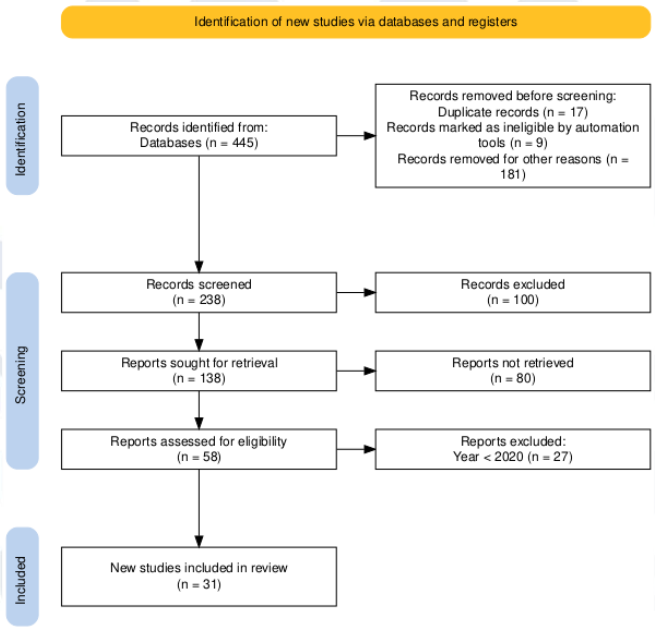
# BAB II

# LANDASAN TEORI

## 2.1 Penelitian Terdahulu

Penelitian terdahulu didapatkan melalui *Systematic Literature Review* (SLR) dengan metode PRISMA. SLR adalah salah satu metode literatur yang umumnya digunakan untuk menemukan, menyoroti, mengases, serta menganalisis upaya para peneliti yang melakukan penelitian pada bidang tertentu [16]. Metode PRISMA adalah panduan berbasis bukti yang dirancang untuk membantu penulis dalam melaporkan berbagai tinjauan sistematis dan meta-analisis yang mengevaluasi manfaat [17]. Proses SLR dengan metode PRISMA yang dilakukan pada penelitian ini dimulai dengan cara mengumpulkan data mengenai artikel-artikel jurnal yang berkaitan dengan topik penelitian yang dipilih, dengan memasukkan *keyword* "*house price prediction and machine learning", "machine learning and hyperparameter tuning",* dan *"optuna".* Hasilnya, sebanyak 445 data berhasil dikumpulkan. Dari 445 data artikel jurnal tersebut, perlu dilakukan proses *filter* sehingga didapatkan 31 data yang kemudian 10 diantaranya akan digunakan dalam penelitian ini. Berikut ini adalah diagram PRISMA yang digunakan untuk mencari penelitian terdahulu yang digunakan pada penelitian ini.




Gambar 2.1 Diagram PRISMA

Selanjutnya, tabel 2.1 dan 2.2 berisi 10 artikel jurnal penelitian terdahulu yang didapatkan dengan teknik SLR melalui metode PRISMA.

Tabel 2.1 Tabel Penelitian Terdahulu Mengenai Algoritma *Machine Leaning*

| Nama Jurnal | Nama Artikel | Penulis | Metode | Hasil |
|---|---|---|---|---|
| *Journal of Applied Computer Science and Technology (JACOST) Vol. 4 No. 1 (2023)* [10] | Analisis Perbandingan Metode Regresi Linier, *Random Forest Regression*, dan *Gradient Boosted Trees Regression Method* untuk Prediksi Harga Rumah | Evita Fitri | - *Multiple Linear Regression*<br>- *Random Forest*<br>- *Gradient Boosted Trees* | $R^2$ *Random Forest:* 81,5 % |
| *Journal on Advanced Research in Electrical Engineering (JAREE) Vol. 7 No. 1 (2023)* [11] | *House Price Prediction using Multiple Linear Regression and KNN* | Fransiskus Dwi Febriyanto, Endroyono, Yoyon Kusnendar | - *Multiple Linear Regression*<br>- *KNN* | $R^2$ *Multiple Linear Regression:* 70,7% |
| *Internation Journal of INformation Engineering and Electronic Business (IJIEEB) Vol. 12 No. 2 (2020)* [12] | *House Price Prediction Modeling Using Machine Learning* | M. Thamarai, SP. Malarvizhi | - *Multiple Linear Regression*<br>- *Decision Tree* | *RMSE Multiple Linear Regression:* 2,45 |

| Nama Jurnal | Nama Artikel | Penulis | Metode | Hasil |
|---|---|---|---|---|
| *Concurrency and Computation: Practice and Experience Vol. 4 No. 27 (2022)* [13] | *House Price Prediction Using Hedonic Pricing Model and Machine Learning Techniques* | John Zaki, Anand Nayyar, Surjeet Dalal, Zainab H. Ali | - *XGBoost*<br>- *Hedonic Pricing Model* | $R^2$ *XGBoost:* 84,1% |
| Jurnal Informatik Vol. 17 No. 3 (2021) [15] | Analisis Prediksi Harga Rumah Sesuai Spesifikasi Menggunakan *Multiple Linear Regression* | Muhammad Labib Mu'tashim, Sekar Ayu Damayanti, Hanan Nadia Zaki, Toni Muhayat, Rio Wirawan | - *Multiple Linear Regression* | $R^2$ *Multiple Linear Regression:* 66% |
| *PLOS ONE Vol. 16 No. 2 (2021)* [18] | *Prediction the Rental Value of Houses in Household Surveys in Tanzania, Uganda, and Malawi: Evaluations of HedonicPricing and Machine Learning Approaches* | Weldensie T. Embaye, Yacob Abrehe Zereyesus, Bowen Chen | - *Decision Tree*<br>- *Random Forest*<br>- *Gradient Boosting* | *MSE Random Forest:* 0,83 |
| *Highlights in Business, Economics and Management Vol. 21 (2023)* [19] | *House Price Prediction and Analysis Based on Random Forest and XGBoost Models* | Han Li | - *Random Forest*<br>- *XGBoost* | $R^2$ *XGBoost:* 89% |

Tabel 2.2 Tabel Penelitian Terdahulu Mengenai *Optuna*

| Nama Jurnal | Nama Artikel | Penulis | Metode | Hasil |
|---|---|---|---|---|
| *Jurnal Informatika & Rekayaya Elektronika Vol. 7 No. 1 (2024) [20]* | *Analysis of the Application of Hyperparameter Tuning in Machine Learning to Increase the Accuracy of Sales-level Prediction* | Sugiyanti, Muhammad Haris | *Hyperparameter Tuning* dengan *Optuna* | *Optuna Hyperparameter Tuning* mampu mengurangi *error* pada algoritma *XGBoost* |
| Jurnal Informatika Polinema Jurusan Teknologi Informasi Politeknik Negeri Malang Vol. 10 No. 4 (2024) | Analisis Perbandingan *Linear Regression* dan *Random Forest Regression* untuk Prediksi Batas Kredit: Pendekatan Optimasi *Hyperparameter* | Algies Rifkha Fadillah, Mohammad Nurkamal Fauzan | - *Multiple Linear Regression*<br>- *Random Forest*<br>- *Hyperparameter Tuning* dengan *Optuna* | *Hyperparameter Tuning* dengan *Optuna* mampu meningkatkan performa *Random Forest* dan *Multiple Linear Regression* |
| *2024 IEEE 2nd Internasional Conference on Sensors, Electronics, and Computer Engineering (ICSECE)* [20] | *Research on Electricity Price Forecast Model Based On Electricity Price Formation Mechanism and XGBoost and Highly Automated Optuna Algorithm* | Ting Li | *Hyperparameter Tuning dengan Optuna* pada *XGBoost* | *Optuna* mampu melakukan *hyperparameter tuning* secara otomatis pada *XGBoost* |

Pada penelitian terdahulu [10], penelitian tersebut melakukan komparasi pada tiga algoritma *machine learning* (*Multiple Linear Regression, Random Forest,* dan *Gradient Boosting)* dalam memprediksi harga rumah di Jakarta Selatan. Hasilnya, algoritma *Random Forest* memberikan akurasi hasil prediksi yang cukup baik dengan nilai *R-squared* 81,5%. Selain itu, pada penelitian [18] disebutkan bahwa algoritma *Ranadom Forest* mampu memberikan nilai *Mean Squared Error* terendah diantara algoritma *Decision Tree* dan *Gradient Boosting,* yaitu sebesar 0,83. Berbeda dengan dua penelitian terdahulu yang disebutkan sebelumnya, penelitian [13] dan [19] menggunakan algoritma *XGBoost* untuk membuat prediksi harga rumah. Dari penelitian [13] dan [19], algoritma *XGBoost* yang digunakan masing-masing memberikan nilai *R-squared* sebesar 84,1% dan 89%. Sementara itu, pada penelitian [11], [12], [15], ketiga penelitian tersebut menggunakan algoritma *Multiple Linear Regression* untuk memprediksi harga rumah yang masing-masing hasilnya memberikan nilai *R-squared* 77,07% [11] dan 66% [15], serta *Root Mean Squared Error* (RMSE) sebesar 2,45 [12],

Di sisi yang lain, penelitian [20], [21], [22], menyebutkan bahwa melakukan *hyperparameter tuning* dapat meningkatkan akurasi serta mengurangi *error* yang ada pada algoritma *machine learning Multiple Linear Regression, Random Forest,* dan *XGBoost.* Ketiga penelitian tersebut menggunakan *library* Optuna dalam melakukan *hyperparameter tuning* pada masing-masing algoritma *machine learning* yang digunakan pada masing-masing penelitian.

## 2.2 Objek Penelitian

### 2.2.1 Rumah

Apabila mengacu pada Kamus Besar Bahasa Indonesia, rumah memiliki pengertian sebagai bangunan tempat tinggal [23]. Dalam Undang-Undang No.4 Tahun 1992, rumah diartikan sebagai bangunan yang berfungsi sebagai tempat tinggal atau hunian dan sarana pembinaan keluarga [24]. Dilansir dari situs Pinhome.id, rumah adalah sebuah bangunan yang dijadikan tempat tinggal untuk mendapatkan perlindungan dari segala kondisi alam yang berada di sekitarnya oleh manusia [25]. Berdasarkan ketiga pengertian rumah sebelumnya, maka dapat disimpulkan

bahwa rumah adalah bangunan tempat tinggal dan sarana pembinaan keluarga, yang digunakan untuk beraktivitas, beristirahat, dan tidur setelah melakukan aktivitas harian. Rumah terbagi ke dalam beberapa kategori, yaitu rumah tapak, rumah vertikal, rumah tunggal, *town house*, *cluster*, rumah toko, dan rumah kantor [25]. Fokus dari penelitian ini adalah untuk memprediksi harga rumah tapak yang ada di wilayah Kabupaten Tangerang.

### 2.2.2 Sumber Data

Sumber data yang digunakan dalam penelitian ini bersumber dari salah satu *website marketplace* properti yang ada di Indonesia, yaitu rumah123.com. Rumah123.com merupakan salah satu platform teknologi terkemuka untuk melakukan jual, beli, dan sewa properti di Indonesia, yang telah beroperasi sejak 2007 dan telah melayani jutaan pengguna [26]. Rumah123.com menyediakan tempat pemasangan iklan secara daring yang bisa diakses melalui internet dari Indonesia maupun luar Indonesia. Situs rumah123.com dipilih dibandingkan situs-situs jual beli properti yang lain karena data-data yang disediakan pada situs rumah123 tergolong lengkap dan terpercaya. Selain itu, situs rumah123.com juga mengizinkan pengguna untuk melakukan pengambilan data dengan teknik *web scraping* dengan persyaratan tertentu agar tidak melanggar aturan situs mengenai *web scraping* [27].

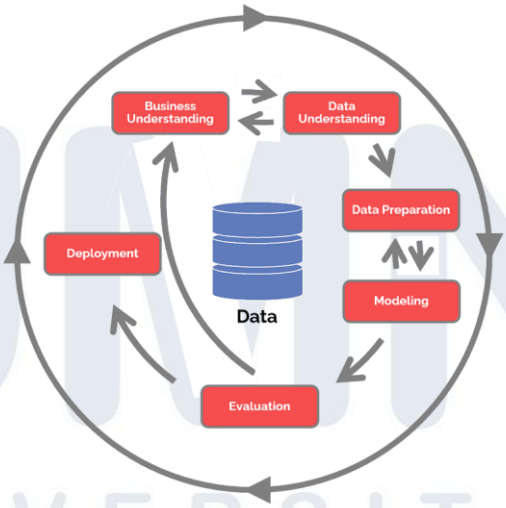
Teknik *web scraping* digunakan dengan bantuan dari *package Selenium dan BeautifulSoup4* yang tersedia secara gratis untuk bahasa pemrograman *python. Web scraping* dilakukan sebanyak dua kali: Pertama, menggunakan *package selenium* untuk mendapatkan tautan URL dari setiap rumah yang muncul di situs rumah123.com. Kedua, menggunakan *package beautifulsoup4* untuk mendapatkan spesifikasi rumah dari tautan URL yang telah didapatkan sebelumnya. Proses *web scraping* memakan waktu sekitar dua minggu (minggu pertama bulan Februari 2025 sampai dengan minggu kedua bulan Februari 2025) untuk mendapatkan sekitar 1806 data harga rumah di Kabupaten Tangerang beserta dengan spesifikasinya.

*Web scraping* dilakukan selama dua minggu karena pada *website* rumah123.com terdapat *bot* yang dapat mendeteksi tindakan *web scraping* pada level IP. Jika ketahuan maka *website* akan memunculkan sebuah *captcha* sehingga *code* untuk melakukan *web scraping*-nya tidak dapat di digunakan. Selain itu, waktu yang diperlukan untuk memperoleh 100 data spesifikasi harga rumah dari proses *web scraping* ini memakan waktu sekitar 1 jam. Oleh karena itu, *web scraping* dilakukan secara bertahap selama dua minggu untuk mendapatkan data yang diperlukan.

## 2.3 Algoritma dan *Framework*

### 2.3.1 CRISP-DM

*Cross Industry Standard Process for Data Mining* atau yang biasa disebut sebagai CRISP-DM adalah salah satu metodologi standar yang digunakan dalam sebuah siklus proyek *data mining* [28]. Dalam CRISP-DM, terdapat enam tahapan iteratif yang dimulai dari tahap *business understanding* hingga *deployment* [29]. Untuk ilustrasinya, dapat dilihat pada gambar 2.2 di bawah.




Gambar 2.2 Siklus CRISP-DM
Sumber: [30]

Berikut ini adalah penjelasannya:

1. ***Business Understanding*:** Pada tahap ini, penentuan tipe *data mining* yang akan dilakukan, metrik pengukuran keberhasilan

proyek, dan gambaran besar dari masalah yang akan diselesaikan harus didefinisikan dengan jelas [29].

2. ***Data Understanding***: Pada tahap ini, proses pengumpulan *dataset* yang akan digunakan sampai dengan proses *exploratory data analysis* akan dilakukan guna mendapatkan *insight* dari dataset [29].
3. ***Data Preparation***: Pada tahap ini, akan dilakukan *data pre-processing. Data pre-processing* disini mencakup *data cleaning*, standarisasi/normalisasi, mengatasi *outliers*, dan sebagainya [29].
4. ***Modeling:*** Pada tahap ini, akan dilakukan pemodelan data ke dalam model *machine learning* yang akan digunakan [29].
5. ***Evaluation***: Pada tahap ini, akan dilakukan evaluasi terhadap performa model yang telah dibuat pada tahap *modeling* sebelumnya. Apabila model yang telah dibuat sebelumnya memberikan performa yang memuaskan, maka dapat lanjut ke tahap terakhir dalam CRISP-DM [29]. Tetapi, jika performa model kurang memuaskan, dapat dilakukan optimasi model kembali.
6. ***Deployment***: Merupakan tahap terakhir dalam CRISP-DM. Pada tahap ini, model yang memiliki performa terbaik dapat di *deploy*, menyesuaikan dengan bisnis yang sudah didefinisikan di tahap awal [29].

### 2.3.2 *Multiple Linear Regression*

*Multiple Linear Regression* atau yang bisa disebut sebagai regresi linear berganda merupakan model regresi yang melibatkan lebih dari satu variabel [15]. Pada *Multiple Linear Regression,* prediksi dilakukan dengan menggunakan data pada skala interval atau rasio, serta melibatkan lebih dari satu variabel independen [15]. Rumus dari *Multiple Linear Regression* dapat dilihat pada persamaan 2.1 di bawah.

$$Y = \alpha + \beta 1\, X1 + \cdots + \beta n\, Xn + e \quad (2.1)$$

Sumber: [15]

Di mana:

$Y$ : Variabel dependen

$X$ : Variabel independen

$\alpha$ : Konstanta (*Intercept*)

$\beta$ : Koefisien determinasi (*Slope*)

$e$ : *error*

### 2.3.3 *Random Forest*

Algoritma *Decision Tree* memiliki kelemahan, yaitu rentan terhadap terjadinya *overfitting*, sehingga model tidak mampu melakukan generalisasi dengan baik [31]. Salah satu cara untuk meningkatkan kemampuan generalisasi dari model *Decision Tree* adalah dengan memperhitungkan sebagian dari *dataset* dan membangun banyak *Decision Tree* berdasarkan sebagian dari *dataset* tersebut (*bootstrap*), hal ini diperkenalkan oleh Ho pada tahun 1995 dan kemudian dikembangkan serta diresmikan dengan nama *Random Forest* oleh Breiman pada tahun 2001 [31]. *Random forest* adalah model algoritma pembelajaran berdasarkan pohon ansambel (*ensemble tree*), yang menghitung rata-rata dari banyaknya pohon individu (*individual tree*) [31].

Cara kerja dari *Random Forest* terbagi ke dalam dua fase: fase pertama untuk menggabungkan sejumlah N *decision tree* dan fase kedua untuk membuat prediksi dari N *Decision Tree* yang dibuat pada fase pertama [31]. Rinciannya adalah sebagai berikut.

1. Algoritma akan memiliki sampel acak dari *dataset* yang ada (*bootstrapping*).
2. Membuat *Decision Tree* untuk setiap sampel yang dipilih.
3. Hasil prediksi dari setiap *Decision Tree* akan dilakukan proses perhitungan prediksi, dengan menggunakan nilai rata-rata (untuk masalah regresi).
4. Algoritma akan memilih hasil akhir prediksi dengan *vote* yang paling banyak dipilih.

Rumus perhitungan prediksi dengan algoritma *Random Forest* dapat dilihat pada persamaan 2.2 di bawah.

$$Prediksi = \frac{1}{K}\sum_{k=1}^{K} w_k h_k(x) \quad (2.2)$$

Sumber: [19]

Di mana:

K : Jumlah *Decision Tree.*

$w_k$ : Bobot dari $h_k$.

$h_k(x)$ : prediksi yang dihasilkan oleh *Decision Tree* ke-k untuk data *input* x.

Perhitungan *error* pada algoritma *Random Forest* dikalkulasikan menggunakan *out-of-bag* (OOB) *error* pada saat proses pelatihan model [31]. Langkah-langkahnya adalah:

1. Tiap pohon dibangun di atas sampel *bootstrap* yang berbeda.
2. Tiap sampel *bootstrap*, secara acak meninggalkan sekitar satu per tiga observasi.
3. Nilai observasi yang ditinggalkan ini yang disebut dengan sampel OOB.

Oleh karena itu, untuk menemukan parameter yang akan menghasilkan *error* OOB paling rendah menjadi pertimbangan yang krusial dalam melakukan *tuning* parameter dari model *random forest.* Selain itu, ukuran dari *subset* data yang digunakan juga akan berpengaruh dalam mengontrol kedalaman (*depth*) dari masing-masing pohon.

### 2.3.4 *Extreme Gradient Boosting Tree* (XGBoost)

Algoritma *Extreme Gradient Boosting* atau yang biasa disebut dengan *XGBoost* adalah algoritma pembelajaran mesin berbasis pohon yang di usulkan oleh Chen Tianqi pada tahun 2016 [20]. Pada dasarnya, algoritma *XGBoost* menggunakan algoritma *gradient boosting,* yang secara bertahap melakukan *training* pada sejumlah model *Decision Tree* [20]. Setiap iterasinya berfokus untuk mengoreksi kesalahan prediksi yang dihasilkan

dari model sebelumnya, sekaligus mengoptimalkan struktur dari struktur *Decision Tree* dengan pembelajaran berbobot (*weighted learning*) dan perhitungan penurunan gradien [20]. Persamaan 2.3 adalah rumus prediksi model *XGBoost* pada umumnya:

$$\hat{y}_i = \sum_{k=1}^{K} f_{k(x_i)}, \quad f_k \in F \tag{2.3}$$

Sumber: [20]

Di mana:

$\hat{y}_i$ : Prediksi model pada data ke-*i.*

$K$ : Jumlah total pohon (*tree*) yang dibangun.

$f_k$ : Model pohon ke-*k* yang termasuk dalam ruang fungsi *F* (kumpulan dari *possible trees*).

$x_i$ : Fitur dari data ke-*i.*

$F$ : Ruang fungsi dari *decision trees.*

*XGBoost* menggunakan teknik regularisasi untuk membatasi kompleksitas model agar tidak terjadi *overfitting* dan meningkatkan kemampuan model dalam memprediksi data baru (generalisasi) [20]. Selain itu, *XGBoost* juga menerapkan pendekatan ekspansi Taylor tingkat dua untuk menyederhanakan proses minimalisasi *loss function,* yang berfungsi sebagai panduan dalam membangun *decision tree* pada setiap iterasi [20]. Persamaan 2.4 di bawah merupakan rumus minimalisir *loss function* yang digunakan pada *XGBoost*.

$$\text{Obj}^{(t)*} = \gamma \cdot T_t - \frac{1}{2} \sum \frac{G_j^2}{H_j + \lambda} \tag{2.4}$$

Sumber: [20]

Di mana:

$\gamma$ : Penalti untuk jumlah daun (*leaf nodes*), agar pohon tidak terlalu besar. (L1 *Regularization*)

$T_t$ : Jumlah daun pada pohon ke-*t.*

$G_j$ : Jumlah turunan pertama (gradien/kemiringan) pada *node j* (kesalahan model).

$H_j$ : Jumlah turunan kedua (hessian/seberapa tajam kemiringannya) pada *node j* (perubahan kesalahan).

$\lambda$ : Penalti untuk mencegah bobot daun terlalu ekstrem (L2 *Regularization*).

Secara garis besarnya, algoritma *XGBoost* bekerja dengan cara melakukan iterasi pada banyak *decision tree.* Pada setiap *node* dalam *tree, XGBoost* akan menghitung gradien (gradien, $G_j$) dan turunan kedua (hessian, $H_j$) dari *loss function* untuk mengetahui seberapa besar *error* yang terjadi pada model. Berdasarkan rumus perhitungan minimalisir *loss function* dia atas, *XGBoost* akan membangun *decision tree* pada setiap iterasi, dan semua pohon tersebut akan dijumlahkan untuk membentuk model akhir *XGBoost* yang kuat. Berikut ini adalah cara kerja dari algoritma *XGBoost*.

1. Membangun pohon regresi baru untuk memperbaiki kesalahan prediksi yang dibuat sebelumnya.
2. Menghitung nilai awal turunan pertama (gradien) dan turunan kedua (hessian), agar model dapat terus diperbaiki di setiap iterasi.
3. Menentukan titik pemisah (*splitting point*) terbaik menggunakan algoritma *greedy* (memilih solusi terbaik pada setiap langkah).
4. Menambahkan pohon regresi ke dalam model secara bertahap.
5. Ulangi langkah 1 sampai 4 sampai model mencapai kondisi konvergen.

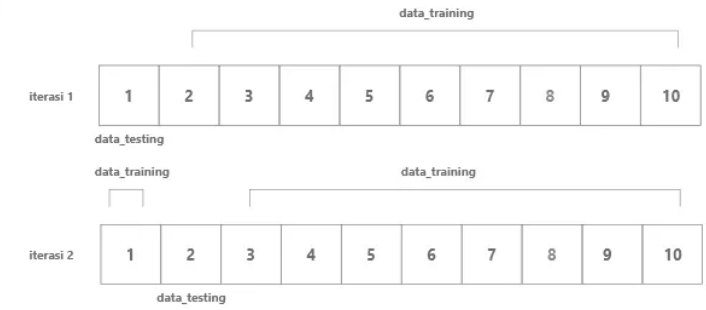
### 2.3.5 *Hyperparameter Tuning* dengan Optuna

Optuna merupakan salah satu *framework hyperparameter tuning* pada *machine learning* yang bekerja berdasarkan metode optimasi Bayesian, yang secara cerdas mampu untuk melakukan eksplorasi pada *hyperparameter* untuk menemukan kombinasi terbaik yang mampu memaksimalkan atau meminimalisir fungsi objektif (*objective function*) dengan sumber daya komputasi seminimal mungkin [20]. Optuna dapat mencari *hyperparameter* terbaik secara cepat dan efisien, sehingga dapat meningkatkan performa dari model [20]. Optuna bekerja dengan

memanfaatkan probabilitas untuk membangun model fungsi target, yang kemudian digunakan untuk memilih konfigurasi yang paling efektif dalam meningkatkan kinerja model [20].

### 2.3.6 Validasi Model

*Cross-validation* adalah salah satu metode yang umum digunakan untuk mencegah terjadinya *overfitting* dengan menggunakan seluruh *dataset* sebagai data latih dan data validasi [32]. Data latih akan dibagi ke dalam *K* bagian yang bisa disebut *fold.* Data akan dilatih dengan menggunakan data pertama dari *fold* sebagai data latih, kemudian hasil akurasinya akan di validasi pada sisa *foldnya.* Proses ini dilakukan sebanyak *K* kali, di mana dengan setiap *K-fold* digunakan satu kali untuk validasi [32]. Untuk memperjelas, ilustrasi 2.3 di bawah merupakan gambaran umum mengenai teknik *K-Fold cross validation.*




Gambar 2.3 *K-Fold Cross Validation*
Sumber: [33]

### 2.3.7 Metrik Evaluasi

Untuk kasus regresi, jenis metrik evaluasi yang digunakan berbeda dengan metrik evaluasi yang digunakan pada kasus klasifikasi. Pada kasus regresi, metrik evaluasi yang umumnya dipakai adalah $R^2$, *Mean Absolute Percentage Error* (MAPE), *Root Mean Squared Error* (RMSE), dan *Root Mean Squared Error* (RMSE) [34] [35]. Akan tetapi, pada penelitian ini, hanya metrik $R^2$, MAPE, dan RMSE saja yang akan digunakan. Berikut ini adalah penjelasan mengenai metrik $R^2$, MAPE, dan RMSE.

**a. *R-Squared* ($R^2$)**

$R^2$ atau yang bisa disebut dengan *R-squared* adalah nilai yang menampilkan pengaruh variabel independen terhadap variabel independen [36]. Nilai $R^2$ berkisar antara 0 sampai dengan 1. Rumus dari $R^2$ dapat dilihat pada persamaan 2.5 di bawah:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y})^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2} \quad (2.5)$$

Sumber: [36]

Di mana:

$y_t$ : data yang diuji ke-t (t=1, …,n),

$\hat{y}$ : prediksi dari respons ke-t (t=1, …,n),

$\bar{y}$ : rata-rata,

$n$ : banyaknya data yang diuji.

Selanjutnya, tabel 2.2.3 di bawah menunjukkan interpretasi dari nilai $R^2$.

Tabel 2.2.3 Interpretasi Nilai $R^2$

| Interval Koefisien | Hubungan |
|---|---|
| 1 – 0,8 | Sangat kuat |
| 0,6 – 0,79 | Kuat |
| 0,4 – 0,56 | Cukup kuat |
| 0,2 – 0,39 | Lemah |
| 0 – 0,19 | Sangat lemah |

Sumber: [36]

**b. *Mean Absolute Percentage Error* (MAPE)**

*Mean Absolute Percentage Error* (MAPE) menghitung rata-rata persentase kesalahan absolut antara nilai prediksi dengan nilai sebenarnya, yang dinyatakan dalam bentuk persentase [37]. Rumus dari MAPE dapat dilihat pada persamaan 2.6 di bawah.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| * 100\% \quad (2.6)$$

Sumber: [37]

Di mana:

$A_t$ : nilai aktual ke-*t*,

$F_t$ : nilai prediksi ke-*t*,

*n* : banyaknya data yang diuji.

Semakin kecil nilai perhitungan MAE, maka semakin akurat hasil prediksi tersebut [37].

**c. *Root Mean Squared Error* (RMSE)**

*Root Mean Squared Error* (RMSE) akan menghitung selisih kuadrat antara nilai prediksi dan nilai aktual untuk setiap pengamatan, kemudian menentukan rata-rata dari selisih kuadrat tersebut, dan akhirnya mengambil akar kuadrat dari rata-rata tersebut [37]. Rumus perhitungan nilai MAPE dapat dilihat pada persamaan 2.7 di bawah.

$$RMSE = \sqrt[2]{\frac{1}{n}\sum_{t=1}^{n}(y_t - f_t)^2} \quad (2.7)$$

Sumber: [37]

Di mana:

$f_t$ : nilai prediksi ke-t,

$y_t$ : nilai aktual ke-*t*,

*n* : banyaknya data yang diuji.

Semakin kecil nilai perhitungan RMSE, maka semakin akurat hasil prediksi tersebut [37].

## 2.4 *Tools* dan Teknik

### 2.4.1 Python

*Python* merupakan salah satu bahasa pemrograman tingkat tinggi yang mudah untuk dipelajari [38]. *Python* dikatakan sebagai bahasa pemrograman tingkat tinggi karena memiliki sintaksis kalimat yang menyerupai bahasa yang digunakan oleh manusia. Oleh karena itu, *python* dianggap mudah untuk dipelajari.

*Python* menjadi salah satu *tools* yang banyak digunakan di bidang *data science*. Hal ini didukung dengan banyaknya *library* yang memang ditujukan untuk bidang *data science*, seperti *pandas*, *numpy*, *scikit-learn*, *matplotlib*, dan *seaborn*. *Library pandas* dan *numpy* digunakan untuk memanipulasi data, *library scikit-learn* digunakan untuk pemodelan *machine learning*, dan *library matplotlib* dan juga *seaborn* digunakan untuk visualisasi data.

Selain itu, *python* juga menyediakan *library* untuk melakukan *web scraping* dengan *library selenium* dan *beautifulsoup4*. *Library selenium* umumnya digunakan untuk melakukan *web crawling* pada halaman *website* yang dinamis (menggunakan *javascript*) [39]. Dalam penelitian ini, penggunaan *library selenium* digunakan untuk melakukan *web crawling* pada situs rumah123.com untuk mendapatkan tautan URL dari tiap rumah yang ditampilkan di situs rumah123.com. Sementara itu, *library beauitfulsoup4* digunakan untuk mendapatkan spesifikasi rumah dari tautan URL yang telah didapatkan menggunakan *library selenium* sebelumnya.

Terakhir, *python* juga menyediakan *library streamlit* untuk mempermudah melakukan *deployment* dari model *machine learning* yang telah dibuat dalam bentuk *website* [40].

### 2.4.2 Jupyter

*Jupyter* adalah salah satu *software open-source* yang populer digunakan dalam dunia pemrograman, lebih spesifiknya di bidang *data science*, analisis data, dan pengembangan perangkat lunak. *Jupyter* merupakan *software open-source* yang digunakan oleh komunitas untuk membangun perangkat lunak yang interaktif dengan dukungan puluhan bahasa pemrograman [41]. Bahasa pemrograman yang paling banyak dipakai di *jupyter* adalah *Python*. *Jupyter* memungkinkan penggunanya untuk berbagi dokumen yang menggabungkan komutasi kode komputer, narasi, dan elemen lainnya dalam satu dokumen yang disebut *notebook* [42].

Berikut ini adalah fitur yang dapat digunakan pada *jupyter* [42].

1. Interaktif: *Jupyter* memungkinkan pengguna untuk menjalankan kode *python* secara interaktif. Pengguna dapat menjalankan satu perintah atau sekelompok perintah kode, dan hasilnya akan ditampilkan segera di dalam *notebook*.
2. Pemrograman dalam blok: Dalam *jupyter notebook*, kode akan dijalankan per blok sehingga tampilannya rapi.
3. Visualisasi: Pengguna dapat membuat visualisasi data dengan mudah dan menarik.
4. Dukungan berbagai bahasa pemrograman: Dukungan bahasa pemrograman yang luas, mulai dari *python*, R, dan lainnya.
5. Kemudahan berbagi: pengguna dapat dengan mudah berbagi *notebook* dengan orang lain, dengan format statis (HTML) ataupun interaktif (melalui *jupyter notebook server*).