, \hat{y})}{\partial \hat{y}} \Big|_{\hat{y} = \hat{y}_i^{(t-1)}} \dots\dots\dots(2.2)$$

### 2.4.2 *Fungsi Objektif dan Regularisasi XGBoost*

*XGBoost* [7] memperluas gradient boosting konvensional dengan mengoptimalkan fungsi objektif regularisasi yang mencakup *term* penalti kompleksitas model:

$$\mathcal{L}^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t)}) + \Omega(f_t) \dots\dots\dots(2.3)$$

Term regularisasi  $\Omega(f)$  didefinisikan sebagai:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2 + \alpha \sum_j w_j \dots\dots\dots(2.4)$$

di mana  $T$  adalah jumlah daun,  $w$  adalah vektor skor daun,  $\gamma$  adalah penalti jumlah daun (minimum L1 pada *gain*),  $\lambda$  adalah koefisien regularisasi L2 pada skor daun, dan  $\alpha$  adalah koefisien regularisasi L1 (*Lasso*) pada skor daun. Perluasan *Taylor* orde kedua dari fungsi *loss* memungkinkan *XGBoost* menentukan *split* terbaik secara analitik:

$$Score_{split} = \frac{1}{2} + \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}, - \gamma \dots \dots \dots (2.5)$$

di mana  $G_L = \sum_{i \in I_L} g_i$  dan  $H_L = \sum_{i \in I_L} h_i$  adalah jumlah gradien dan Hessian pada partisi kiri, dan analog untuk partisi kanan. Penggunaan informasi Hessian (orde kedua) ini memberikan *XGBoost* konvergensi yang lebih cepat dan stabil dibandingkan *gradient boosting* konvensional yang hanya menggunakan gradien orde pertama.

### 2.4.3 Inovasi Teknis *XGBoost*

Selain fungsi objektif yang kuat, *XGBoost* memperkenalkan sejumlah inovasi teknis yang menjadikannya pilihan dominan untuk data tabular:

1. *Approximate Greedy Algorithm*: Menggunakan *weighted quantile sketch* untuk menemukan *candidate split points* secara efisien tanpa harus mengevaluasi semua nilai unik dari setiap fitur, memungkinkan skalabilitas pada dataset besar.
2. *Sparsity-Aware Split Finding*: Menambahkan *default direction* pada setiap *node* untuk menangani *missing values* secara otomatis—data dengan nilai hilang diarahkan ke *branch* yang menghasilkan *gain* terbaik selama pelatihan.
3. *Column Block* untuk Komputasi Paralel: Menyimpan data dalam blok kolom yang telah diurutkan, memungkinkan evaluasi *split* pada berbagai fitur secara paralel.
4. *Cache-Aware Access*: Mengoptimalkan akses memori dengan mengalokasikan data *gradient* ke *buffer internal*, mengurangi *cache miss* pada perangkat keras modern.
5. *Subsampling* Ganda: Mendukung *row subsampling* (*subsample*), *column subsampling* per pohon (*colsample\_bytree*), dan per level pohon (*colsample\_bylevel*)—memberikan randomisasi tambahan yang mengurangi korelasi antar pohon dalam *ensemble*.

Kombinasi dari inovasi-inovasi teknis di atas menjadikan *XGBoost* sebagai algoritma yang tidak hanya akurat, tetapi juga efisien secara komputasi. Dalam penelitian ini, *XGBoost* digunakan sebagai representasi metode *ensemble* berbasis gradient boosting yang akan dibandingkan dengan *KAN* pada dataset keuangan personal.

## 2.5 Kolmogorov-Arnold Networks (KAN)

### 2.5.1 Landasan Teoritis: Teorema Kolmogorov-Arnold

Teorema superposisi *Kolmogorov-Arnold* [5], yang pertama kali dikemukakan oleh *Kolmogorov* pada tahun 1957 dan disempurnakan oleh *Arnold*, menyatakan bahwa setiap fungsi kontinu multivariat  $f$  yang terdefinisi pada hiperkubus  $[0,1]^n$  dapat dinyatakan secara eksak sebagai superposisi terbatas dari fungsi kontinu univariat. Secara formal:

$$f(x_1, x_2, \dots, x_n) = \sum_{q=0}^{2^n} \Phi_q \left( \sum_{p=1}^n \varphi_{q,p}(x_p) \right) \dots\dots\dots (2.6)$$

di mana  $\varphi_{q,p}: [0,1] \rightarrow \mathbb{R}$  adalah fungsi kontinu *univariat inner* (*inner functions*) dan  $\Phi_q: \mathbb{R} \rightarrow \mathbb{R}$  adalah fungsi kontinu *univariat outer* (*outer functions*). Teorema ini memiliki implikasi fundamental: setiap kompleksitas multivariat yang tampaknya memerlukan representasi dimensi tinggi dapat pada prinsipnya didekomposisi menjadi superposisi fungsi-fungsi satu variabel yang jauh lebih sederhana. Dengan kata lain, kelas fungsi "multivariat" pada dasarnya tidak ada—semua fungsi kontinu multivariat dapat direduksi menjadi komposisi fungsi univariat.

Namun, *Kolmogorov* dalam formulasi aslinya tidak mensyaratkan bahwa  $\varphi_{q,p}$  dan  $\Phi_q$  harus *smooth* atau mudah dipelajari—mereka dapat menjadi fungsi yang sangat tidak teratur (*irregular*). Kontribusi utama *Liu et al.* [4] dalam *KAN* adalah memodelkan fungsi-fungsi ini menggunakan *B-splines* yang *smooth* dan dapat dioptimalkan melalui *backpropagation*, sehingga menjadikan teorema *Kolmogorov-Arnold* praktis sebagai fondasi arsitektur jaringan saraf.

### 2.5.2 Arsitektur dan Komponen KAN

*KAN* dibangun dari unit-unit lapisan yang disebut *KAN layer*. Sebuah *KAN layer* dengan dimensi *input*  $n_{in}$  dan dimensi *output*  $n_{out}$  didefinisikan sebagai matriks fungsi aktivasi yang dapat dipelajari:

$$\Phi = @ \varphi_{p,q} B, \quad p = 1, \dots, n_{in}, \quad q = 1, \dots, n_{out} \dots\dots\dots (7)$$

di mana setiap  $\varphi_{\{p,q\}}$  adalah sebuah fungsi univariat yang dapat dipelajari (*learnable univariate function*) yang ditempatkan pada *edge* dari *node*  $p$  di lapisan *input* ke *node*  $q$  di lapisan *output*. *Output node*  $q$  dihitung sebagai:

$$x_{out,q} = \sum_{p=1}^{n_{in}} \varphi_{p,q} [x_{in,p}], \quad q = 1, \dots, n_{out} \dots\dots\dots (2.8)$$

Setiap fungsi aktivasi  $\varphi_{\{p,q\}}$  diimplementasikan sebagai kombinasi dari dua komponen:

$$\varphi(x) = w_b \cdot b(x) + w_s \cdot spline(x) \dots\dots\dots (2.9)$$

di mana  $b(x) = \text{SiLU}(x) = x/(1+e^{-x})$  adalah fungsi basis (*residual connection*),  $spline(x)$  adalah fungsi *B-spline* kubik, sedangkan  $w_b$  dan  $w_s$  adalah bobot skalar yang dapat dipelajari. Fungsi *B-spline* diparameterisasi sebagai:

$$spline(x) = \sum_i c_i \cdot B_i(x) \dots\dots\dots (2.10)$$

dengan  $B_i(x)$  adalah *B-spline* basis function ke- $i$  dari order  $k$  dengan  $G+1$  titik *grid* yang seragam pada *interval input*, dan  $c_i$  adalah koefisien *spline* yang dapat dilatih. Jumlah total parameter yang dapat dipelajari pada satu *edge* adalah  $k+G+2$  ( $k+G$  koefisien *spline* ditambah  $w_b$  dan  $w_s$ ).

Sebuah jaringan *KAN* dengan  $L$  lapisan dinotasikan sebagai  $KAN[n_0, n_1, \dots, n_L]$ , di mana  $n_\ell$  adalah dimensi lapisan ke- $\ell$ . Kontras dengan  $MLP[n_0, n_1, \dots, n_L]$  di mana setiap lapisan memiliki  $n_{\ell-1} \times n_\ell$  parameter bobot ditambah  $n_\ell$  bias, sebuah lapisan *KAN* memiliki  $n_{\ell-1} \times n_\ell$  fungsi aktivasi masing-masing dengan  $k+G+2$  parameter. *KAN* berpotensi memerlukan parameter total yang lebih sedikit daripada *MLP* yang setara ketika *order spline*  $k$  dan *grid size*  $G$  kecil.

### 2.5.3 Pelatihan dan Optimasi *KAN*

Pelatihan *KAN* menggunakan algoritma *backpropagation* standar untuk mengoptimalkan semua parameter  $\{w_b, w_s, c_i\}$  pada setiap *edge* secara simultan. Fungsi *loss* yang dioptimalkan adalah *MSE* untuk tugas regresi, dengan tambahan regularisasi *entropy* aktivasi untuk mendorong *sparsity*:

$$\mathcal{L}_{total} = MSE(y, \hat{y}) + \lambda_{ent} \cdot l_{ent}(\Phi) \dots\dots\dots (2.11)$$

di mana  $l_{ent}(\Phi) = \sum_{p,q} \|\phi_{p,q}\|_1 / \|\phi_{p,q}\|_\infty$  mengukur total entropi aktivasi seluruh *edge* dan  $\lambda_{ent}$  adalah hyperparameter regularisasi. Regularisasi ini secara alami mendorong jaringan untuk menggunakan hanya *subset* fitur yang paling informatif, menghasilkan model yang lebih *sparse* dan lebih interpretatif.

*KAN* mendukung mekanisme *grid refinement* yang unik: setelah pelatihan awal dengan *grid* kasar (*G* kecil), resolusi *grid* dapat ditingkatkan secara bertahap (dengan menggandakan *G*) sambil menginisialisasi koefisien *spline* baru berdasarkan aproksimasi dari model sebelumnya. Mekanisme ini memungkinkan pelatihan yang progresif dari representasi kasar ke halus, menghasilkan konvergensi yang lebih stabil dan akurasi yang lebih baik.

#### 2.5.4 *KAN* 0.2 dan Perkembangan Terbaru

*Liu et al.* [6] memperkenalkan *KAN* 2.0 sebagai generasi lanjutan dari arsitektur *KAN* orisinal. *KAN* 2.0 memperkenalkan beberapa inovasi penting yang meningkatkan kemampuan dan fleksibilitas arsitektur:

1. *MultKAN*: Menambahkan operasi perkalian sebagai alternatif penjumlahan dalam konstruksi jaringan, memungkinkan representasi fungsi yang mengandung perkalian variabel (seperti interaksi fitur) secara lebih alami dan efisien.
2. *KAN* pada *Manifold*: Memungkinkan penerapan *KAN* pada ruang manifold yang didefinisikan oleh pengguna, membuka aplikasi *KAN* pada domain geometri dan fisika komputasional.
3. Peningkatan Antarmuka Simbolik: Fitur untuk mengekstrak dan menyederhanakan ekspresi matematis simbolik yang mendasari data diperkuat, meningkatkan kemampuan penemuan ilmiah (*scientific discovery*) dari *KAN*.

Perkembangan ini menunjukkan bahwa *KAN* bukan sekadar arsitektur eksperimental, melainkan platform yang terus berkembang dengan basis teori yang kuat. Meskipun demikian, sebagian besar evaluasi *KAN* masih berfokus pada dataset sains dan matematika. Penelitian ini memberikan kontribusi dengan mengevaluasi *KAN* (dalam implementasi *pykan* standar) pada dataset keuangan tabular yang berbeda karakteristiknya.

Berbeda dengan *MLP* yang menempatkan fungsi aktivasi tetap pada setiap node, *KAN* menempatkan fungsi aktivasi yang dapat dipelajari pada setiap *edge*-nya. Pendekatan ini memungkinkan model untuk lebih fleksibel dalam menangkap hubungan *non-linear*, sekaligus lebih mudah diinterpretasikan karena setiap fungsi pada *edge* dapat divisualisasikan.

Sebagian besar penelitian *KAN* sejauh ini berfokus pada dataset sains dan matematika [4][6]. Evaluasi *KAN* pada data keuangan tabular seperti yang digunakan dalam penelitian ini masih jarang dilakukan. Oleh karena itu, penelitian ini memberikan kontribusi dengan menguji sejauh mana kemampuan *KAN* dalam memprediksi pengeluaran bulanan berdasarkan profil demografis-finansial, dan membandingkan hasilnya secara langsung dengan *XGBoost*.

## 2.6 Teknik Preprocessing dan Feature Engineering

*Preprocessing* data dan *feature engineering* merupakan tahapan kritis yang secara langsung menentukan kualitas model machine learning yang dihasilkan, terlepas dari algoritma yang dipilih [3]. Bagian ini membahas teknik-teknik utama yang relevan untuk dataset *Personal Finance ML*.

### 2.6.1 Penanganan Missing Values

*Missing values* (nilai yang hilang) adalah situasi umum dalam data dunia nyata. Pendekatan penanganan yang dipilih harus mempertimbangkan mekanisme terjadinya *missing values*: *Missing Completely at Random (MCAR)*, *Missing at Random (MAR)*, atau *Missing Not at Random (MNAR)* [3]. Untuk data keuangan personal, *missing values* pada fitur numerik umumnya ditangani dengan imputasi *median* (*robust* terhadap *outlier*) atau *mean*, sedangkan fitur kategoris dengan modus. Untuk *XGBoost*, nilai hilang dapat dibiarkan sebagaimana adanya karena *XGBoost* memiliki mekanisme *sparsity awareness built-in* [7].

### 2.6.2 Penanganan dan Deteksi Outlier

*Outlier* adalah observasi yang menyimpang secara signifikan dari mayoritas data. Pada data keuangan personal, outlier sering terjadi pada fitur seperti pendapatan, tabungan, dan rasio utang—mencerminkan keberagaman kondisi finansial pengguna yang ekstrem. Penanganan *outlier* yang umum digunakan adalah *Winsorization* pada batas persentil, *Modified Z-Score*, dan *IQR-based clipping*. *Winsorization* pada persentil ke-1 dan ke-99

dipilih dalam penelitian ini karena mempertahankan ukuran sampel (tidak menghapus *record*) sembari mengurangi pengaruh nilai ekstrem.

### 2.6.3 Encoding Fitur Kategoris

Fitur kategoris harus dikonversi ke representasi numerik sebelum dapat digunakan oleh sebagian besar algoritma *ML*. Dalam penelitian ini, dua metode *encoding* yang digunakan adalah *One-Hot Encoding* untuk fitur nominal dan *Ordinal Encoding* untuk fitur *ordinal* (*education\_level*). Pemilihan ini disesuaikan dengan karakteristik fitur kategoris yang ada dalam *Personal Finance ML Dataset*.

### 2.6.4 Normalisasi dan Standarisasi Fitur

Normalisasi fitur bertujuan untuk memastikan semua fitur berada pada skala yang sebanding, sehingga tidak ada fitur yang mendominasi proses pembelajaran hanya karena skala nilainya yang besar. *Min-Max Scaling* mentransformasi semua nilai ke rentang  $[0,1]$ :

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.12)$$

*Z-score Standardization* mentransformasi fitur sehingga memiliki mean nol dan standar deviasi satu:

$$x_{std} = \frac{x - \mu}{\sigma} \quad (2.13)$$

*Min-Max Scaling* dipilih dalam penelitian ini karena lebih cocok untuk *B-splines* pada *KAN* yang beroperasi pada rentang *input* yang tertentu. Perlu dicatat bahwa untuk *XGBoost*, normalisasi secara matematis tidak diperlukan (pohon keputusan bersifat *scale-invariant* terhadap transformasi monoton), namun tetap diterapkan untuk konsistensi *pipeline*.

## 2.7 Metrik Evaluasi Model Regresi

Evaluasi kinerja model regresi memerlukan penggunaan metrik kuantitatif yang mengukur seberapa dekat prediksi model dengan nilai aktual [12]. Penelitian ini menggunakan empat metrik evaluasi yang saling melengkapi

### 2.7.1 Mean Absolute Error (MAE)

*MAE* mengukur rata-rata nilai absolut kesalahan prediksi:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \dots\dots\dots (2.14)$$

*MAE* bersatuan sama dengan variabel target (*USD*), *robust* terhadap *outlier*, dan mudah diinterpretasikan. *MAE* digunakan sebagai metrik primer dalam penelitian ini.

Salah satu sifat penting *MAE* yang perlu dipahami dalam interpretasi hasil adalah bahwa nilainya bersatuan sama dengan variabel target yang diprediksi. Chai dan Draxler [21] menegaskan bahwa *MAE* dapat langsung dibandingkan dengan nilai aktual karena keduanya berada dalam satuan yang identik. Dengan demikian, apabila variabel target merupakan nilai finansial dalam satuan Dolar Amerika Serikat (*USD*), maka nilai *MAE* yang dihasilkan pun akan bersatuan *USD* — bukan merupakan nilai tak berdimensi yang harus mendekati nol secara absolut. Sebagai contoh, *MAE* sebesar 590 *USD* pada prediksi pengeluaran bulanan dengan rata-rata aktual 2.419 *USD* berarti model meleset rata-rata 24,4% dari nilai aktual, yang tercermin pula pada nilai *MAPE* sebesar 28,47%. Interpretasi *MAE* yang tepat harus selalu mengacu pada skala dan satuan variabel target, bukan pada nilai absolutnya secara terpisah..

### 2.7.2 Root Mean Square Error (RMSE)

*RMSE* mengukur akar rata-rata kuadrat kesalahan prediksi:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \dots\dots\dots (2.15)$$

Type equation here.

*RMSE* lebih sensitif terhadap kesalahan besar karena pengkuadratan, sehingga berguna untuk mengidentifikasi model yang cenderung membuat kesalahan prediksi ekstrem.

Seperti halnya *MAE*, *RMSE* juga memiliki satuan yang sama dengan variabel target. Chai dan Draxler (2014) dalam jurnal *Geoscientific Model Development* menegaskan bahwa *RMSE* merupakan metrik standar evaluasi model yang bersatuan sama dengan variabel yang diprediksi, sehingga dapat diinterpretasikan secara langsung dalam konteks domain masalah [21]. Dalam prediksi nilai finansial seperti pengeluaran bulanan dalam *USD*, nilai *RMSE*

yang mencapai ratusan atau ribuan bukan berarti model buruk — melainkan mencerminkan skala variabel target itu sendiri. Hal ini sejalan dengan pernyataan bahwa *RMSE* digunakan justru ketika dibutuhkan metrik yang selaras dengan satuan domain masalah (*units of the problem domain*), sehingga lebih mudah diinterpretasikan oleh pemangku kepentingan. Untuk menilai apakah nilai *MAE* dan *RMSE* tergolong baik atau buruk, diperlukan perbandingan relatif terhadap rata-rata variabel target — bukan penilaian nilai absolut semata..

### 2.7.3 Mean Absolute Percentage Error (MAPE)

*MAPE* mengukur rata-rata kesalahan relatif dalam persentase:

$$MAPE = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i} \dots\dots\dots (2.16)$$

*MAPE* bersifat scale-independent sehingga mudah dikomunikasikan dan dibandingkan lintas domain. Keterbatasannya adalah tidak terdefinisi ketika  $y_i = 0$  dan memberikan penalti asimetris pada *under-prediction*.

### 2.7.4 Koefisien Determinasi ( $R^2$ )

$R^2$  mengukur proporsi variansi variabel target yang dijelaskan oleh model:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \dots\dots\dots (2.17)$$

Nilai  $R^2$  berkisar antara  $-\infty$  hingga 1.  $R^2=1$  berarti *model* sempurna;  $R^2=0$  berarti *model* tidak lebih baik dari prediksi rata-rata;  $R^2<0$  berarti *model* lebih buruk dari prediksi rata-rata.  $R^2$  digunakan untuk menilai kualitas *fit model* secara keseluruhan.

Secara umum, interpretasi keempat metrik dilakukan berdasarkan arah nilai dan karakteristik masing-masing metrik. *MAE*, *RMSE*, dan *MAPE* memiliki interpretasi yang semakin baik apabila nilainya semakin kecil, sedangkan  $R^2$  memiliki interpretasi yang semakin baik apabila nilainya semakin mendekati 1. Namun, tidak terdapat batas universal untuk menentukan nilai *MAE* dan *RMSE* sebagai baik atau buruk karena keduanya bergantung pada skala variabel target. Oleh karena itu, penelitian ini membandingkan nilai metrik antar-model pada dataset dan test set yang sama.

Tabel 2.1 Kriteria Interpretasi Metrik Evaluasi

| Metrik | Arah Nilai yang Lebih Baik | Interpretasi                                                      |
|--------|----------------------------|-------------------------------------------------------------------|
| MAE    | Semakin Kecil              | Kesalahan absolut rata-rata semakin rendah                        |
| RMSE   | Semakin Kecil              | Kesalahan prediksi, terutama kesalahan besar, semakin rendah      |
| MAPE   | Semakin Kecil              | Persentase kesalahan prediksi semakin rendah                      |
| $R^2$  | Semakin Mendekati 1        | Proporsi variasi target yang dapat dijelaskan model semakin besar |

Dengan demikian, pemilihan model terbaik dilakukan dengan mempertimbangkan nilai metrik secara keseluruhan, dengan MAE sebagai metrik utama penelitian, serta mempertimbangkan hasil pengujian statistik pada perbandingan KAN dan XGBoost.

## 2.8 Uji Signifikansi Statistik: *Wilcoxon Signed-Rank Test*

Uji *Wilcoxon Signed-Rank Test* [13] adalah uji statistik non-parametrik yang digunakan untuk membandingkan dua sampel berpasangan ketika asumsi normalitas distribusi tidak dapat dipenuhi. Uji ini pertama kali dikemukakan oleh *Frank Wilcoxon* pada tahun 1945 dan secara luas digunakan dalam penelitian *machine learning* untuk membandingkan kinerja dua algoritma.

Dalam konteks penelitian ini, uji *Wilcoxon Signed-Rank Test* digunakan untuk menguji apakah terdapat perbedaan kinerja yang signifikan secara statistik antara algoritma KAN dan XGBoost berdasarkan hasil 10-Fold Cross Validation. Pengujian dilakukan terhadap nilai MAE yang diperoleh kedua model pada setiap fold, sehingga terdapat 10 pasangan nilai MAE yang bersifat berpasangan.

Prosedur pengujian dilakukan sebagai berikut:

1. Membagi data training dan validation menjadi 10 *fold*, sedangkan *test set* tidak digunakan dalam proses *cross-validation*.
2. Pada setiap *fold*, melakukan *preprocessing* secara independen berdasarkan data *training-fold*, meliputi perhitungan batas *Winsorization* dan *fitting Min-Max Scaler*.
3. Melatih model *KAN* dan *XGBoost* menggunakan data *training-fold*.
4. Menghitung nilai *MAE* masing-masing model pada *validation-fold*.
5. Membentuk 10 pasangan nilai *MAE KAN* dan *XGBoost* dari seluruh fold.
6. Menerapkan *Wilcoxon Signed-Rank Test* terhadap pasangan nilai *MAE* tersebut.

Hipotesis yang digunakan adalah:

1.  $H_0$ : Tidak terdapat perbedaan yang signifikan secara statistik antara kinerja *KAN* dan *XGBoost* berdasarkan *MAE* pada *10-Fold Cross Validation*.
2.  $H_1$ : Terdapat perbedaan yang signifikan secara statistik antara kinerja *KAN* dan *XGBoost* berdasarkan *MAE* pada *10-Fold Cross Validation*.

Tingkat signifikansi yang digunakan adalah  $\alpha = 0,05$ . Jika  $p\text{-value} < 0,05$ , maka  $H_0$  ditolak dan perbedaan kinerja dinyatakan signifikan secara statistik. Sebaliknya, jika  $p\text{-value} \geq 0,05$ , maka  $H_0$  gagal ditolak karena belum terdapat bukti yang cukup untuk menyatakan adanya perbedaan yang signifikan.

Penggunaan *Wilcoxon Signed-Rank Test* dipilih karena nilai *MAE* dari setiap fold membentuk pasangan pengamatan antara *KAN* dan *XGBoost*, sehingga pengujian dapat dilakukan secara berpasangan tanpa mensyaratkan distribusi normal.

## 2.9 Flowchart

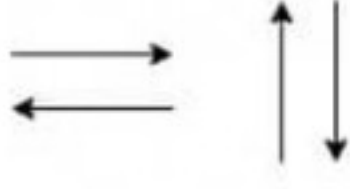
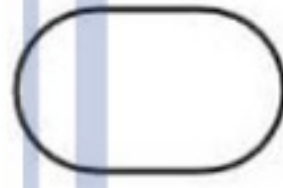
*Flowchart* (diagram alir) adalah representasi grafis dari alur suatu proses atau algoritma yang menggunakan simbol-simbol standar untuk menggambarkan langkah-langkah, keputusan, dan urutan operasi secara sistematis [16]. *Flowchart* digunakan secara luas dalam rekayasa perangkat lunak dan analisis sistem sebagai alat bantu perancangan untuk memvisualisasikan logika program sebelum implementasi, sehingga memudahkan pemahaman, komunikasi, dan dokumentasi sistem.

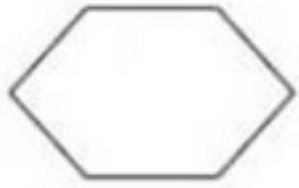
Menurut *Pressman* dan *Maxim* [16], *flowchart* merupakan salah satu teknik pemodelan proses yang paling mendasar dalam rekayasa perangkat lunak. *Flowchart* memungkinkan pengembang untuk memetakan alur kerja sistem, mengidentifikasi titik keputusan, dan mendeteksi potensi *bottleneck* atau kesalahan logika sebelum penulisan kode. Terdapat beberapa jenis *flowchart*, antara lain:

1. *System Flowchart*, yang menggambarkan aliran data dan kontrol antarkomponen sistem.
2. *Program Flowchart*, yang merinci logika algoritma dalam program.
3. *Process Flowchart*, yang menggambarkan urutan aktivitas dalam suatu proses bisnis.

Dalam penelitian ini, *flowchart* digunakan untuk menggambarkan alur tahapan penelitian secara keseluruhan (Gambar 3.1), alur kerja aplikasi *Smart-Budgeting* (Gambar 3.3), serta alur modul prediksi *XGBoost* dan *KAN*. Simbol-simbol standar yang digunakan dalam pembuatan *flowchart* pada penelitian ini mengacu pada standar *American National Standards Institute (ANSI)* dan *International Organization for Standardization (ISO)* sebagaimana dirangkum dalam Tabel 2.2.

Tabel 2.2 Simbol - Simbol pada Flowchart [19]

| No. | Gambar                                                                              | Nama                      | Deskripsi                                                                                                                                  |
|-----|-------------------------------------------------------------------------------------|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  |    | <i>Flow</i>               | Simbol yang digunakan untuk menggabungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga dengan Connecting Line. |
| 2.  |    | <i>On-Page Reference</i>  | Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang sama.                                                         |
| 3.  |   | <i>Off-Page Reference</i> | Simbol untuk keluar - masuk atau penyambungan proses dalam lembar kerja yang berbeda.                                                      |
| 4.  |  | <i>Terminator</i>         | Simbol yang menyatakan awal atau akhir suatu program.                                                                                      |
| 5.  |  | <i>Process</i>            | Simbol yang menyatakan suatu proses yang dilakukan komputer.                                                                               |
| 6.  |  | <i>Decision</i>           | Simbol yang menunjukkan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, yaitu ya dan tidak.                               |
| 7.  |  | <i>Input/Output</i>       | Simbol yang menyatakan proses input atau output tanpa tergantung peralatan.                                                                |
| 8.  |  | <i>Manual Operation</i>   | Simbol yang menyatakan suatu proses yang tidak dilakukan oleh komputer.                                                                    |
| 9.  |  | <i>Document</i>           | Simbol yang menyatakan bahwa input berasal dari dokumen dalam bentuk fisik, atau output yang perlu dicetak.                                |
| 10. |  | <i>Predefine Process</i>  | Simbol untuk pelaksanaan suatu bagian (sub-program) atau procedure.                                                                        |

|     |                                                                                   |                    |                                                                                                    |
|-----|-----------------------------------------------------------------------------------|--------------------|----------------------------------------------------------------------------------------------------|
| 11. |  | <i>Display</i>     | Simbol yang menyatakan peralatan output yang digunakan.                                            |
| 12. |  | <i>Preparation</i> | Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal. |

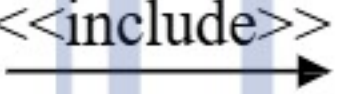
## 2.10 Diagram Use Case (UML)

*Unified Modeling Language (UML)* adalah bahasa pemodelan standar yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan artefak sistem perangkat lunak [17]. *UML* pertama kali distandardisasi oleh *Object Management Group (OMG)* pada tahun 1997 dan telah menjadi standar industri yang diterima secara universal dalam rekayasa perangkat lunak berorientasi objek. *UML* menyediakan berbagai jenis diagram yang saling melengkapi untuk menggambarkan aspek struktural maupun perilaku sistem.

*Use Case Diagram* merupakan salah satu diagram perilaku (*behavioral diagram*) dalam *UML* yang digunakan untuk menggambarkan fungsionalitas sistem dari sudut pandang pengguna (aktor) [18]. *Use Case Diagram* mendeskripsikan interaksi antara aktor dengan sistem melalui serangkaian skenario yang disebut *use case*. Menurut Rosa dan Shalahuddin [18], *use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. *Use case diagram* sangat berguna dalam tahap analisis kebutuhan (*requirements analysis*) karena memungkinkan tim pengembang dan pemangku kepentingan (*stakeholder*) untuk bersama-sama memahami dan menyepakati cakupan sistem yang akan dibangun.

Dalam penelitian ini, *use case diagram* digunakan untuk menggambarkan interaksi pengguna (*User* dan *ML Backend*) dengan fitur-fitur aplikasi *Smart-Budgeting*, mencakup fitur pencatatan transaksi, visualisasi keuangan, dan prediksi pengeluaran. Komponen-komponen standar yang digunakan dalam *use case diagram* pada penelitian ini mengacu pada standar *UML 2.5* sebagaimana dirangkum dalam Tabel 2.3.

Tabel 2.3 Simbol - Simbol pada Use Case Diagram [20]

| No. | Gambar                                                                              | Nama                   | Deskripsi                                                                                  |
|-----|-------------------------------------------------------------------------------------|------------------------|--------------------------------------------------------------------------------------------|
| 1.  |    | <i>Use Case</i>        | Digunakan sebagai penerangan tentang apa yang akan dikerjakan oleh suatu sistem.           |
| 2.  |    | <i>Actor</i>           | Digunakan sebagai penggambaran orang yang menyediakan atau menerima informasi dari sistem. |
| 3.  |    | <i>Sistem Boundary</i> | Menggambarkan jangkauan sistem.                                                            |
| 4.  |  | <i>Association</i>     | Menggambarkan bagaimana orang dapat terlibat dalam <i>use case</i> .                       |
| 5.  |  | <i>Generalization</i>  | Diperlukan saat adanya situasi atau kondisi khusus                                         |
| 6.  |  | <i>Extend</i>          | Perluasan suatu <i>use case</i> lain dikarenakan syarat-syarat telah terpenuhi.            |
| 7.  |  | <i>Include</i>         | Penjelasan bahwa <i>use case</i> ada pada <i>use case</i> lainnya.                         |

## 2.11 Penelitian Terdahulu

Beberapa penelitian terdahulu yang relevan dengan topik ini dirangkum dalam Tabel 2.1 sebagai landasan ilmiah dan konteks komparatif.

*Liu et al.*, memperkenalkan arsitektur *KAN* pada *ICLR 2025* sebagai alternatif dari *MLP* berbasis teorema *Kolmogorov-Arnold* [4]. Mereka mengevaluasi *KAN* pada berbagai tugas termasuk *fitting* fungsi matematika, pemodelan fenomena fisika, dan beberapa *benchmark ML* klasik. Hasil menunjukkan *KAN* dapat mencapai akurasi kompetitif atau lebih baik dari *MLP* yang setara dengan parameter yang lebih sedikit, khususnya untuk dataset berskala kecil-menengah dengan struktur fungsi yang *smooth*. Namun, evaluasi *KAN* pada dataset tabular keuangan berbasis profil demografis-finansial tidak dilakukan dalam penelitian tersebut, sehingga kinerja komparatifnya terhadap metode *ensemble* seperti *XGBoost* untuk tugas ini masih merupakan pertanyaan terbuka.

*Liu et al.*, lebih lanjut mengembangkan *KAN* menjadi *KAN 2.0* dengan penambahan *MultKAN* dan peningkatan kemampuan simbolik [6]. Meskipun *KAN 2.0* belum dievaluasi secara ekstensif pada dataset keuangan, peningkatan kapabilitas jaringan dalam menangkap interaksi perkalian antar fitur berpotensi relevan untuk data keuangan di mana interaksi antara pendapatan, rasio utang, dan skor kredit sering bersifat multiplikatif.

*Chen dan Guestrin*, memperkenalkan *XGBoost* pada *KDD 2016*, menunjukkan bahwa sistem ini mencapai kinerja *state-of-the-art* pada berbagai *benchmark* dengan efisiensi komputasi yang jauh lebih tinggi dibandingkan implementasi *gradient boosting* sebelumnya [7]. Sejak saat itu, *XGBoost* dan variannya (*LightGBM*, *CatBoost*) secara konsisten mendominasi kompetisi *machine learning* untuk data tabular. Penelitian ini menggunakan *XGBoost* sebagai *baseline* kuat (*strong competitive baseline*) yang harus dibandingkan secara *fair* dengan *KAN*.

Berbagai algoritma *machine learning* seperti *Random Forest*, *XGBoost*, dan *Artificial Neural Network* banyak digunakan dalam prediksi data tabular karena kemampuannya dalam memodelkan hubungan non-linear antar variabel. Di antara algoritma tersebut, *XGBoost* dikenal memiliki performa yang unggul dalam hal akurasi dan efisiensi komputasi, sehingga sering digunakan sebagai *baseline* dalam berbagai penelitian prediksi.

*Hastie, Tibshirani*, dan *Friedman* dalam buku referensi komprehensif "*The Elements of Statistical Learning*" memberikan fondasi teoritis yang mendalam untuk algoritma *ML* termasuk *gradient boosting*, *decision trees*, regularisasi, dan metrik evaluasi regresi [3]. Buku ini menjadi landasan untuk pemahaman teoritis metrik *MAE*, *RMSE*,  $R^2$ , dan prosedur *preprocessing* yang digunakan dalam penelitian ini.

Prasetyo dalam buku "*Data Mining: Konsep dan Aplikasi Menggunakan MATLAB*" memberikan referensi komprehensif dalam bahasa Indonesia mengenai berbagai teknik *machine learning* dan *preprocessing data* [10]. Buku ini menjadi referensi penting untuk kontekstualisasi teknik-teknik yang digunakan dalam penelitian ini ke dalam perspektif akademik Indonesia.

*Hyndman dan Koehler*, menganalisis secara sistematis kelebihan dan kekurangan berbagai metrik akurasi prediksi termasuk *MAE*, *RMSE*, *MAPE*, dan turunannya [12]. Mereka menyimpulkan bahwa tidak ada satu metrik tunggal yang superior untuk semua kondisi penggunaan *multiple metrics* secara komplementer memberikan gambaran yang paling komprehensif mengenai kualitas prediksi. Rekomendasi ini menjadi dasar pemilihan empat metrik evaluasi dalam penelitian ini.

*Lundberg dan Lee* memperkenalkan *SHAP* (*SHapley Additive exPlanations*) sebagai kerangka untuk interpretabilitas model machine learning. Konsep *SHAP* digunakan sebagai referensi dalam memahami metode interpretabilitas model, sedangkan penelitian ini menggunakan feature importance bawaan XGBoost sebagai salah satu analisis untuk melihat kontribusi relatif fitur terhadap model [14].

*Akiba et al.* memperkenalkan *Optuna* sebagai *framework* optimasi *hyperparameter* yang menggunakan pendekatan *Tree-structured Parzen Estimator (TPE)* dan mendukung mekanisme *pruning* terhadap trial yang kurang menjanjikan [15]. Penelitian tersebut menjadi salah satu referensi mengenai teknik optimasi *hyperparameter* pada *machine learning*. Namun, *Optuna* tidak digunakan dalam eksperimen final penelitian ini karena konfigurasi *hyperparameter* model ditentukan secara eksplisit berdasarkan rancangan eksperimen..

Tabel 2.4 Ringkasan Penelitian Terdahulu

| No. | Peneliti / Tahun       | Algoritma Utama          | Domain / Dataset                    | Metrik / Hasil                                    | Relevansi dengan Penelitian Ini                                                |
|-----|------------------------|--------------------------|-------------------------------------|---------------------------------------------------|--------------------------------------------------------------------------------|
| [4] | Liu et al. (2025)      | <i>KAN vs MLP</i>        | <i>Math &amp; Physics Benchmark</i> | Akurasi kompetitif dengan parameter lebih sedikit | Arsitektur <i>KAN</i> orisinal; evaluasi pada data tabular keuangan belum ada. |
| [6] | Liu et al. (2024)      | <i>KAN 2.0 (MultKAN)</i> | <i>Scientific Discovery</i>         | Kemampuan simbolik meningkat                      | Perkembangan terkini <i>KAN</i> ; basis untuk <i>future work</i> .             |
| [7] | Chen & Guestrin (2016) | <i>XGBoost</i>           | <i>General ML Benchmark</i>         | <i>State-of-the-art</i> pada                      | Baseline kuat yang digunakan; fondasi teknis <i>XGBoost</i> .                  |

| No.  | Peneliti / Tahun      | Algoritma Utama     | Domain / Dataset                   | Metrik / Hasil                               | Relevansi dengan Penelitian Ini                                                                           |
|------|-----------------------|---------------------|------------------------------------|----------------------------------------------|-----------------------------------------------------------------------------------------------------------|
|      |                       |                     |                                    | ratusan <i>benchmark</i>                     |                                                                                                           |
| [14] | Lundberg & Lee (2017) | <i>SHAP</i>         | <i>General ML Interpretability</i> | <i>Unified feature attribution</i>           | Digunakan untuk analisis <i>feature importance</i> yang lebih akurat.                                     |
| [15] | Akiba et al. (2019)   | <i>Optuna (TPE)</i> | <i>Hyperparameter Optimization</i> | Efisiensi lebih baik dari <i>Grid Search</i> | Referensi teknik optimasi <i>hyperparameter</i> ; tidak digunakan dalam eksperimen final penelitian ini.. |

Berdasarkan tinjauan penelitian terdahulu yang komprehensif di atas, terdapat kesenjangan penelitian yang jelas: belum ada studi yang secara langsung membandingkan *KAN* dan *XGBoost* pada dataset keuangan personal berbasis profil demografis-finansial seperti *Personal Finance ML Dataset*. Penelitian ini hadir untuk mengisi kesenjangan tersebut dengan metodologi yang terstruktur, *rigorous*, dan berorientasi pada aplikasi praktis (*Smart-Budgeting*).