Laporan UAS
PROJECT-BASED EXAM
AMS233225-03 – Analisis Prediktif

Dosen Pengajar:

Robyn Irawan, M.Sc.

Disusun oleh :

Vincent Matthew Tjandra	6162001068
Bryan Ernestin	6162001097
Leonardo Alindra	6162001111
Bisma Leksono	6162001117

Program Studi Matematika
Fakultas Teknologi Informasi dan Sains
Universitas Katolik Parahyangan

2024

ABSTRAK

Laporan ini menyajikan analisis regresi lanjutan pada data "House Prices - Advanced Regression Techniques" dari Kaggle. Metode Multiple Imputation by Chained Equations (MICE) digunakan untuk mengatasi nilai yang hilang, sedangkan koefisien korelasi dan uji ANOVA digunakan untuk pemilihan fitur. Selanjutnya, model Random Forest, Light GBM, Neural Network, SVM, dan Elastic Net diterapkan untuk memprediksi harga rumah, dan hasilnya dievaluasi menggunakan platform Kaggle. Hasil eksperimen menunjukkan bahwa model LGBM memiliki nilai Root Mean Squared Error (RMSE) terkecil, sementara model XGBoost memiliki nilai R^2 tertinggi. Lebih lanjut, model LGBM juga menghasilkan nilai Root Mean Squared Logarithmic Error (RMSLE) terkecil dibandingkan dengan model lainnya. Analisis signifikansi variabel menunjukkan bahwa lima variabel yang paling berpengaruh terhadap harga jual rumah adalah **ExterQual**, **GarageCars**, **KitchenQual**, **OverallQual**, dan **GrLivArea**. Kesimpulan dari penelitian ini memberikan wawasan penting dalam memahami faktor-faktor yang memengaruhi harga jual rumah dan menyajikan pemahaman yang mendalam tentang performa model regresi yang berbeda pada dataset ini. Implikasi praktis dari temuan ini dapat membantu pemangku kepentingan, termasuk pengembang dan pemasar properti, dalam membuat keputusan yang lebih informasional dan akurat terkait harga jual properti.

Kata-kata kunci : Random Forest, XGBoost, LGBM, Neural Network, SVM, Elastic Net, Feature Permutation, ANOVA, *R-squared*, RMSLE

Daftar Isi

Daftar Gambar

Daftar Tabel

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Rumah merupakan salah satu kebutuhan primer manusia. Selain untuk memenuhi kebutuhan tempat tinggal, banyak orang membeli rumah untuk tujuan investasi. Seiring berkembangnya zaman, rumah tetap menjadi salah satu aset yang digemari oleh investor dengan profil risiko konservatif dan cenderung mengalami kenaikan dari tahun ke tahun. Menurut salah satu *developer* properti di Indonesia, PT Pudjiadi Prestige Tbk [1], kenaikan harga properti disebabkan karena adanya efek infrastruktur, serta pertambahan jumlah penduduk dan kebutuhan akan hunian yang terus bertambah, sedangkan porsi ketersediaan tempat tinggal tidak kunjung bertambah. Dalam praktik perdagangan rumah, terdapat potensi kesalahan dalam menetapkan harga rumah yang dilakukan oleh pihak yang bersangkutan (*developer* maupun pembeli). Kesalahan dalam menentukan harga rumah dapat disebabkan karena salah satu pihak mengutamakan kepentingan pribadi, tidak memiliki informasi yang cukup, serta tidak menyesuaikan dengan kondisi rumah.

Seiring kemajuan ilmu pengetahuan dan teknologi (IPTEK), berbagai metode pembelajaran mesin (*machine learning*) dapat digunakan untuk memprediksi harga rumah berdasarkan spesifikasi rumah. Pada penelitian ini, akan digunakan beberapa metode pembelajaran mesin seperti *Random Forest*, *eXtreme Gradient Boosting*, *Light Gradient Boosting Machine*, *Neural Network*, *Support Vector Machine*, serta *Elastic Net Regression* untuk memprediksi harga rumah. Dari seluruh metode pembelajaran mesin yang digunakan, akan dicari metode yang paling baik untuk memprediksi harga rumah berdasarkan spesifikasinya. Selain mencari model pembelajaran mesin terbaik, penelitian ini bertujuan untuk menentukan spesifikasi rumah apa yang paling memengaruhi harga jual rumah. Dengan mengetahui spesifikasi yang paling memengaruhi harga jual rumah, diharapkan dapat membantu developer untuk menentukan skala prioritas yang harus ditonjolkan saat membangun rumah sehingga biaya pembangunan tidak membengkak dan rumah dapat dijual dengan harga yang terjangkau.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah:

1. Apa metode pembelajaran mesin yang paling baik untuk memprediksi harga jual rumah ?
2. Apa saja spesifikasi rumah yang paling memengaruhi harga jual rumah ?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan, tujuan dari penelitian ini adalah:

1. Mengetahui metode pembelajaran mesin yang paling baik untuk memprediksi harga jual rumah.
2. Mengetahui spesifikasi rumah yang paling memengaruhi harga jual rumah.

[1] PT Pudjiadi Prestige. "Ini Dia Alasan Mengapa Harga Properti Terus Naik Tap Tahun". (2020). https://pudjiadiprestige.co.id/news/ini-dia-alasan-mengapa-harga-properti-terus-naik-tiap-tahun.html, diakses tanggal 21 Januari 2024.

BAB 2

LANDASAN TEORI

Pada bab ini, akan dibahas mengenai teori dari tiga metode yang digunakan dalam pra-pengolahan data seperti algoritma MICE, koefisien korelasi *Pearson*, uji ANOVA (*Analysis of Variance*), model pembelajaran mesin yang akan dibangun yaitu *Random Forest*, *XGBoost*, *LGBM*, *Neural Network*, *SVM*, *Elastic Net Regression*, metode tambahan untuk mengetahui variabel bebas paling penting pada model yaitu *Feature Permutation Importance*, serta cara untuk mengevaluasi model dengan RMSE (*Root Mean Squared Error*) dan nilai R^2.

2.1 Pra-Pengolahan Data

2.1.1 Algoritma MICE

Proses imputasi MICE LGBM dimulai dengan memilih banyaknya iterasi k. Selanjutnya, setiap nilai NA pada masing-masing kolom diisi oleh nilai sementara yang dipilih secara *random* dari kolom yang bersesuaian. Untuk suatu kolom p, hilangkan lagi nilai sementara yang sudah diisi dan bangun model LightGBM dengan variabel terikat p dan variabel bebas $\sim p$. Model dilatih dengan observasi-observasi yang tidak memiliki nilai NA pada kolom p. Selanjutnya, model digunakan untuk memprediksi nilai kosong NA pada kolom p. Proses tersebut dilakukan untuk setiap kolom dengan nilai NA pada dataset hingga iterasi pertama selesai. Nilai awal yang dipilih secara *random* akan digantikan oleh prediksi model pada setiap iterasinya sehingga prediksi akan menjadi semakin baik setiap iterasinya.

2.1.2 Koefisien Korelasi *Pearson*

Koefisien korelasi *Pearson* adalah nilai penentu seberapa kuat relasi antara dua variabel numerik. Formula koefisien korelasi *Pearson* (r) diberikan pada persamaan (2.1).

$$r = \frac{n\sum X_i Y_i - \sum X_i \sum Y_i}{\sqrt{n\sum X_i^2 - (\sum X_i)^2}\sqrt{n\sum Y_i^2 - (\sum Y_i)^2}} \tag{2.1}$$

dengan n adalah jumlah pasangan (X, Y), X adalah nilai variabel X (variabel bebas), dan Y adalah nilai variabel Y (variabel terikat). Nilai $|r|$ yang mendekati 1 menunjukkan kedua variabel memiliki korelasi kuat, sedangkan nilai r yang mendekati 0 menunjukkan kedua variabel memiliki korelasi lemah.

2.1.3 Uji ANOVA (*Analysis of Variance*)

ANOVA merupakan alat analisis untuk menguji hipotesis penelitian yang menilai apakah ada perbedaan rata-rata antar kelompok. Secara spesifik, jenis ANOVA yang digunakan adalah *one-way ANOVA*. *One-way ANOVA* adalah jenis ANOVA yang digunakan apabila yang akan dianalisis terdiri dari satu variabel bebas dan satu variabel terikat. Pada uji ANOVA, terdapat dua hipotesis yang akan diuji yaitu

H_0 : Tidak ada signifikan di antara rata-rata kelompok
H_1 : Terdapat perbedaan signifikan di antara rata-rata kelompok

Kemudian, akan digunakan distribusi F untuk menghitung *p-value*. Jika *p-value* lebih kecil dari tingkat signifikansi (α) 0,05, maka H_0 ditolak yang berarti ada perbedaan signifikan di antara rata-rata kelompok.

2.2 Model

2.2.1 *Random Forest*

Random forest merupakan metode pembelajaran mesin yang merupakan pengembangan dari metode CART (*Classification and Regression Trees*). Data *house price* yang digunakan pada proyek ini menggunakan variabel terikat yang bersifat numerik maka akan digunakan metode pohon regresi. Pohon regresi merupakan model berbentuk pohon yang berfungsi untuk memprediksi variabel terikat jenis numerik kontinu dengan menggunakan nilai rata-rata (mean). Pohon regresi merupakan algoritma yang sering digunakan karena proses pembangunannya yang cepat serta hasil model yang dibangun mudah untuk diinterpretasikan. Kelemahan dari model pohon regresi adalah memiliki tingkat akurasi yang tidak terlalu tinggi karena hanya melakukan prediksi berdasarkan kelompok, serta bersifat sensitif terhadap perubahan data. Kelemahan ini yang hendak diperbaiki oleh metode *Random Forest*.

Pada dasarnya, *Random Forest* adalah kumpulan metode pohon regresi yang dibangun dengan sampel acak sehingga membentuk hutan (*forest*). Dengan menggunakan metode *Random Forest* diharapkan dapat menjaga akurasi tetap baik dengan jumlah data yang besar dengan variabel bebas yang banyak dan meminimalkan eror untuk mengurangi *overfitting*. Hasil akhir dari metode *Random Forest* berbentuk simulasi numerik yaitu nilai *mean squared error* (MSE). Dalam konteks penelitian ini, metode *Random Forest* memiliki banyak pohon dengan setiap cabang pohon merupakan variabel yang terpilih karena memiliki kemampuan yang baik untuk memprediksi. Variabel bebas yang dijadikan cabang pada pohon dapat dipertimbangkan sebagai faktor-faktor yang penting dalam menentukan harga jual rumah. Dalam pembangunan model *Random Forest* terdapat beberapa parameter yang digunakan yaitu

1. Jumlah pohon yang dibuat
   Semakin banyak pohon, semakin baik kemampuan metode *Random Forest* untuk mengatasi *overfitting* dan menghasilkan prediksi yang lebih stabil. Namun, semakin banyak pohon juga meningkatkan waktu pelatihan model.

2. Kedalaman pohon
   Semakin dalam pohon, metode dapat memahami hubungan yang lebih kompleks antara variabel bebas dan variabel terikat dalam data pelatihan. Namun, semakin dalam pohon juga dapat menyebabkan *overfitting*.

3. Jumlah sampel minimum yang diperlukan untuk membagi suatu cabang dalam pohon
   Semakin banyak jumlah sampel minimum yang diperlukan untuk membagi suatu cabang dalam pohon , semakin sedikit pembagian cabang terjadi, hal ini membantu menghindari *overfitting* dengan memaksa model untuk mempertimbangkan lebih banyak sampel sebelum membagi.

4. Jumlah sampel minimum yang dibutuhkan untuk membentuk daun
   Semakin banyak jumlah sampel minimum yang dibutuhkan untuk membentuk daun membantu menghasilkan model yang lebih konservatif dan mencegah *overfitting* pada sampel kecil.

2.2.2 *Gradient Boosting Machine* (GBM)

Gradient Boosting Machine (GBM) merupakan metode pembelajaran mesin yang memanfaatkan *ensemble learning*, di mana beberapa model lemah digabungkan untuk membentuk model yang lebih kuat. Model pembelajar lemah (*weak learner*) merupakan model yang memiliki akurasi lebih baik dibandingkan dengan tebakan acak, sedangkan model pembelajar kuat (*strong learner*) merupakan model yang memiliki akurasi prediksi yang tinggi. Pada GBM, model dasar (*base learner*) yang digunakan adalah pohon keputusan, dimana GBM akan membangun model prediksi aditif dengan menggunakan model dasar secara iteratif [3]. *Gradient Boosting Machine* membangun model secara berurutan, di mana setiap model berusaha untuk memperbaiki kesalahan prediksi model sebelumnya.

XGBoost

XGBoost (*eXtreme Gradient Boosting*) adalah pengembangan dari metode GBM yang dikembangkan oleh (Chen,2016) [5]. XGBoost memiliki kelebihan waktu komputasi yang cepat serta memiliki kinerja yang baik dibandingkan metode pembelajaran mesin lainnya. Faktor utama yang membuat XGBoost unggul dari metode pembelajaran mesin lainnya adalah skalabilitas pada semua skenario yang dipengaruhi oleh peningkatan algoritma dan pengoptimalan sistem. Algoritma yang ditingkatkan oleh *XGBoost*

adalah *split finding algorithm* yang berfungsi untuk mencari split terbaik. Faktor lain yang membuat *XGBoost* unggul adalah kemampuannya menangani nilai yang hilang secara otomatis dan proses pelatihannya yang dapat berjalan secara paralel membuatnya sangat efisien terutama pada dataset yang besar. Dalam pembangunan model *XGBoost* terdapat beberapa hal yang harus dispesifikasi diantaranya

1. Fungsi objektif
   Fungsi objektif adalah fungsi yang akan dioptimalkan selama pelatihan model. Pada penelitian ini, model digunakan untuk tugas regresi sehingga fungsi objektif yang akan digunakan adalah *squared loss function* yang terdapat pada persamaan (2.2).

$$l(y_i, \hat{y}_i) = \frac{1}{2}(y_i - \hat{y}_i)^2 \tag{2.2}$$

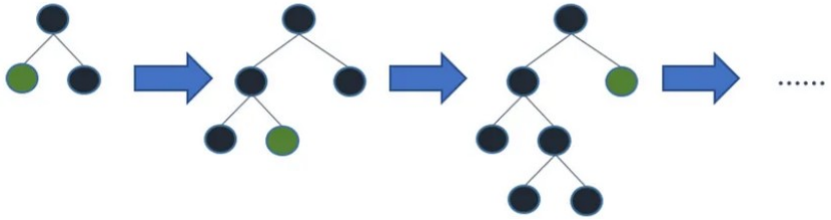
   dengan y_i merupakan nilai sebenarnya dan $\hat{y}_i$ merupakan nilai prediksi.

2. Tingkat pembelajaran *learning rate*
   Tingkat pembelajaran (*learning rate*) menentukan seberapa besar langkah model yang akan diambil selama proses pembelajaran. Tingkat pembelajaran yang rendah memerlukan lebih banyak iterasi untuk konvergensi, tetapi bisa menghasilkan model yang lebih stabil. Sebaliknya, tingkat pembelajaran yang tinggi dapat menyebabkan *overshooting*.

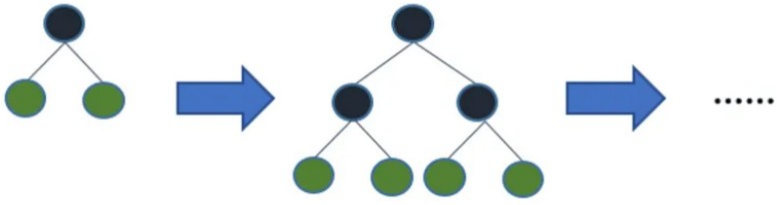
Selain fungsi objektif dan tingkat pembelajaran, kedalaman pohon dan jumlah sampel minimum di setiap daun juga harus dispesifikasi.

Light Gradient Boosting Machine (LGBM)

Pada bulan April 2017, Microsoft memperkenalkan LGBM yang merupakan sebuah *library* gradien boosting baru [4]. LGBM merupakan *gradient boosting* yang dirancang untuk menangani dataset berukuran besar secara efisien. LGBM memiliki waktu komputasi lebih cepat dibandingkan dengan *XGBoost*. Hal ini disebabkan karena LGBM menggunakan metode *one-sided sampling* berbasis gradien untuk membagi pohon sehingga mengurangi penggunaan memori dan meningkatkan akurasi. LGBM juga menggunakan pertumbuhan *leaf-wise* daripada *level-wise* seperti yang diilustrasikan pada Gambar 2.1 dan Gambar 2.2.



Gambar 2.1: Pertumbuhan *Leaf-Wise* LGBM



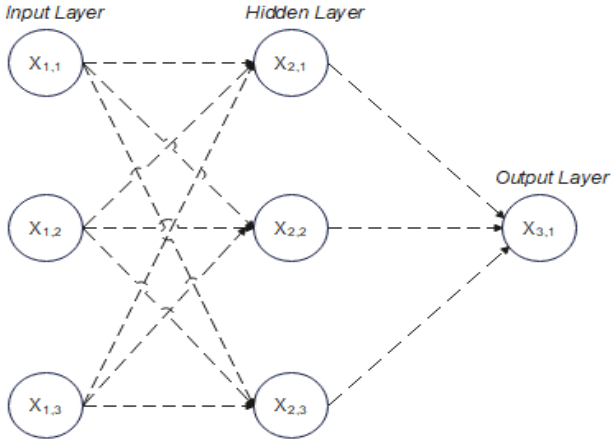
Gambar 2.2: Pertumbuhan *Level-Wise* Algoritma *Boosting* yang Lain

Dalam pembangunan model LGBM terdapat beberapa hal yang harus dispesifikasi yaitu fungsi objektif, metrik evaluasi, jenis algoritma *boosting*, banyak daun dalam setiap pohon, tingkat pembelajaran, serta *feature fraction* yang menentukan porsi fitur yang akan digunakan dalam setiap iterasi. Misal *feature fraction* bernilai 0,9 memiliki arti bahwa 90% dari fitur akan dipilih secara acak untuk setiap iterasi. Hal ini dapat membantu mengurangi *overfitting* dan meningkatkan kecepatan pelatihan.

2.2.3 Neural Network (NN)

Neural Network merupakan suatu model pembelajaran yang terinspirasi dari jaringan sel otak [6]. Tujuan utama dari *Neural Network* adalah mempelajari hubungan dan pola dalam data dengan cara mengoptimalkan parameter bobot dan bias antara neuron. Dalam proses pelatihannya yang dilakukan secara iteratif, *Neural Network* akan terus melakukan penyesuaian terhadap besarnya bobot dan bias guna memperoleh prediksi dengan tingkat akurasi yang optimal.

Neural Network terdiri dari tiga jenis lapisan yaitu lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*). Setiap lapisan terdiri dari beberapa neuron. Neuron merupakan komponen *Neural Network* yang memiliki fungsi sebagai tempat penyimpanan data. *Input layer* adalah lapisan yang berfungsi untuk membaca data dan terletak di posisi paling awal dari rangkaian *neural network*. *Hidden layer* adalah lapisan yang mengestrak pola-pola data yang digunakan. Pada *Neural Network*, semakin banyak *hidden layer* yang digunakan, maka semakin baik hasil prediksi yang diperoleh. Lapisan yang terakhir yaitu *output layer* merupakan lapisan yang mengeluarkan hasil akhir dari suatu proses dalam *Neural Network*. Untuk memberikan gambaran lebih lanjut mengenai *Neural Network*, berikut ini adalah contoh bentuk *Neural Network* dengan tiga lapisan.



Gambar 2.3: *Neural Network* dengan Tiga Lapisan

Dalam pembangunan model *neural network* terdapat beberapa hal yang harus dispesifikasi diantaranya

1. Fungsi aktivasi
   Fungsi aktivasi merupakan fungsi non-linear yang menerima *output* dari lapisan sebelumnya, *output* tersebut akan ditransformasikan kemudian dikirimkan sebagai *input* ke neuron-neuron pada lapisan berikutnya. Secara spesifik, pada penelitian ini, fungsi aktivasi yang akan digunakan adalah fungsi ReLu (*Rectified Linear Unit*). Fungsi aktivasi ReLu dapat didefinisikan pada persamaan (2.3).

$$f(z) = max(0, z) \tag{2.3}$$

2. Fungsi biaya (*loss*)
   Fungsi biaya merupakan fungsi yang digunakan untuk mengukur seberapa baik atau buruknya prediksi model *neural network* terhadap data yang sebenarnya. Tujuannya adalah untuk menemukan parameter model yang menghasilkan nilai prediksi yang paling dekat dengan nilai yang sebenarnya. Secara spesifik, pada penelitian ini, fungsi biaya yang dipakai adalah *mean squared error*. Fungsi *mean squared error* diberikan pada persamaan (2.4).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.4}$$

   dengan N adalah jumlah data, y_i adalah nilai data ke-i, dan $\tilde{y}_i$ adalah nilai prediksi untuk data ke-i.

3. *Optimizer*
   Optimizer merupakan metode yang digunakan untuk menyesuaikan nilai bobot dan bias pada setiap iterasi dengan tujuan untuk meminimalkan nilai fungsi biaya. Secara spesifik, pada penelitian ini, *optimizer* yang akan digunakan adalah "Adam" (*Adaptive Moment Estimation*). *Optimizer* Adam adalah gabungan dari dua *optimizer* yang dikembangkan dari Momentum dan RMSProp (*Root Mean Square Propagation*).

2.2.4 *Support Vector Machine* (SVM)

Support vector machine merupakan salah satu metode *supervised learning* yang dapat digunakan untuk menyelesaikan masalah regresi (*support vector regression*) [1]. SVM dapat mengatasi masalah regresi linear maupun non linear. Ide utama dari SVM adalah untuk mencari *hyperplane* terbaik dengan memaksimalkan jarak antar kelas. *Hyperplane* adalah sebuah fungsi yang dapat digunakan untuk memisahkan antar kelas. Dalam praktiknya, dua buah kelas tidak selalu terpisah secara sempurna. Untuk mengatasi masalah ini, dibutuhkan kernel untuk mentransformasikan data ke ruang dimensi yang lebih tinggi yang disebut ruang kernel. Pada penelitian ini, fungsi kernel yang digunakan adalah kernel linear dengan formula pada persamaan (2.5).

$$K(x, x') = x \cdot x' \tag{2.5}$$

2.2.5 *Elastic Net Regression*

Elastic Net Regression pertama kali diperkenalkan oleh Zou dan Hastie pada tahun 2005 [2]. *Elastic Net Regression* merupakan algoritma regresi linear yang menambahkan dua penalti ke fungsi objektif kuadrat terkecil (*least-squares*). Dua penalti ini disebut norm L1 dan L2 dari vektor koefisien dimana keduanya dikalikan dengan dua hiperparameter yaitu α dan λ. Norm L1 digunakan untuk melakukan seleksi fitur, sementara norm L2 digunakan untuk melakukan penyusutan fitur. Model *elastic net regression* diberikan pada persamaan (2.6).

$$y = b_0 + b_1 x_1 + b_2 + ... + b_n x_n + e \tag{2.6}$$

dengan y merupakan variabel terikat, b_0 adalah intersep, $b_1, ..., b_n$ adalah koefisien regresi, dan e merupakan eror. *Elastic net regression* ingin meminimalkan fungsi objektif pada persamaan (2.7).

$$RSS + \lambda[(1-\lambda)||\beta||_2 + \alpha||\beta||_1] \tag{2.7}$$

dengan RSS adalah *residual sum of squares*, λ adalah parameter regularisasi, β adalah vektor koefisien, α adalah parameter penggabungan antara norm L1 dan L2, $||\beta||_2$ adalah norm L2 dari β, dan $||\beta||_1$ adalah norm L1 dari β. Keunggulan utama dari *elastic net regression* adalah kemampuannya dalam hal seleksi fitur karena mampu menyusutkan koefisien variabel yang tidak relevan ke nol akibatnya akan menghasilkan model dengan variabel yang lebih sedikit sehingga lebih mudah diinterpretasi dan tidak rawan *overfitting*.

2.3 *Feature Permutation Importance*

Feature Permutation Importance adalah metode untuk mengetahui pengaruh suatu variabel bebas dengan cara mengacak nilai pada kolom variabel bebas tersebut untuk kemudian diamati dan dibandingkan kinerja modelnya sebelum dan sesudah diacak. Berikut ini adalah langkah-langkah dari metode *Feature Permutation Importance*:

1. Ambil model yang telah dilatih dengan data latih.
2. Hitung kinerja model pada dataset asli.
3. Untuk setiap variabel bebas j :
   i. Acak nilai pada kolom variabel bebas j pada dataset asli.
   ii. Hitung kinerja model pada dataset yang nilai kolom pada variabel bebas j nya telah diacak.
   iii. Hitung *feature permutation importance* = kinerja awal model - kinerja model setelah suatu variabel bebas diacak.
4. Ulangi langkah 3i-iii sebanyak n kali lalu hitung rata-rata dari *feature permutation importance*.

2.4 Evaluasi Model

Pada penelitian ini, terdapat dua metrik evaluasi model yang digunakan yaitu *Root Mean Squared Error* dan nilai R^2. Formula menghitung RMSE diberikan pada persamaan (2.8).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.8}$$

Formula menghitung nilai R^2 diberikan pada persamaan (2.9).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}\left(Y_i - \bar{Y}_i\right)^2} \tag{2.9}$$

dengan y_i adalah nilai sebenarnya dari observasi ke-i, $\hat{y}_i$ adalah nilai prediksi model untuk observasi ke-i, $\bar{y}_i$ adalah nilai rata-rata untuk observasi ke-i, dan n adalah jumlah observasi dalam data. Model dikatakan baik apabila memiliki nilai RMSE yang kecil serta nilai R^2 yang mendekati 1.

BAB 3

METODOLOGI

Pada bagian ini akan dibahas mengenai langkah-langkah atau metodologi yang dilakukan dalam penelitian ini. Pertama akan dibahas mengenai proses pra pengolahan data, analisis data eksploratif, pembangunan model, serta evaluasi model.

3.1 Pra Pengolahan Data

Data yang digunakan dalam penelitian ini merupakan dataset yang berisi spesifikasi dan harga jual rumah di Ames, Iowa, Amerika Serikat yang tersedia pada Kaggle[1]. Tiga baris paling atas dan bawah dari dataset yang digunakan ditampilkan pada Tabel 3.1. Dataset yang digunakan terdiri dari 1460 baris yang merepresentasikan rumah. Pada dataset ini, terdapat 80 kolom variabel ditambah satu kolom 'Id'. 80 variabel tersebut dapat dikelompokkan menjadi 34 variabel numerik dan 46 variabel kategorik.

Tabel 3.1: Tampilan Data

Id	MSSubClass	MSZoning	...	SaleType	SaleCondition	SalePrice
1	60	RL	$\cdots$	WD	Normal	208500
2	20	RL	$\cdots$	WD	Normal	181500
3	60	RL	$\cdots$	WD	Normal	223500
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
1458	70	RL	$\cdots$	WD	Normal	266500
1459	20	RL	$\cdots$	WD	Normal	142125
1460	20	RL	$\cdots$	WD	Normal	147500

Langkah pertama dalam data *pre-processing* adalah mengindentifikasi kolom dengan observasi **NA**. Observasi NA sendiri merupakan singkatan dari "*Not Available*" atau "*Not Applicable*." Dalam konteks analisis data, NA sering digunakan untuk menunjukkan bahwa nilai tidak tersedia atau tidak berlaku untuk suatu observasi atau variabel tertentu. Dalam dataset, terdapat 19 kolom yang memiliki nilai NA. Akan tetapi, berdasarkan deskripsi data pada Kaggle, 14 dari 19 kolom tersebut merupakan variabel kategorik dengan NA sebagai salah satu kategorinya. Dengan demikian, nilai NA pada 14 variabel tersebut dibentuk menjadi salah satu kategori pada variabel yang bersesuaian.

Setelah itu, tersisa 5 kolom yang masih memiliki nilai NA seperti yang ditunjukkan pada Tabel 3.2. Kolom "**GarageYrBlt**" memiliki banyaknya nilai NA yang sama seperti kolom "**GarageType**", "**GarageFinish**", dan "**GarageQual**" yang merupakan variabel kategorik dengan salah satu kategorinya adalah NA yang berarti tidak memiliki garasi. Terlebih lagi, nilai NA tersebut berada pada observasi yang sama juga. Dengan demikian, dapat diduga bahwa nilai NA pada kolom "**GarageYrBlt**" juga merepresentasikan rumah tidak memiliki garasi.

Tabel 3.2: Kolom-Kolom pada Dataset yang Memiliki Nilai NA

No.	**Kolom**	**Banyak Nilai NA**
1.	LotFrontage	259
2.	MasVnrType	8
3.	MasVnrArea	8
4.	Electrical	1
5.	GarageYrBlt	81

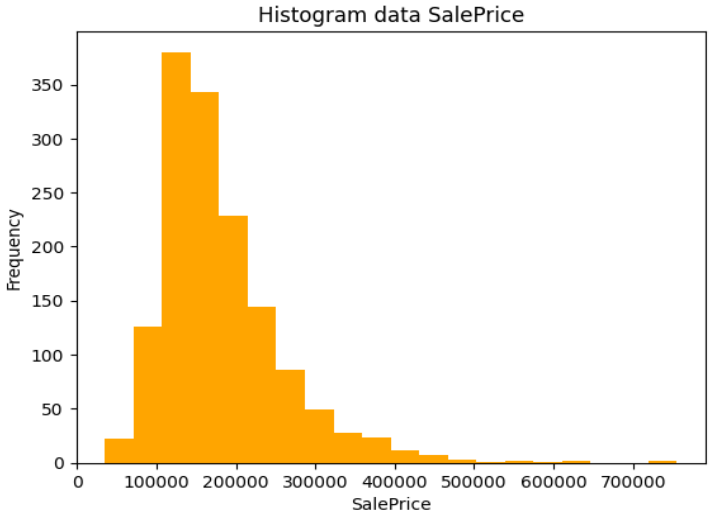
[1] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Nilai NA pada empat kolom selain "GarageYrBlt" pada Tabel 3.2 masih cukup banyak sehingga tidak bijak untuk langsung menghapus baris data. Oleh karena itu, nilai-nilai NA tersebut akan diisi menggunakan metode imputasi *Multiple Iteration Chained Equation with Light Gradient Boosting Machine* (MICE LGBM). Imputasi ini akan dilakukan secara iteratif dimana setiap hasil dari setiap iterasi berhubungan hingga membentuk rantai.

3.2 Analisis Data Eksploratif

3.2.1 Distribusi Data SalePrice

Data SalePrice memiliki rata-rata 180.921, 19589 dan standar deviasi 79.442, 503. Selanjutnya, dibentuk suatu histogram, yaitu grafik yang digunakan untuk memvisualisasikan distribusi frekuensi dari suatu data SalePrice. Histogram dapat memberikan gambaran visual mengenai sebaran data dan membantu dalam mengevaluasi apakah distribusi data mirip dengan distribusi normal atau tidak. Distribusi normal memiliki karakteristik khusus, yaitu memiliki bentuk kurva lonceng atau *bell-shaped*.



Gambar 3.1: Histogram dari Variabel SalePrice

Dapat dilihat dari Gambar 3.1, bahwa data cenderung condong ke kanan. Artinya, variabel **SalePrice** tidak memiliki distribusi normal.

3.2.2 Koefisien Korelasi *Pearson*

Koefisien korelasi Pearson adalah metrik statistik yang digunakan untuk mengukur kekuatan dan arah hubungan linier antara dua variabel. Dalam konteks analisis variabel terhadap variabel SalePrice, peneliti memilih seluruh variabel numerik terhadap variabel **SalePrice** yang lebih besar dari $0,5$. Hal ini dikarenakan nilai koefisien korelasi Pearson yang lebih besar dari $0,5$, hal ini menunjukkan adanya hubungan positif yang kuat antara variabel tersebut dan variabel **SalePrice**. Dapat dilihat pada Tabel 3.3 merupakan seluruh variabel yang nilainya $> 0,5$ jika dibandingkan dengan variabel **SalePrice**.

3.2.3 Uji ANOVA (*Analysis of Variance*)

Pada test ANOVA yang dilakukan terhadap variabel terhadap SalePrice dengan menggunakan p-value sebagai kriteria, ditemukan bahwa variabel '**Street**', '**Utilities**', dan '**LandSlope**' memiliki nilai p-value yang lebih besar dari 0,05. Hasil ini menunjukkan bahwa ketiga variabel tersebut tidak memiliki pengaruh yang signifikan terhadap perubahan **SalePrice** dalam model statistik yang digunakan. Oleh karena itu, dapat dipertimbangkan untuk tidak menyertakan variabel '**Street**', '**Utilities**', dan '**LandSlope**' dalam analisis atau model lebih lanjut, sehingga fokus dapat difokuskan pada variabel-variabel lain yang memiliki pengaruh yang lebih besar terhadap **SalePrice**.

Tabel 3.3: Korelasi Antara Variabel Numerik dengan **SalePrice**

Variabel	Nilai
OverallQual	0,790982
GrLivArea	0,708624
GarageCars	0,640409
GarageArea	0,623431
TotalBsmtSF	0,613581
1stFlrSF	0,605852
FullBath	0,560664
TotRmsAbvGrd	0,533723
YearBuilt	0,522897
YearRemodAdd	0,507101

3.3 Pembangunan dan Evaluasi Model

Data yang sudah melalui tahap pra-pengolahan data serta telah melalui proses pemilihan variabel bebas yang signifikan pengaruhnya terhadap variabel terikat ('SalePrice') akan dibagi menjadi 80% data latih dan 20% data uji. Data latih akan digunakan untuk membangun model dengan spesifikasi pada Tabel 3.4:

Tabel 3.4: Spesifikasi Model

Model	Spesifikasi Model
Random Forest	Jumlah pohon yang dibuat : 150 Kedalaman pohon : 8 Jumlah sampel minimum untuk membagi cabang : 2 Jumlah sampel minimum pada daun : 2
XGBoost	Jumlah pohon yang dibuat : 150 Tingkat pembelajaran : 0,2 Kedalaman pohon : 3 Jumlah sampel minimum pada daun : 3
LGBM	Jumlah daun : 15 Tingkat pembelajaran : 0,1 Feature fraction : 0,9 Bagging fraction : 0,8
Neural Network	Banyak layer : 3 (1 Input layer, 1 Hidden layer, 1 Output layer) Banyak neuron : Input layer = 128 unit, Hidden layer = 64 unit, Output layer = 1 unit
SVM	Fungsi kernel : Linear Nilai C : 1
Elastic Net	α : 1 Rasio L1 : 0,5

Pilihan parameter pada Tabel 3.4 merupakan parameter terbaik karena sudah melalui proses *tuning* parameter. Model yang dibangun akan dievaluasi kinerjanya dengan menggunakan RMSE (2.8) dan nilai R^2 (2.9). Model dengan kinerja terbaik akan dianalisa variabel bebas yang paling memengaruhi variabel terikat ('SalePrice') dengan *feature permutation importance*.

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil dan Analisis

Setelah proses pelatihan, model akan dievaluasi dengan mencari nilai RMSE dan R^2 menggunakan data uji. Selanjutnya, model juga akan digunakan untuk melakukan prediksi data SalePrice menggunakan *test data* lalu di*upload* ke https://www.kaggle.com/ untuk mendapatkan performa model dalam bentuk Root Mean Squared Logarithmic Error (RMSLE). Hasil evaluasi model dapat dilihat pada Tabel 4.1.

Tabel 4.1: Hasil RMSE, Setiap Model

Metode	Nilai RMSE	Nilai R^2	Nilai RMSLE Kaggle
LGBM	29.623,336	0,879	0,14488
XGBoost	29.822,316	0,884	0,14797
Elastic Net	30.406,715	0,879	0,15242
Random Forest	30.406,715	0,879	0,16385
SVM	77.745,633	0,212	0,34408
Neural Network	50.377,829	0,669	0,5762
Decision Tree with Bagging	32.010,946	0,866	0,16448

Berdasarkan Tabel 4.1, model LGBM dan XGBoost memiliki nilai RMSE pertama dan kedua paling rendah, yaitu $29.623,336$ dan $29.822,316$ secara berurutan. Kedua model ini memiliki performa yang cukup baik mengingat data SalePrice memiliki rata-rata $180.921,19589$ dan standar deviasi $79.442,503$. Artinya, *error* kedua model tersebut tidak besar. Selanjutnya, model LGBM dan XGBoost memiliki nilai R^2 sebesar $0,879$ dan $0,884$. Artinya, variabel-variabel bebas pada kedua model tersebut dapat menjelaskan sekitar 90% variasi data SalePrice. Berdasarkan nilai RMSLE yang diperoleh dari Kaggle, model LGBM dan XGBoost juga memiliki nilai terkecil pertama dan kedua. Dengan demikian, dapat disimpulkan bahwa di antara model-model yang dibangun, model LGBM dan XGBoost paling cocok untuk menjelaskan dan memprediksi data SalePrice.

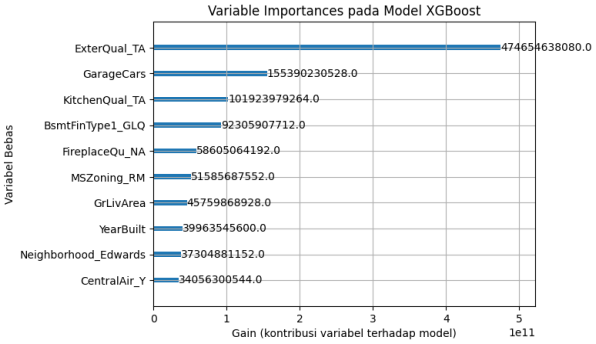
4.2 *Business Inquiries*

Untuk mengetahui variabel bebas yang signifikan pengaruhnya terhadap harga jual rumah (**SalePrice**) akan dihitung *feature permutation importance* pada dua model terbaik yaitu XGBoost dan *Light Gradient Boosting Machine* (LGBM). Hasil perhitungan *Feature Permutation Importance* terlihat pada Gambar 4.1 dan Gambar 4.2.

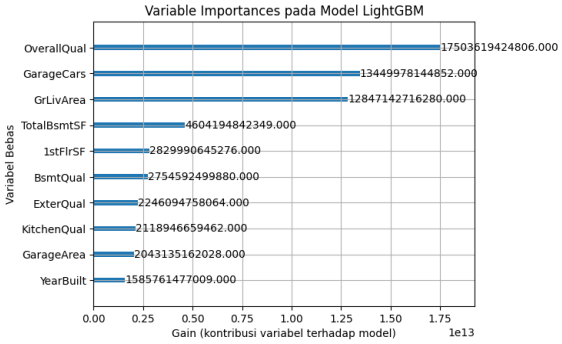
Hasil perhitungan *Feature Permutation Importance* pada kedua model menunjukkan terdapat lima variabel bebas yang paling signifikan pengaruhnya terhadap harga jual rumah yaitu:

1. ExterQual
   ExterQual merupakan variabel kategorik yang menyatakan kualitas material pada eksterior rumah. ExterQual memiliki lima kategori yaitu Ex (*Excellent*), Gd (*Good*), TA (*Average/ Typical*), Fa (*Fair*), Po (*Poor*).

2. GarageCars
   GarageCars merupakan variabel numerik yang menyatakan ukuran garasi dalam kapasitas mobil.



Gambar 4.1: Variabel Terpenting dengan Model *XGBoost*



Gambar 4.2: Variabel Terpenting dengan Model *LGBM*

3. KitchenQual
   KitchenQual merupakan variabel kategorik yang menyatakan kualitas dapur rumah. KitchenQual memiliki lima kategori yaitu Ex (*Excellent*), Gd (*Good*), TA (*Average/ Typical*), Fa (*Fair*), Po (*Poor*).

4. OverallQual
   OverallQual merupakan variabel kategorik yang menyatakan nilai skor terhadap keseluruhan material dan kualitas *finishing* rumah. OverallQual memiliki sepuluh kategori yaitu 1 (*Very Poor*), 2 (*Poor*), 3 (*Fair*), 4 (*Below Average*), 5 (*Average*), 6 (*Above Average*), 7 (*Good*), 8 (*Very Good*), 9 (*Excellent*), 10 (*Very Excellent*).

5. GrLivArea
   GrLivArea merupakan variabel numerik yang menyatakan luas ruang tinggal di atas *grade* (lantai dasar) dalam satuan *square feet*.

Lima variabel bebas diatas merupakan aspek-aspek paling penting yang harus diperhatikan oleh *developer* rumah dalam membangun rumah di Ames, Iowa, Amerika Serikat. Apabila ingin membangun rumah untuk segmentasi masyarakat menengah ke atas maka aspek yang harus ditonjolkan adalah kualitas material pada eksterior rumah yang sangat baik, ukuran garasi yang besar sehingga dapat memuat banyak mobil, kualitas dapur yang sangat baik, kualitas material dan *finishing* rumah secara keseluruhan yang sangat baik, serta memiliki ruang tinggal di atas lantai dasar yang luas.

Berdasarkan data U.S. Census Bureau[1], pendapatan median rumah tangga serta pendapatan per kapita di Ames pada tahun 2022 secara berturut-turut adalah $57.428 dan $31.050. Sementara untuk skala nasional di Amerika Serikat, keduanya memiliki nilai berturut-turut $68.700 dan $35.672. Hal ini menunjukkan daya beli masyarakat Ames yang tidak terlalu kuat karena memiliki pendapatan yang dibawah pendapatan nasional. Dengan fakta ini, bijak bagi *developer* rumah untuk membangun rumah yang layak dan nyaman huni daripada membangun rumah mewah.

Melalui pemeriksaan statistik, diperoleh informasi bahwa mayoritas kualitas material eksterior rumah, dapur, *finishing* rumah pada dataset yang diolah adalah TA (*Average/ Typical*), memiliki rata-rata kapasitas garasi untuk dua mobil, serta memiliki rata-rata luas ruang tinggal di atas lantai dasar sebesar 1.515,4637 *square feet*. Temuan ini memberikan informasi bahwa selama ini *developer* rumah di Ames telah menjual rumah yang sesuai dengan daya beli masyarakat Ames dengan memberikan spesifikasi rumah yang standar (*Average/ Typical*). Hal ini juga sekaligus memberikan tantangan bisnis bagi pihak *developer* rumah baru di Ames untuk dapat meningkatkan kualitas material eksterior rumah, dapur, *finishing* rumah menjadi Gd (*Good*), memperluas kapasitas garasi dan luas ruang tinggal tanpa menaikkan harga jual rumah secara signifikan agar tetap sesuai dengan daya beli masyarakat Ames. Hal ini dapat dilakukan dengan mengalokasikan *budget* lebih pada kualitas material eksterior rumah, dapur, *finishing* rumah, garasi, ruang tinggal dan menekan *budget* pada spesifikasi rumah yang tidak signifikan menurut model. Harapannya dengan meningkatkan kualitas dan menjaga harga jual tidak naik secara signifikan dapat memberikan *business advantage* bagi *developer* baru.

[1] https://www.census.gov/quickfacts/fact/table/amescityiowa/INC110222

BAB 5

KESIMPULAN

5.1 Kesimpulan

Dari pembahasan bab-bab sebelumnya peneliti mendapatkan kesimpulan, yaitu :

1. Nilai RMSE terbaik terdapat pada model *LGBM*, yaitu senilai $29.623,336$
2. Nilai R^2 terbaik terdapat pada model *XGBoost*, *Elastic Net*, yaitu senilai $0,884$
3. Nilai RMSLE terbaik terdapat pada model *LGBM* yaitu bernilai $0,14488$
4. Berdasarkan *business inquiries*, terdapat lima variabel yang paling signifikan pengaruhnya terhadap harga jual rumah yaitu **ExterQual**, **GarageCars**, **KitchenQual**, **OverallQual** dan **GrLivArea**.

Lampiran Kode Program

```
# -*- coding: utf-8 -*-
"""House Price (Final).ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1CJPPVTNZmO_NWmEz7WOxJOCdcW9cQGF4

# **Import library**
"""

pip install tensorflow_decision_forests

!pip install miceforest --no-cache-dir

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import miceforest as mf
from scipy.stats import f_oneway
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.inspection import permutation_importance
from xgboost import XGBRegressor
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.linear_model import ElasticNet
from sklearn.svm import SVR
from sklearn.preprocessing import LabelEncoder
import lightgbm as lgb
from sklearn.ensemble import BaggingRegressor
import tensorflow_decision_forests as tfdf

"""# **Data pre-processing**"""

# Membaca dan mengimport data csv
hp = pd.read_csv("train.csv")
hp.shape
hp.head(5)

# Membuang kolom Id
hp = hp.drop("Id", axis=1)
hp.shape

# Mendefinisikan variabel-variabel numerik dan kategorik

# Kolom-kolom kategorik
kolom_kategorik = ["MSSubClass", "MSZoning", "Street", "Alley", "LotShape", "LandContour", "Utilities", "LotConfig",
                   "LandSlope", "Neighborhood", "Condition1", "Condition2", "BldgType", "HouseStyle", "OverallQual",
                   "OverallCond", "RoofStyle", "RoofMatl", "Exterior1st", "Exterior2nd", "MasVnrType", "ExterQual",
                   "ExterCond", "Foundation", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2",
                   "Heating", "HeatingQC", "CentralAir", "Electrical", "KitchenQual", "Functional", "FireplaceQu",
                   "GarageType", "GarageFinish", "GarageQual", "GarageCond", "PavedDrive", "PoolQC", "Fence", "MiscFeature",
                   "SaleType", "SaleCondition"]

# Kolom-kolom numerik
kolom_numerik = []
for col in hp:
  if col not in kolom_kategorik:
```

```
    kolom_numerik.append(col)

print(kolom_numerik)
print("Banyaknya variabel numerik adalah ", len(kolom_numerik))

print(kolom_kategorik)
print("Banyaknya variabel kategorik adalah ", len(kolom_kategorik))

# Mengubah semua variabel kategorik ke dalam bentuk "category"
for col in kolom_kategorik:
    hp[col] = hp[col].astype("category")

# Mendefinisikan kategori-kategori untuk setiap variabel kategorik
kat_semua_var = [[20, 30, 40, 45, 50, 60, 70, 75, 80, 85, 90, 120, 150, 160, 180, 190],
    ['A', 'C (all)', 'FV', 'I', 'RH', 'RL', 'RP', 'RM'], ['Grvl', 'Pave'], ['Grvl', '
    Pave'], ['IR1', 'IR2', 'IR3', 'Reg'], ['Bnk', 'HLS', 'Low', 'Lvl'], ['AllPub', '
    NoSewr', 'NoSeWa', 'ELO'], ['Corner', 'CulDSac', 'FR2', 'FR3', 'Inside'],
                ['Gtl', 'Mod', 'Sev'], ['Blmngtn', 'Blueste', 'BrDale', 'BrkSide', '
    ClearCr', 'CollgCr','Crawfor', 'Edwards', 'Gilbert', 'IDOTRR', 'MeadowV', 'Mitchel',
    'NAmes', 'NoRidge', 'NPkVill', 'NridgHt', 'NWAmes', 'NoRidge', 'NridgHt', 'OldTown',
     'SWISU', 'Sawyer', 'SawyerW', 'Somerst', 'StoneBr', 'Timber', 'Veenker'],
                ['Artery', 'Feedr', 'Norm', 'PosA', 'PosN', 'RRAe', 'RRAn', 'RRNe', '
    RRNn'], ['Artery', 'Feedr', 'Norm', 'PosA', 'PosN', 'RRAe', 'RRAn', 'RRNe', 'RRNn'],
     ['1Fam', '2fmCon', 'Duplex', 'Twnhs', 'TwnhsE'], ['1.5Fin', '1.5Unf', '1Story', '
    2.5Fin', '2.5Unf', '2Story', 'SFoyer', 'SLvl'],
                [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], ['
    Flat', 'Gable', 'Gambrel', 'Hip', 'Mansard', 'Shed'], ['ClyTile', 'CompShg', '
    Membran', 'Metal', 'Roll', 'Tar&Grv', 'WdShake', 'WdShngl'], ['AsbShng', 'AsphShn',
    'BrkComm', 'BrkFace', 'CBlock', 'CemntBd', 'HdBoard', 'ImStucc', 'MetalSd', 'Other',
     'Plywood', 'Precast', 'Stone', 'Stucco', 'VinylSd', 'Wd Sdng', 'WdShing'],
                ['AsbShng', 'AsphShn', 'BrkComm', 'BrkFace', 'CBlock', 'CemntBd', '
    HdBoard', 'ImStucc', 'MetalSd', 'Other', 'Plywood', 'Precast', 'Stone', 'Stucco', '
    VinylSd', 'Wd Sdng', 'WdShing'], ['BrkCmn', 'BrkFace', 'CBlock', 'None', 'Stone'], [
    'Ex', 'Fa', 'Gd', 'TA', 'Po'], ['Ex', 'Fa', 'Gd', 'Po', 'TA'],
                ['BrkTil', 'CBlock', 'PConc', 'Slab', 'Stone', 'Wood'], ['Ex', 'Fa', '
    Gd', 'Po', 'TA'], ['Ex', 'Fa', 'Gd', 'Po', 'TA'], ['Av', 'Gd', 'Mn', 'No'], ['ALQ',
    'BLQ', 'GLQ', 'LwQ', 'Rec', 'Unf'], ['ALQ', 'BLQ', 'GLQ', 'LwQ', 'Rec', 'Unf'], ['
    Floor', 'GasA', 'GasW', 'Grav', 'OthW', 'Wall'],
                ['Ex', 'Fa', 'Gd', 'Po', 'TA'], ['N', 'Y'], ['FuseA', 'FuseF', 'FuseP',
     'Mix', 'SBrkr'], ['Ex', 'Fa', 'Gd', 'TA', 'Po'], ['Maj1', 'Maj2', 'Min1', 'Min2', '
    Mod', 'Sev', 'Typ', 'Sal'], ['Ex', 'Fa', 'Gd', 'Po', 'TA'], ['2Types', 'Attchd', '
    Basment', 'BuiltIn', 'CarPort', 'Detchd'], ['Fin', 'RFn', 'Unf'], ['Ex', 'Fa', 'Gd',
     'Po', 'TA'],
                ['Ex', 'Fa', 'Gd', 'Po', 'TA'], ['N', 'P', 'Y'], ['Ex', 'Fa', 'Gd', 'Ta
    '], ['GdPrv', 'GdWo', 'MnPrv', 'MnWw'], ['Gar2', 'Othr', 'Shed', 'TenC', 'Elev'], ['
    COD', 'CWD', 'Con', 'ConLD', 'ConLI', 'ConLw', 'New', 'Oth', 'WD', 'VWD'], ['Abnorml
    ', 'AdjLand', 'Alloca', 'Family', 'Normal', 'Partial']]

# Menyesuaikan kategori pada setiap variabel kategorik
for i in range(len(kolom_kategorik)):
  for level in kat_semua_var[i]:
    if level not in hp[kolom_kategorik[i]].cat.categories:
      hp[kolom_kategorik[i]] = hp[kolom_kategorik[i]].cat.add_categories(level)

# Menghitung jumlah data pada setiap kolomnya yang memiliki NA
def count_col_na(dataframe):
    col_na = dataframe.isnull().sum()
    all_col_na = []

    for i in range(len(col_na)):
        temp = []
        if col_na[i] != 0:
            temp.extend([i, hp.columns[i], col_na[i]])
            all_col_na.append(temp)

    a = all_col_na # kolom-kolom yang memiliki NA
    b = len(all_col_na) # banyaknya kolom dengan nilai NA
    return a,b

list_col_na, sum_col_na = count_col_na(hp)
print(list_col_na)
print(sum_col_na) # terdapat 19 kolom yang memiliki NA

```

```
# 14 dari 19 kolom memiliki nilai NA sebagai salah satu dari kategorinya, sehingga NA akan diubah sebagai "category"
na_as_category = ["Alley", "BsmtQual", "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "FireplaceQu",
                  "GarageType", "GarageFinish", "GarageQual", "GarageCond", "PoolQC", "Fence", "MiscFeature"]
for col in na_as_category:
    hp[col] = hp[col].cat.add_categories("NA").fillna("NA")

list_col_na2, sum_col_na2 = count_col_na(hp)
print(list_col_na2)
print(sum_col_na2) # terdapat 5 kolom tersisa dengan nilai NA

# dari list_col_na terlihat bahwa banyaknya NA pada "GarageYrBlt" sama dengan "GarageType", "GarageFinish", dan "GarageQual"
# artinya kita dapat cukup yakin bahwa observasi dengan nilai NA pada "GarageYrBlt" tidak memiliki garasi

# mengubah "GarageYrBlt" menjadi variabel kategorik dengan membuat ke dalam interval tahun
int_tahun = [0, 1900, 1905, 1910, 1915, 1920, 1925, 1930, 1935, 1940, 1945, 1950, 1955, 1960,
             1965, 1970, 1975, 1980, 1985, 1990, 1995, 2000, 2005, 2010, 2300]
hp["GarageYrBlt"] = pd.cut(hp["GarageYrBlt"], bins=int_tahun)

hp["GarageYrBlt"] = hp["GarageYrBlt"].astype("category")
hp["GarageYrBlt"] = hp["GarageYrBlt"].cat.add_categories("NA").fillna("NA")

# mengubah nama kategori
hp["GarageYrBlt"] = hp["GarageYrBlt"].cat.rename_categories(["0sampai1900", "1900sampai1905",  "1905sampai1910", "1910sampai1915", "1915sampai1920", "1920sampai1925", "1925sampai1930",
                                                             "1930sampai1935", "1935sampai1940", "1940sampai1945", "1945sampai1950", "1950sampai1955", "1955sampai1960", "1960sampai1965",
                                                             "1965sampai1970", "1970sampai1975", "1975sampai1980", "1980sampai1985", "1985sampai1990", "1990sampai1995", "1995sampai2000",
                                                             "2000sampai2005", "2005sampai2010", "2010sampai2300", "NA"])

kolom_kategorik.append("GarageYrBlt")

list_col_na3, sum_col_na3 = count_col_na(hp)
print(list_col_na3)
print(sum_col_na3) # artinya hanya ada 4 kolom yang benar2 memiliki nilai kosong NA

# menghitung jumlah baris yang memiliki NA
sum_row_na = hp.shape[0] - hp.dropna().shape[0]
percentage = sum_row_na/len(hp) * 100
print("Persentase observasi dengan nilai NA adalah {}%".format(percentage)) # terdapat 18,22% observasi yang memiliki NA (cukup banyak sehingga tidak baik untuk langsung dihapus)

# Proses Imputation menggunakan Multiple Iteration Chained Equation (MICE) dengan model LightGBM

# Melakukan One hot encoding
hp_encoded = pd.get_dummies(hp, columns = kolom_kategorik, prefix = kolom_kategorik, drop_first = False)

# Membuat kernel
kds = mf.ImputationKernel(
  hp_encoded,
  save_all_iterations=True,
  random_state=100
)

# Jalankan algoritma MICE Light GBM untuk 10 kali iterasi
kds.mice(10)

# Dataset lengkap
hp_imputed = kds.complete_data()
```

```

# Memeriksa apakah seluruh data sudah terisi atau belum
list_col_na4, sum_col_na4 = count_col_na(hp_imputed)
print(list_col_na4)
print(sum_col_na4) # semua data observasi sudah terisi

data_onehot = hp_imputed.copy()

# Proses mengubah kembali bentuk one hot encode ke bentuk semula
col_category = hp.select_dtypes("category").columns

reverse_onehot = [] # list untuk mengelompokkan nama-nama kolom dari suatu variabel, cth : ['Street_Grvl', 'Street_Pave']

for col_name in col_category:
    temp = []
    temp = [col for col in hp_imputed.columns if col_name + "_" in col]
    reverse_onehot.append(temp)

# # Menghilangkan elemen duplikat (Garage)
# del reverse_onehot[29][6:11]

for i in range(len(reverse_onehot)):
    subset_data = hp_imputed[reverse_onehot[i]]
    hp_imputed[col_category[i]] = (subset_data.iloc[:,:] == 1).idxmax(1) # membuat kolom dengan nama semula dan datanya diisi dengan kolom bernilai 1
    hp_imputed.drop(reverse_onehot[i], axis=1, inplace=True) # membuang kolom2 one hot encode

for col in kolom_kategorik:
    hp_imputed[col] = hp_imputed[col].astype("category")
    hp_imputed[col] = hp_imputed[col].str.replace(col+"_", "")  # menghilangkan nama variabel pada data

# Menyimpan data lengkap sebagai variabel data_house_price
data_house_price = hp_imputed
data_house_price

"""# **Explanatory Data Analysis**"""

# Melihat statistik dasar dari variabel SalePrice
data_house_price["SalePrice"].describe()
# SalePrice memiliki nilai rata2 180.921,196 dan standar deviasi 79.442,503

# Melihat distribusi dari data SalePrice
n, bins, patches = plt.hist(data_house_price["SalePrice"], bins=20, color="orange")
plt.title("Histogram data SalePrice")
plt.xlabel("SalePrice")
plt.ylabel("Frequency")
plt.show()

n

# Melihat statistik dasar dari variabel GarageCars
data_house_price["GarageCars"].describe()

# Melihat statistik dasar dari variabel GrLivArea
data_house_price["GrLivArea"].describe()

# Mengetahui modus dari variabel ExterQual
modus_exter_qual = data_house_price["ExterQual"].mode()
print("Modus dari kolom ExterQual:", modus_exter_qual)

# Mengetahui modus dari variabel KitchenQual
modus_kitchen_qual = data_house_price["KitchenQual"].mode()
print("Modus dari kolom KitchenQual:", modus_kitchen_qual)

# Mengetahui modus dari variabel ExterQual
modus_overall_qual = data_house_price["OverallQual"].mode()
print("Modus dari kolom OverallQual:", modus_overall_qual)

# Mencari variabel numerik yang signifikan pengaruhnya terhadap SalePrice
# Matriks korelasi antara variabel numerik
```

```
korelasi_numerik = data_house_price.corr()
# Korelasi dengan variabel 'SalePrice'
korelasi_saleprice = korelasi_numerik['SalePrice']
# Menyortir nilai korelasi dari yang tertinggi sampai terendah
korelasi_saleprice_sorted = korelasi_saleprice.sort_values(ascending=False)
print("Korelasi antara variabel numerik dengan SalePrice (sorted):")
print(korelasi_saleprice_sorted)

# Melakukan uji ANOVA untuk setiap kolom kategorik
# Menyimpan kolom yang p-value lebih dari 0.05
kolom_p_value_lebih_dari_005 = []
for kolom in kolom_kategorik:
    kategori_unik = data_house_price[kolom].unique()
    grup_data = [data_house_price["SalePrice"][data_house_price[kolom] == kategori] for
    kategori in kategori_unik]
    # Melakukan uji ANOVA
    hasil_anova, p_value = f_oneway(*grup_data)
    # Menyimpan kolom yang p-value lebih dari 0.05
    if p_value > 0.05:
        kolom_p_value_lebih_dari_005.append(kolom)
    # Menampilkan hasil uji untuk setiap kolom kategorik
    print(f"Uji ANOVA untuk {kolom}:")
    print(f"  - Statistik Uji: {hasil_anova}")
    print(f"  - Nilai p: {p_value}")
    print("\n")
# Menampilkan variabel kategorik dengan p-value lebih dari 0.05
print("Variabel kategorik dengan p-value lebih dari 0.05:")
print(kolom_p_value_lebih_dari_005)

data_onehot

# Buat dataframe baru yang isinya variabel-variabel penting saja
# Daftar kolom yang ingin disertakan dalam variabel baru
kolom_variabel_baru = ['OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea', '
    TotalBsmtSF', '1stFlrSF',
                       'FullBath', 'TotRmsAbvGrd','YearBuilt','YearRemodAdd',
                       'MSZoning', 'Alley', 'LotShape', 'LandContour', 'LotConfig', '
    Neighborhood',
                       'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle'
    , 'RoofMatl',
                       'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual', '
    ExterCond', 'Foundation',
                       'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', '
    BsmtFinType2', 'Heating',
                       'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual', '
    Functional', 'FireplaceQu',
                       'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageQual', '
    GarageCond', 'PavedDrive',
                       'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition','
    SalePrice']
# Membuat DataFrame baru hanya dengan kolom-kolom yang dipilih
data_baru = data_house_price[kolom_variabel_baru].copy()

kolom_data_baru_onehot = []

for col in kolom_variabel_baru:
  for col_name in data_onehot:
    if col in col_name:
      if col_name not in kolom_data_baru_onehot:
        kolom_data_baru_onehot.append(col_name)

data_baru_onehot = data_onehot[kolom_data_baru_onehot]
data_baru_onehot = data_baru_onehot.drop("BsmtFullBath", axis=1)

# Menampilkan DataFrame baru
data_baru

"""# **Pembangunan Model**

Random Forest
"""

# Memisahkan variabel bebas dan variabel terikat
```

```
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi 80% data latih dan 20% data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Membuat model Random Forest
model_rf = RandomForestRegressor(random_state=42)

# Menentukan grid parameter yang akan diuji
param_grid = {
    'n_estimators': [50, 100, 150],  # Jumlah pohon dalam ensemble
    'max_depth': [2, 4, 6, 8],  # Kedalaman maksimum setiap pohon
    'min_samples_split': [2, 5, 10],  # Jumlah sampel minimum yang diperlukan untuk
    membagi node
    'min_samples_leaf': [1, 2, 4]  # Jumlah sampel minimum yang diperlukan di setiap
    leaf node
}

# Membuat objek GridSearchCV
grid_search_rf = GridSearchCV(estimator=model_rf, param_grid=param_grid,
                              scoring='neg_mean_squared_error', cv=5, n_jobs=-1)

# Melakukan pencarian parameter terbaik
grid_search_rf.fit(X_train, y_train)

# Menampilkan parameter terbaik
best_params = grid_search_rf.best_params_
print(f"Parameter Terbaik: {best_params}")

# Memprediksi nilai SalePrice pada data uji dengan model yang telah di-tune
y_pred_tuned = grid_search_rf.best_estimator_.predict(X_test)

# Mengukur performa model yang telah di-tune menggunakan RMSE
mse_tuned = mean_squared_error(y_test, y_pred_tuned)
rmse_tuned = np.sqrt(mse_tuned)
print(f"RMSE setelah tuning: {rmse_tuned}")

# Mengukur R-squared setelah tuning
r_squared_tuned = r2_score(y_test, y_pred_tuned)
print(f"R-squared setelah tuning: {r_squared_tuned}")

"""XGBoost"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi 80% data latih dan 20% data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Membuat model XGBoost Regressor
model_xgb = XGBRegressor(objective='reg:squarederror', random_state=42)

# Menentukan grid parameter yang akan diuji
param_grid = {
    'n_estimators': [50, 100, 150],  # Jumlah pohon dalam ensemble
    'learning_rate': [0.01, 0.1, 0.2],  # Tingkat pembelajaran
    'max_depth': [3, 5, 7],  # Kedalaman maksimum setiap pohon
    'min_child_weight': [1, 3, 5]  # Minimum jumlah sampel yang diperlukan di setiap
    daun
}

# Membuat objek GridSearchCV
grid_search_xgb = GridSearchCV(estimator=model_xgb, param_grid=param_grid,
                               scoring='neg_mean_squared_error', cv=5, n_jobs=-1)

# Melakukan pencarian parameter terbaik
grid_search_xgb.fit(X_train, y_train)

# Menampilkan parameter terbaik
```

```
best_params = grid_search_xgb.best_params_
print(f"Parameter Terbaik: {best_params}")

# Memprediksi nilai SalePrice pada data uji dengan model yang telah di-tune
y_pred_tuned_xgb = grid_search_xgb.best_estimator_.predict(X_test)

# Mengukur performa model yang telah di-tune menggunakan RMSE
mse_tuned = mean_squared_error(y_test, y_pred_tuned_xgb)
rmse_tuned = np.sqrt(mse_tuned)
print(f"RMSE setelah tuning: {rmse_tuned}")

# Mengukur R-squared setelah tuning
r_squared_tuned = r2_score(y_test, y_pred_tuned_xgb)
print(f"R-squared setelah tuning: {r_squared_tuned}")

"""Neural Network"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Standarisasi data menggunakan StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Membuat model Neural Network
model_nn = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)  # Output layer tanpa aktivasi untuk tugas regresi
])

# Menentukan optimizer, loss function, dan metrik evaluasi
model_nn.compile(optimizer='adam', loss='mean_squared_error', metrics=['mse'])

# Melatih model dengan data pelatihan
model_nn.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Evaluasi performa model menggunakan data pengujian
y_pred_nn = model_nn.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred_nn)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Mengukur R-squared
r_squared = r2_score(y_test, y_pred_nn)
print(f"R-squared: {r_squared}")

"""Elastic Net"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Standarisasi data menggunakan StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Membuat model Elastic Net
model_elastic_net = ElasticNet()

# Menentukan grid parameter yang akan diuji
param_grid = {
```

```
    'alpha': [0.01, 0.1, 1.0],  # Parameter untuk kontrol kebijakan regularisasi total
    'l1_ratio': [0.1, 0.5, 0.9]  # Rasio campuran L1 dan L2 regularisasi
}

# Membuat objek GridSearchCV
grid_search_en = GridSearchCV(estimator=model_elastic_net, param_grid=param_grid,
                              scoring='neg_mean_squared_error', cv=5)

# Melakukan pencarian parameter terbaik
grid_search_en.fit(X_train_scaled, y_train)

# Menampilkan parameter terbaik
best_params = grid_search_en.best_params_
print(f"Parameter Terbaik: {best_params}")

# Memprediksi nilai SalePrice pada data uji dengan model yang telah di-tune
y_pred_tuned_en = grid_search_en.best_estimator_.predict(X_test_scaled)

# Mengukur performa model yang telah di-tune menggunakan RMSE
mse_tuned = mean_squared_error(y_test, y_pred_tuned)
rmse_tuned = np.sqrt(mse_tuned)
print(f"RMSE setelah tuning: {rmse_tuned}")

# Mengukur R-squared setelah tuning
r_squared_tuned = r2_score(y_test, y_pred_tuned)
print(f"R-squared setelah tuning: {r_squared_tuned}")

"""SVM"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
    =42)

# Standarisasi data menggunakan StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Membuat model Support Vector Machine Regressor
model_svm = SVR(kernel='linear', C=1.0)

# Melatih model dengan data pelatihan
model_svm.fit(X_train_scaled, y_train)

# Memprediksi nilai SalePrice pada data uji
y_pred_svm = model_svm.predict(X_test_scaled)

# Mengukur performa model menggunakan Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_svm)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Mengukur R-squared
r_squared = r2_score(y_test, y_pred_svm)
print(f"R-squared: {r_squared}")

"""LGBM"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru['SalePrice']  # Variabel terikat

# Label Encoding untuk kolom-kolom kategorikal
label_encoder = LabelEncoder()
for column in X.select_dtypes(include=['object']).columns:
    X[column] = X[column].astype(str)  # Mengubah semua nilai menjadi string
    X[column] = label_encoder.fit_transform(X[column])

# Membagi data menjadi data latih & data uji
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Membuat dataset LightGBM
train_data = lgb.Dataset(X_train, label=y_train)
test_data = lgb.Dataset(X_test, label=y_test, reference=train_data)

# Membuat parameter LightGBM
params = {
    'objective': 'regression',
    'metric': 'rmse',
    'boosting_type': 'gbdt',
    'num_leaves': 31,
    'learning_rate': 0.05,
    'feature_fraction': 0.9
}

# Menentukan grid parameter yang akan diuji
param_grid = {
    'num_leaves': [15, 31, 50],
    'learning_rate': [0.01, 0.05, 0.1],
    'feature_fraction': [0.8, 0.9, 1.0],
    'bagging_fraction': [0.8, 0.9, 1.0]
}

# Membuat objek GridSearchCV
grid_search_lgb = GridSearchCV(estimator=lgb.LGBMRegressor(**params), param_grid=param_grid,
                               scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=-1)

# Melakukan pencarian parameter terbaik
grid_search_lgb.fit(X_train, y_train)

# Menampilkan parameter terbaik
best_params = grid_search_lgb.best_params_
print(f"Parameter Terbaik: {best_params}")

# Membuat model LightGBM dengan parameter terbaik
best_params.update(params)
bst_tuned_lgb = lgb.train(best_params, train_data, num_boost_round=1000, valid_sets=[train_data, test_data])

# Memprediksi nilai SalePrice pada set pengujian dengan model yang telah di-tune
y_pred_tuned_lgb = bst_tuned_lgb.predict(X_test, num_iteration=bst_tuned_lgb.best_iteration)

# Mengukur performa model yang telah di-tune menggunakan RMSE
mse_tuned = mean_squared_error(y_test, y_pred_tuned_lgb)
rmse_tuned = np.sqrt(mse_tuned)
print(f"RMSE setelah tuning: {rmse_tuned}")

# Mengukur R-squared setelah tuning
r_squared_tuned = r2_score(y_test, y_pred_tuned)
print(f"R-squared setelah tuning: {r_squared_tuned}")

"""Decision Tree with Bagging"""

# Memisahkan variabel bebas dan variabel terikat
X = data_baru_onehot.drop('SalePrice', axis=1)  # Variabel bebas
y = data_baru_onehot['SalePrice']  # Variabel terikat

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Membuat model DecisionTreeRegressor sebagai model dasar
base_model = DecisionTreeRegressor(random_state=42)

# Membuat model BaggingRegressor
bagging_model = BaggingRegressor(base_model, n_estimators=5, random_state=42)

# Menentukan daftar parameter yang akan diuji
param_grid = {
```

```
    'n_estimators': [5, 10, 15],  # Jumlah model dalam ensemble
    'max_samples': [0.5, 0.7, 1.0],  # Proporsi sampel untuk setiap model
    'base_estimator__max_depth': [None, 5, 10]  # Kedalaman maksimum setiap model dasar
}

# Membuat objek GridSearchCV untuk tuning parameter
grid_search_dt = GridSearchCV(bagging_model, param_grid, cv=5, scoring='
    neg_mean_squared_error', n_jobs=-1)

# Melatih model dengan tuning parameter
grid_search_dt.fit(X_train, y_train)

# Menampilkan parameter terbaik
best_params = grid_search_dt.best_params_
print(f"Best Parameters: {best_params}")

# Membuat model dengan parameter terbaik
best_bagging_model = grid_search_dt.best_estimator_

# Memprediksi nilai SalePrice pada data uji
y_pred_dt = best_bagging_model.predict(X_test)

# Mengukur performa model menggunakan Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred_dt)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Mengukur R-squared
r_squared = r2_score(y_test, y_pred_dt)
print(f"R-squared: {r_squared}")

"""# Test Data Pre-processing"""

# Test data pre-processing
test_data = pd.read_csv("test.csv")

# Membuang kolom Id
test_data = test_data.drop("Id", axis=1)

# Mengubah semua variabel kategorik ke dalam bentuk "category"
for col in kolom_kategorik[0:len(kolom_kategorik)-1]:
    test_data[col] = test_data[col].astype("category")

# Menyesuaikan kategori pada setiap variabel kategorik
for i in range(len(kolom_kategorik)-1):
  for level in kat_semua_var[i]:
    if level not in test_data[kolom_kategorik[i]].cat.categories:
      test_data[kolom_kategorik[i]] = test_data[kolom_kategorik[i]].cat.add_categories(
    level)

# Mengubah "GarageYrBlt" menjadi variabel kategorik dengan membuat ke dalam interval
    tahun
test_data["GarageYrBlt"] = pd.cut(test_data["GarageYrBlt"], bins=int_tahun)

test_data["GarageYrBlt"] = test_data["GarageYrBlt"].astype("category")
test_data["GarageYrBlt"] = test_data["GarageYrBlt"].cat.add_categories("NA").fillna("NA"
    )

# Mengubah nama kategori
test_data["GarageYrBlt"] = test_data["GarageYrBlt"].cat.rename_categories(["0sampai1900"
    , "1900sampai1905",  "1905sampai1910", "1910sampai1915", "1915sampai1920", "1920
    sampai1925", "1925sampai1930",
                                                                "1930sampai1935", "1935
    sampai1940", "1940sampai1945", "1945sampai1950", "1950sampai1955", "1955sampai1960",
     "1960sampai1965",
                                                                "1965sampai1970", "1970
    sampai1975", "1975sampai1980", "1980sampai1985", "1985sampai1990", "1990sampai1995",
     "1995sampai2000",
                                                                "2000sampai2005", "2005
    sampai2010", "2010sampai2300", "NA"])

# Memilih kolom-kolom yang akan digunakan pada test_data
kolom_variabel_baru_t = kolom_variabel_baru[0:len(kolom_variabel_baru)-1] #
```

```
    menghilangkan SalePrice

test_baru = test_data[kolom_variabel_baru_t].copy()

# Menghitung jumlah data pada setiap kolomnya yang memiliki NA
list_col_na_test, sum_col_na_test = count_col_na(test_baru)
print(list_col_na_test)
print(sum_col_na_test)

# 14 dari 24 kolom memiliki nilai NA sebagai salah satu dari kategorinya, sehingga NA akan diubah sebagai "category"
for col in na_as_category:
    test_baru[col] = test_baru[col].cat.add_categories("NA").fillna("NA")

list_col_na2_test, sum_col_na2_test = count_col_na(test_baru)
print(list_col_na2_test)
print(sum_col_na2_test) # terdapat 10 kolom tersisa dengan nilai NA

# Proses Imputation menggunakan Multiple Iteration Chained Equation (MICE) dengan model LightGBM

# Melakukan One hot encoding
kolom_kategorik_t = []
for col in kolom_variabel_baru_t:
  if col in kolom_kategorik:
    kolom_kategorik_t.append(col)

test_encoded = pd.get_dummies(test_baru, columns = kolom_kategorik_t, prefix = kolom_kategorik_t, drop_first = False)

# Membuat kernel
kds_2 = mf.ImputationKernel(
  test_encoded,
  save_all_iterations=True,
  random_state=100
)

# Jalankan algoritma MICE Light GBM untuk 10 kali iterasi
kds_2.mice(10)

# Dataset lengkap
test_imputed = kds_2.complete_data()
test_imputed_oh = test_imputed.copy()

# Memeriksa apakah seluruh data sudah terisi atau belum
imputed_na, sum_imputed_na = count_col_na(test_imputed)
print(imputed_na)
print(sum_imputed_na) # semua data observasi sudah terisi

# Proses mengubah kembali bentuk one hot encode ke bentuk semula

reverse_onehot_t = [] # list untuk mengelompokkan nama-nama kolom dari suatu variabel, cth: ['Alley_Grvl', 'Alley_Pave', 'Alley_NA']

for col_name in kolom_kategorik_t:
    temp = []
    temp = [col for col in test_imputed.columns if col_name+"_" in col]
    reverse_onehot_t.append(temp)

for i in range(len(reverse_onehot_t)):
    subset_data = test_imputed[reverse_onehot_t[i]]
    test_imputed[kolom_kategorik_t[i]] = (subset_data.iloc[:,:] == 1).idxmax(1) # membuat kolom dengan nama semula dan datanya diisi dengan kolom bernilai 1
    test_imputed.drop(reverse_onehot_t[i], axis=1, inplace=True) # membuang kolom2 one hot encode

for col in kolom_kategorik_t:
    test_imputed[col] = test_imputed[col].astype("category")
    test_imputed[col] = test_imputed[col].str.replace(col+"_", "")  # menghilangkan nama variabel pada data

# Mengurutkan kolom
kolom_urut = []
```

```
kolom_urut_oh = []

for col in data_baru.drop('SalePrice', axis=1):
  kolom_urut.append(col)

for col in data_baru_onehot.drop('SalePrice', axis=1):
  kolom_urut_oh.append(col)

test_imputed_urut_oh = test_imputed_oh[kolom_urut_oh].copy()
test_imputed_urut = test_imputed[kolom_urut].copy()

"""# Test Data Prediction"""

# Random Forest
# Melakukan prediksi untuk test data
pred_test_data_rf = grid_search_rf.best_estimator_.predict(test_imputed_urut_oh)

# Membuat dataframe dari hasil prediksi
df_pred_rf = pd.DataFrame()
df_pred_rf["Id"] = np.arange(1461, 1461+len(pred_test_data_rf),1)
df_pred_rf["SalePrice"] = pred_test_data_rf

# Mengexport ke bentuk csv
df_pred_rf.to_csv("prediction_rf.csv",index=False)

# XGBoost
# Melakukan prediksi untuk test data
pred_test_data_xgb = grid_search_xgb.best_estimator_.predict(test_imputed_urut_oh)

# Membuat dataframe dari hasil prediksi
df_pred_xgb = pd.DataFrame()
df_pred_xgb["Id"] = np.arange(1461, 1461+len(pred_test_data_xgb),1)
df_pred_xgb["SalePrice"] = pred_test_data_xgb

# Mengexport ke bentuk csv
df_pred_xgb.to_csv("prediction_xgb.csv",index=False)

# Neural Network
# Melakukan prediksi untuk test data
pred_test_data_nn = model_nn.predict(scaler.fit_transform(test_imputed_urut_oh))

# Membuat dataframe dari hasil prediksi
df_pred_nn = pd.DataFrame()
df_pred_nn["Id"] = np.arange(1461, 1461+len(pred_test_data_nn),1)
df_pred_nn["SalePrice"] = pred_test_data_nn

# Mengexport ke bentuk csv
df_pred_nn.to_csv("prediction_nn.csv",index=False)

# Elastic Net
# Melakukan prediksi untuk test data
pred_test_data_en = grid_search_en.best_estimator_.predict(scaler.fit_transform(
    test_imputed_urut_oh))

# Membuat dataframe dari hasil prediksi
df_pred_en = pd.DataFrame()
df_pred_en["Id"] = np.arange(1461, 1461+len(pred_test_data_en),1)
df_pred_en["SalePrice"] = pred_test_data_en

# Mengexport ke bentuk csv
df_pred_en.to_csv("prediction_en.csv",index=False)

# SVM
# Melakukan prediksi untuk test data
pred_test_data_svm = model_svm.predict(scaler.fit_transform(test_imputed_urut_oh))

# Membuat dataframe dari hasil prediksi
df_pred_svm = pd.DataFrame()
df_pred_svm["Id"] = np.arange(1461, 1461+len(pred_test_data_svm),1)
df_pred_svm["SalePrice"] = pred_test_data_svm
df_pred_svm

# Mengexport ke bentuk csv
```

```
779 df_pred_en.to_csv("prediction_en.csv",index=False)
780
781 # Light GBM
782 # Melakukan prediksi untuk test data
783
784 # Label Encoding untuk kolom-kolom kategorikal
785 label_encoder = LabelEncoder()
786
787 for column in test_imputed_urut.select_dtypes(include=['object']).columns:
788     test_imputed_urut[column] = test_imputed_urut[column].astype(str)  # Mengubah semua
        nilai menjadi string
789     test_imputed_urut[column] = label_encoder.fit_transform(test_imputed_urut[column])
790
791
792 pred_test_data_lgb = bst_tuned_lgb.predict(test_imputed_urut, num_iteration=
        bst_tuned_lgb.best_iteration)
793
794 # Membuat dataframe dari hasil prediksi
795 df_pred_lgb  = pd.DataFrame()
796 df_pred_lgb ["Id"] = np.arange(1461, 1461+len(pred_test_data_lgb ),1)
797 df_pred_lgb ["SalePrice"] = pred_test_data_lgb
798 df_pred_lgb
799
800 # Mengexport ke bentuk csv
801 df_pred_lgb.to_csv("prediction_en.lgb",index=False)
802
803 # Decision Tree with Bagging
804 # Melakukan prediksi untuk test data
805 pred_test_data_dt = grid_search_dt.best_estimator_.predict(test_imputed_urut_oh)
806
807 # Membuat dataframe dari hasil prediksi
808 df_pred_dt = pd.DataFrame()
809 df_pred_dt["Id"] = np.arange(1461, 1461+len(pred_test_data_xgb),1)
810 df_pred_dt["SalePrice"] = pred_test_data_xgb
811
812 # Mengexport ke bentuk excel
813 df_pred_dt.to_csv("prediction_dt.csv",index=False)
814
815 """# Variable Importance"""
816
817 # Akan dibuat plot variable importance untuk 2 model terbaik berdasarkan RMSE dan R^2,
        yaitu model XGBoost dan LightGBM
818
819 # Model XGBoost
820 model_xgb = XGBRegressor(learning_rate=0.2, max_depth=3, min_child_weight=1,
        n_estimators=150)
821 model_xgb.fit(X_train, y_train)
822
823 # Membuat plot feature importance XGBoost
824 from xgboost import plot_importance
825 plot_importance(model_xgb, importance_type="gain", max_num_features=10,
826                 title="Variable Importances pada Model XGBoost", ylabel="Variabel Bebas"
        ,
827                 xlabel="Gain (kontribusi variabel terhadap model)")
828 # Gain importance mengukur peningkatan akurasi model dengan menggunakan suatu variabel
        tertentu dalam splitting
829 # Ukuran ini juga mengukur kualitas yang diberikan oleh splits
830
831 # Membuat plot feature importance LightGBM
832 lgb.plot_importance(bst_tuned_lgb, importance_type="gain", max_num_features=10,
833                 title="Variable Importances pada Model LightGBM", ylabel="Variabel Bebas
        ",
834                 xlabel="Gain (kontribusi variabel terhadap model)")
835 # Gain importance mengukur peningkatan akurasi model dengan menggunakan suatu variabel
        tertentu dalam splitting
836 # Ukuran ini juga mengukur kualitas yang diberikan oleh splits
```

DAFTAR PUSTAKA

[1] David Barbella **andothers**. "Understanding Support Vector Machine Classifications via a Recommender System-Like Approach." **in***DMIN*: 1 (2009), **pages** 305–311.

[2] Trevor Hastie Hui Zou. "Regularization and variable selection via the elastic net". phdthesis. Stanford University, 2005.

[3] J.H.Friedman. "Stochastic Gradient Boosting". **in***Computational Statistics Data Analysis*: (2002).

[4] Haijian Shi. "Best-first decision tree learning". phdthesis. The University of Waikato, 2007.

[5] C.Guestrin T.Q. Chen. "Xgboost : A Scalable Tree Boosting System". **in***Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*: (2016).

[6] Felicia Ilona Thamara. *Memprediksi tingkat pemulihan asuransi grup cacat jangka panjang menggunakan neural network*. Indonesia, 2022.