DOKUMEN PROYEK

12S3205 - PENAMBANGAN DATA

Development of a Predictive Regression Model for House Prices Using Ensemble Stacking Techniques

Disusun Oleh:

12S22015	**Angelina Nadeak**
12S22029	**Jeremy Samosir**
12S22038	**Ade Siahaan**
12S22052	**Rosari Simanjuntak**

PROGRAM STUDI SARJANA SISTEM INFORMASI

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO (FITE)
INSTITUT TEKNOLOGI DEL
TAHUN 2024/2025

DAFTAR ISI

BAB 1 BUSINESS UNDERSTANDING

1.1 Determine Business Objective

Industri properti merupakan salah satu sektor yang sangat dinamis dan memiliki dampak signifikan terhadap perekonomian suatu negara. Harga rumah menjadi indikator utama dalam transaksi jual-beli properti, baik untuk konsumen perorangan, agen properti, maupun perusahaan pengembang real estate. Namun, penentuan harga rumah seringkali bersifat subjektif dan sangat dipengaruhi oleh faktor-faktor eksternal yang sulit diprediksi seperti kondisi pasar, lokasi, dan tren ekonomi.

Tujuan bisnis dari proyek ini adalah menyediakan sistem prediksi harga rumah berbasis machine learning yang mampu mengurangi ketidakpastian dalam proses estimasi harga. Dengan sistem prediksi ini, diharapkan stakeholder properti seperti investor, penjual, pembeli, dan agen real estate dapat mengambil keputusan yang lebih cepat dan tepat.

Manfaat yang ingin dicapai dalam jangka panjang:

- Meminimalisir kesalahan estimasi harga.
- Memberikan insight berbasis data dalam proses negosiasi properti.
- Meningkatkan efisiensi waktu dan biaya dalam menentukan harga jual/beli.
- Meningkatkan daya saing perusahaan properti melalui adopsi teknologi AI.

1.2 Determine Project Goal

Tujuan teknis dari proyek ini adalah mengembangkan model prediksi harga rumah dengan pendekatan ensemble stacking, yang menggabungkan kekuatan dari beberapa algoritma machine learning seperti XGBoost, LightGBM, dan CatBoost. Model stacking ini bertujuan memaksimalkan akurasi prediksi dan meminimalkan error, terutama dalam kondisi data yang kompleks dan beragam.

Target pengembangan model:

- Memprediksi harga rumah berdasarkan fitur properti dengan akurasi tinggi.
- Mengurangi bias prediksi yang disebabkan oleh model tunggal.
- Menghasilkan model yang stabil dan generalisasi dengan baik terhadap data baru.

1.3 Produce Project Plan

Rencana pelaksanaan proyek:

- Tahap 1: Pengumpulan dan eksplorasi data properti.
- Tahap 2: Preprocessing, feature selection dan feature engineering.
- Tahap 3: Pengembangan model ensemble stacking.
- Tahap 4: Evaluasi model menggunakan beberapa metrik evaluasi, di antaranya:
    - RMSE (Root Mean Squared Error) untuk menghitung akar rata-rata kesalahan kuadrat prediksi.
    - MAE (Mean Absolute Error) untuk mengukur rata-rata selisih absolut antara nilai prediksi dan nilai aktual.
    - R^2 (Coefficient of Determination) untuk menilai seberapa besar variasi target yang dapat dijelaskan oleh model.
- Tahap 5: Deployment model dalam bentuk aplikasi prediktif.

Waktu estimasi pengerjaan: 2 bulan Tim pelaksana: Data Scientist, Data Engineer, Business Analyst.

BAB 2 DATA UNDERSTANDING

2.1 Collecting Data

Dataset yang digunakan pada proyek ini diambil dari kompetisi "House Prices - Advanced Regression Techniques" di platform Kaggle. Dataset terdiri dari dua bagian utama, yaitu data pelatihan (train) dan data pengujian (test). File train.csv

```
train=
pd.read_csv("/content/drive/MyDrive/Dami/housepricesadvancedregres
siontechniques/train.csv")

test=
pd.read_csv("/content/drive/MyDrive/Dami/housepricesadvancedregres
siontechniques/test.csv")
```

2.2 Describe Data

Setelah pengumpulan data, proses berikutnya adalah memahami distribusi dan karakteristik data. Dataset ini mencakup:

2.2.1 Describe Data Train

Langkah selanjutnya adalah melakukan analisis deskriptif terhadap dataset pelatihan. Analisis ini dilakukan dengan menggunakan fungsi describe() dari pustaka *pandas*, yang secara otomatis menghasilkan ringkasan statistik dari seluruh fitur numerik dalam dataset.

Code:

```
train.describe()
```

Output:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SalePrice
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	...	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	...	94.244521	46.660274	21.954110	3.409589	15.060959	2.758904	43.489041	6.321918	2007.815753	180921.195890
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	...	125.338794	66.256028	61.119149	29.317331	55.757415	40.177307	496.123024	2.703626	1.328095	79442.502883
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	2006.000000	34900.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.000000	2007.000000	129975.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	...	0.000000	25.000000	0.000000	0.000000	0.000000	0.000000	0.000000	6.000000	2008.000000	163000.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	...	168.000000	68.000000	0.000000	0.000000	0.000000	0.000000	0.000000	8.000000	2009.000000	214000.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000000	547.000000	552.000000	508.000000	480.000000	738.000000	15500.000000	12.000000	2010.000000	755000.000000

8 rows × 38 columns

Interpretasi:

Fungsi describe() akan memebrikan informasi mencakup jumlah data (count), nilai rata-rata (mean), standar deviasi (std), nilai terkecil (min), nilai kuartil 1 (25%), median atau kuartil 2 (50%), kuartil 3 (75%), dan nilai terbesar (max).
Sebagian besar kolom memiliki jumlah data yang lengkap (1.460 baris), tetapi ada beberapa kolom seperti LotFrontage dan MasVnrArea yang memiliki jumlah data lebih sedikit, yang berarti terdapat data yang kosong (missing).

Harga rumah (SalePrice) memiliki nilai rata-rata sekitar 180.921 dengan harga tertinggi mencapai 755.000 dan terendah 34.900. Ini menunjukkan bahwa harga rumah di dataset sangat bervariasi. Nilai standar deviasi yang cukup besar (±79.442) juga memperkuat bahwa penyebaran harga cukup lebar. Beberapa fitur seperti LotArea, BsmtFinSF1, dan GarageArea menunjukkan adanya perbedaan besar antara nilai maksimum dan nilai kuartil atas, menandakan adanya rumah-rumah yang memiliki ukuran atau fitur jauh lebih besar dibandingkan yang lain (outlier).

Sementara itu, banyak fitur seperti PoolArea, ScreenPorch, dan 3SsnPorch memiliki nilai tengah (median) sebesar nol. Artinya, sebagian besar rumah tidak memiliki fasilitas tersebut, namun ada beberapa rumah yang memilikinya dengan ukuran yang cukup besar.

Fitur penilaian kondisi rumah seperti OverallQual (kualitas keseluruhan) dan OverallCond (kondisi keseluruhan) memiliki nilai antara 1 sampai 10. Rata-rata nilainya adalah sekitar 6, yang menunjukkan bahwa sebagian besar rumah berada dalam kondisi cukup baik.

2.2.2 Data Train Shape

Dataset terdiri dari dua bagian:

- Training set: Berisi 1.460 baris data dan 81 fitur.
- Testing set: Berisi 1.459 baris data dan 81 fitur.

Code:

```
# Ukuran data

print(f"Jumlah baris dan kolom (Train): {train.shape}")

print(f"Jumlah baris dan kolom (Test) : {test.shape}")
```

2.2.3 Fitur numerik dan kategorikal

Fitur numerik adalah fitur yang berupa angka dan dapat dihitung secara matematis. Sementara itu, fitur kategorikal merupakan fitur yang merepresentasikan kategori. Untuk mengelompokkan fitur-fitur tersebut, dilakukan seleksi tipe data menggunakan fungsi select_dtypes() dari pustaka *pandas*. Fitur bertipe int64 dan float64 dimasukkan ke dalam daftar fitur numerik, sedangkan fitur bertipe object dimasukkan ke dalam daftar fitur kategorikal.

Code:

```
numeric_features=             train.select_dtypes(include=['int64',
'float64']).columns.tolist()

categorical_features=
train.select_dtypes(include=['object']).columns.tolist()

print(f"Numerik: {numeric_features}")

print(f"Kategorikal: {categorical_features}")
```

Output:

```
Fitur Numerik:
['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
'1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold',
'SalePrice']

Fitur Kategorikal:
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
'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', 'GarageFinish', 'GarageQual', 'GarageCond', 'PavedDrive', 'PoolQC', 'Fence',
'MiscFeature', 'SaleType', 'SaleCondition']

Jumlah fitur numerik: 38
Jumlah fitur kategorikal: 43
```

Output dalam bentuk tabel:

Fitur Numerik	Fitur Kategorikal
Id	MSZoning
MSSubClass	Street
LotFrontage	Alley
LotArea	LotShape
OverallQual	LandContour
OverallCond	Utilities
YearBuilt	LotConfig
YearRemodAdd	LandSlope
MasVnrArea	Neighborhood
BsmtFinSF1	Condition1
BsmtFinSF2	Condition2
BsmtUnfSF	BldgType
TotalBsmtSF	HouseStyle
1stFlrSF	RoofStyle
2ndFlrSF	RoofMatl
LowQualFinSF	Exterior1st
GrLivArea	Exterior2nd
BsmtFullBath	MasVnrType
BsmtHalfBath	ExterQual
FullBath	ExterCond
HalfBath	Foundation
BedroomAbvGr	BsmtQual

KitchenAbvGr	BsmtCond
TotRmsAbvGrd	BsmtExposure
Fireplaces	BsmtFinType1
GarageYrBlt	BsmtFinType2
GarageCars	Heating
GarageArea	HeatingQC
WoodDeckSF	CentralAir
OpenPorchSF	Electrical
EnclosedPorch	KitchenQual
3SsnPorch	Functional
ScreenPorch	FireplaceQu
PoolArea	GarageType
MiscVal	GarageFinish
MoSold	GarageQual
YrSold	GarageCond
SalePrice	PavedDrive
	PoolQC
	Fence
	MiscFeature
	SaleType
	SaleCondition

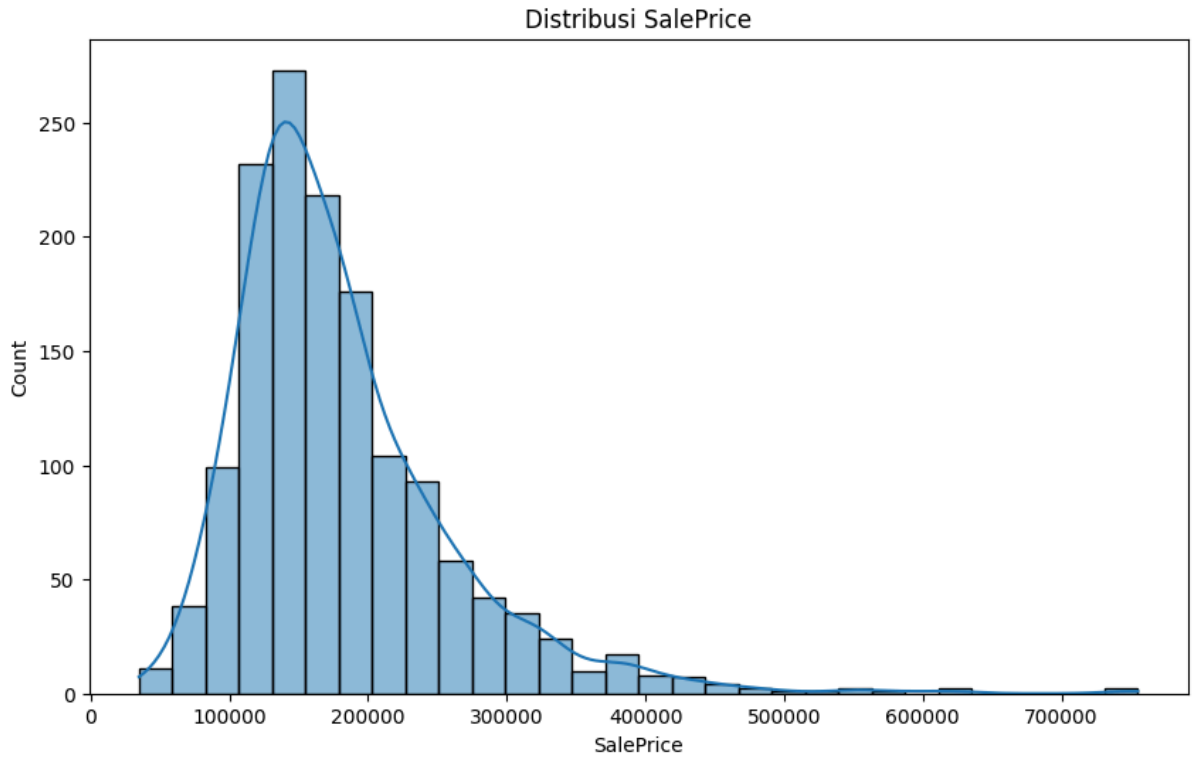
2.2.4 Distribusi Sale Price (target)

Langkah penting dalam memahami data adalah menganalisis distribusi dari variabel target, yaitu SalePrice, yang merepresentasikan harga jual rumah. Memahami distribusi harga jual dapat memberikan wawasan awal mengenai pola persebaran data, keberadaan outlier, serta apakah data bersifat normal atau tidak. Hal ini sangat berguna dalam menentukan pendekatan transformasi data maupun pemilihan algoritma yang tepat untuk model prediktif.

Code:

```
# Distribution of target

plt.figure(figsize=(10,6))

sns.histplot(train['SalePrice'], kde=True, bins=30)

plt.title('Distribusi SalePrice')

plt.show()
```

Output:



Interpretasi:

Distribusi dari variabel *SalePrice* menunjukkan pola positively skewed (skew ke kanan), yang berarti mayoritas harga rumah dalam dataset berada di kisaran harga yang relatif rendah, sementara ekornya memanjang ke arah kanan. Hal ini menunjukkan bahwa terdapat sejumlah rumah dengan harga jual yang jauh lebih tinggi dibandingkan mayoritas, yang merupakan outlier dalam data. Dari visualisasi histogram, terlihat bahwa harga jual terbanyak (mode) berada pada kisaran 120.000 hingga 150.000, yang menjadi rentang harga umum rumah dalam dataset. Namun demikian, terdapat beberapa rumah yang memiliki harga di atas 500.000 hingga mencapai 750.000, yang secara signifikan berbeda dari mayoritas data.

Pola distribusi seperti ini memiliki implikasi penting terhadap pemodelan, khususnya jika akan digunakan algoritma seperti regresi linear, yang mengasumsikan bahwa residual model tersebar normal. Oleh karena itu, transformasi logaritmik pada variabel SalePrice sering digunakan untuk menormalkan distribusi dan mengurangi pengaruh outlier yang ekstrem, sehingga dapat meningkatkan performa dan stabilitas model prediktif.

2.2.5 Correlation Heatmap Semua Fitur (Numerik + Kategorikal)

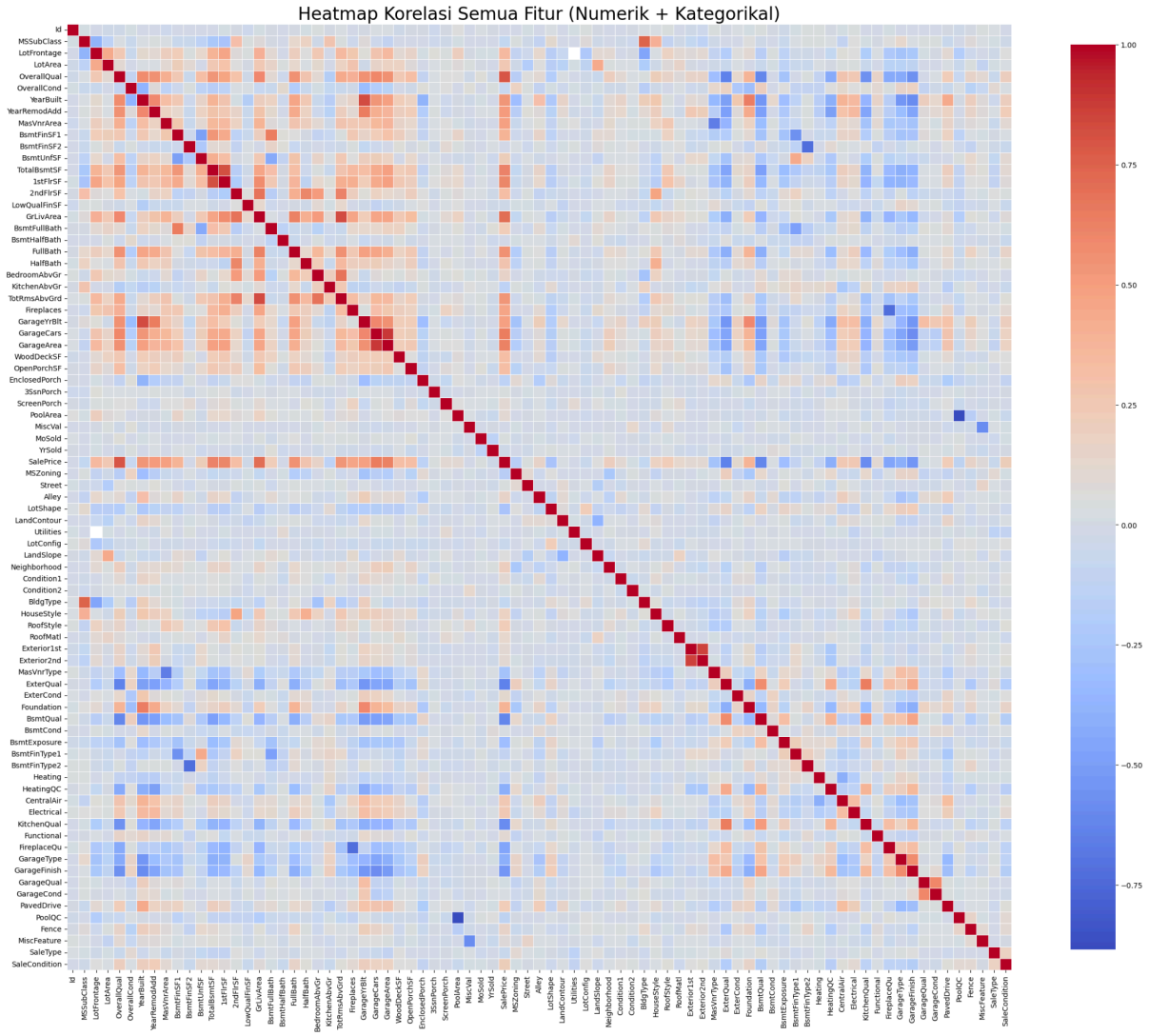
Pada tahap ini dilakukan analisis korelasi untuk seluruh fitur dalam dataset, baik numerik maupun kategorikal. Fitur kategorikal terlebih dahulu dikonversi menggunakan teknik *Label Encoding* agar dapat dihitung korelasinya. Hasilnya divisualisasikan dalam bentuk heatmap untuk memberikan gambaran hubungan antar fitur secara menyeluruh terhadap variabel target maupun antar fitur itu sendiri. Visualisasi ini membantu dalam proses pemilihan fitur yang relevan untuk membangun model prediktif.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Pisahkan fitur numerik dan kategorikal
```

```
numeric_feats = train.select_dtypes(include=['int64', 'float64'])
categorical_feats = train.select_dtypes(include=['object'])


# Label Encoding untuk fitur kategorikal
label_encoded = categorical_feats.copy()
le = LabelEncoder()
for col in label_encoded.columns:
    label_encoded[col] =
le.fit_transform(label_encoded[col].astype(str))


# Gabungkan numerik + label encoded
combined_data = pd.concat([numeric_feats, label_encoded], axis=1)


# Hitung korelasi
corr_matrix = combined_data.corr()


# Visualisasi heatmap
plt.figure(figsize=(24, 22))
sns.heatmap(
    corr_matrix,
    cmap='coolwarm',
    annot=False,
    fmt='.2f',
```

```
    linewidths=0.5,
    linecolor='white',
    square=True,
    cbar_kws={"shrink": 0.8}
)
plt.title('Heatmap Korelasi Semua Fitur (Numerik + Kategorikal)', fontsize=24)
plt.xticks(rotation=90, fontsize=10)
plt.yticks(rotation=0, fontsize=10)
plt.tight_layout()
plt.show()
```

Output:



Interpretasi:

Gambar heatmap di atas menunjukkan hubungan korelasi antar seluruh fitur dalam dataset, baik fitur numerik maupun fitur kategorikal yang telah diubah ke dalam format numerik melalui proses *Label Encoding*. Korelasi yang ditampilkan dihitung menggunakan metode Pearson, yang mengukur sejauh mana hubungan linier antara dua variabel. Warna merah pada heatmap menunjukkan korelasi positif, yaitu apabila nilai suatu variabel meningkat, maka nilai variabel lainnya juga cenderung meningkat. Sebaliknya, warna biru menunjukkan korelasi negatif, yaitu apabila nilai satu variabel meningkat, maka nilai variabel lainnya cenderung menurun. Sementara itu, warna yang mendekati putih mengindikasikan bahwa tidak terdapat hubungan linier yang kuat antara kedua variabel tersebut.

Dari hasil visualisasi ini, dapat diamati bahwa beberapa fitur memiliki korelasi yang cukup kuat satu sama lain. Misalnya, fitur seperti *OverallQual*, *GrLivArea*, dan *GarageCars* tampak memiliki hubungan positif yang cukup kuat terhadap variabel *SalePrice*. Hal ini menunjukkan bahwa rumah dengan kualitas keseluruhan yang lebih baik, luas area bangunan yang lebih besar, dan kapasitas garasi yang memadai cenderung memiliki harga jual yang lebih tinggi. Di sisi lain, terdapat juga fitur-fitur yang memiliki korelasi negatif terhadap *SalePrice*, yang menunjukkan bahwa peningkatan nilai pada fitur-fitur tersebut justru berpotensi menurunkan harga jual rumah.

Secara keseluruhan, heatmap ini sangat berguna dalam proses eksplorasi data, khususnya untuk memahami hubungan antar fitur dan membantu dalam pemilihan fitur yang relevan. Fitur-fitur dengan korelasi yang sangat tinggi satu sama lain dapat dipertimbangkan untuk direduksi guna menghindari masalah multikolinearitas dalam proses pemodelan selanjutnya. Sementara itu, fitur yang menunjukkan korelasi kuat terhadap variabel target dapat dijadikan fokus utama dalam pengembangan model prediksi harga rumah.

2.3 Validation Data

Pengecekan kualiatas data dan konsistendinya dengan melihat missing values pada data:
code:

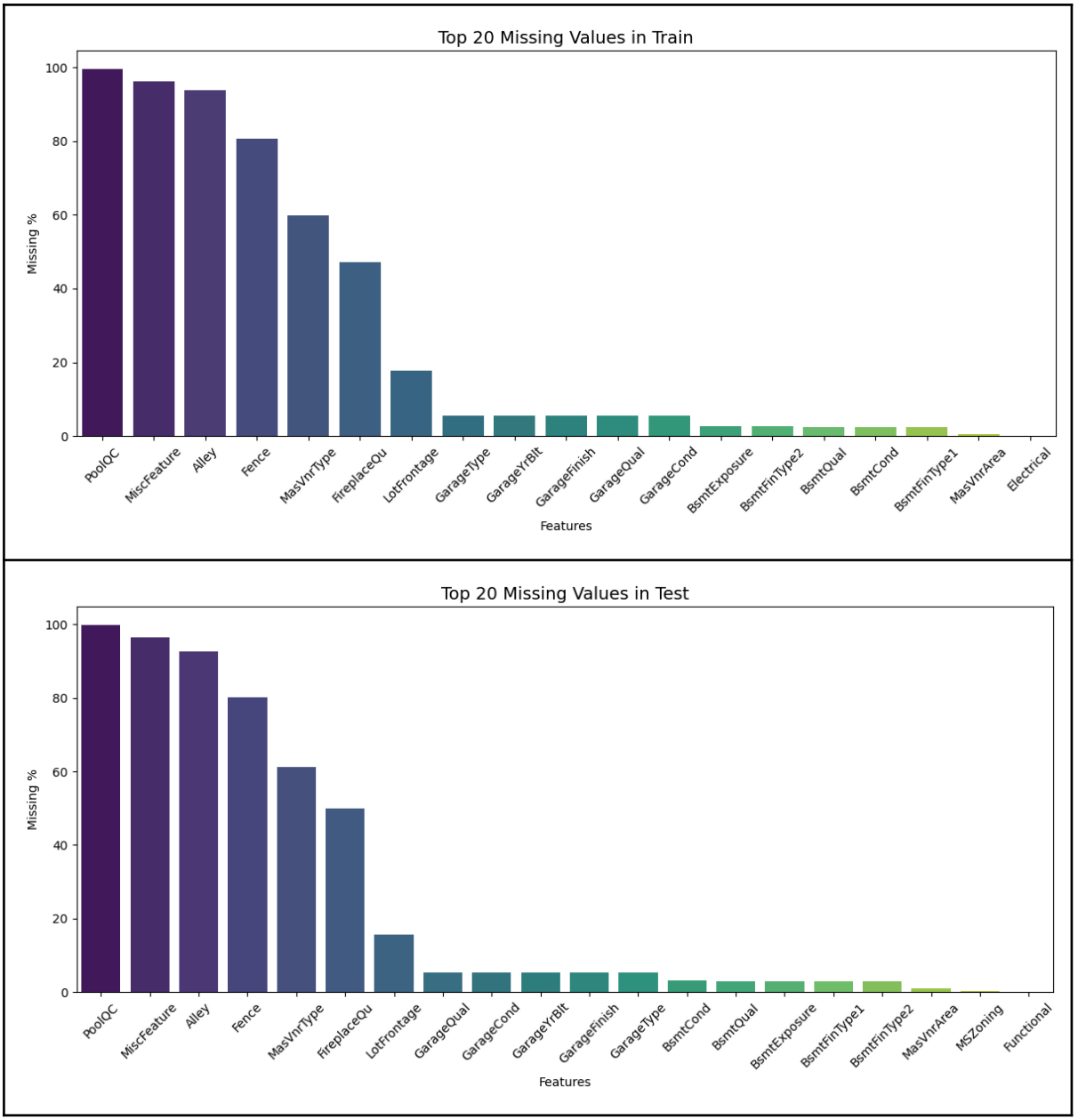
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def missing_value_summary(df, name='Dataset'):
    # Hitung jumlah dan persentase missing values
    missing = df.isnull().sum()
    missing = missing[missing > 0]
    missing_percent = (missing / len(df)) * 100

    # Gabung ke dalam DataFrame
    missing_df = pd.DataFrame({
        'Missing Count': missing,
        'Missing %': missing_percent
    }).sort_values(by='Missing %', ascending=False)

    print(f"\n Missing Value Summary for {name}:")
    print(missing_df)
```

```
    return missing_df

def plot_missing_values(missing_df, name='Dataset', top_n=20):
    if missing_df.empty:
        print(f"\n No missing values in {name}.")
        return

    # Plot top N missing features
    plt.figure(figsize=(12,6))
    sns.barplot(
        x=missing_df.head(top_n).index,
        y=missing_df['Missing %'].head(top_n),
        palette='viridis'
    )
    plt.title(f"Top {top_n} Missing Values in {name}",
fontsize=14)
    plt.xlabel("Features")
    plt.ylabel("Missing %")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()


missing_train_df = missing_value_summary(train, name='Train')
missing_test_df  = missing_value_summary(test, name='Test')

plot_missing_values(missing_train_df, name='Train')
plot_missing_values(missing_test_df, name='Test')
```

Output:

```
Missing Value Summary for Train:
              Missing Count  Missing %
PoolQC                 1453  99.520548
MiscFeature            1406  96.301370
Alley                  1369  93.767123
Fence                  1179  80.753425
MasVnrType              872  59.726027
FireplaceQu             690  47.260274
LotFrontage             259  17.739726
GarageType               81   5.547945
GarageYrBlt              81   5.547945
GarageFinish             81   5.547945
GarageQual               81   5.547945
```

```
GarageCond                81   5.547945
BsmtExposure              38   2.602740
BsmtFinType2              38   2.602740
BsmtQual                  37   2.534247
BsmtCond                  37   2.534247
BsmtFinType1              37   2.534247
MasVnrArea                 8   0.547945
Electrical                 1   0.068493

 Missing Value Summary for Test:
              Missing Count  Missing %
PoolQC                 1456  99.794380
MiscFeature            1408  96.504455
Alley                  1352  92.666210
Fence                  1169  80.123372
MasVnrType              894  61.274846
FireplaceQu             730  50.034270
LotFrontage             227  15.558602
GarageQual               78   5.346127
GarageCond               78   5.346127
GarageYrBlt              78   5.346127
GarageFinish             78   5.346127
GarageType               76   5.209047
BsmtCond                 45   3.084304
BsmtQual                 44   3.015764
BsmtExposure             44   3.015764
BsmtFinType1             42   2.878684
BsmtFinType2             42   2.878684
MasVnrArea               15   1.028101
MSZoning                  4   0.274160
Functional                2   0.137080
BsmtFullBath              2   0.137080
Utilities                 2   0.137080
BsmtHalfBath              2   0.137080
Exterior1st               1   0.068540
Exterior2nd               1   0.068540
TotalBsmtSF               1   0.068540
BsmtUnfSF                 1   0.068540
BsmtFinSF2                1   0.068540
BsmtFinSF1                1   0.068540
KitchenQual               1   0.068540
GarageArea                1   0.068540
GarageCars                1   0.068540
SaleType                  1   0.068540
```



Pengecekan missing value dilakukan terhadap dua dataset, yaitu dataset *train* dan *test*, guna mengidentifikasi kolom-kolom yang memiliki nilai kosong (missing values) dan menentukan strategi penanganannya. Dari hasil analisis pada dataset *train*, ditemukan bahwa beberapa fitur memiliki persentase nilai hilang yang sangat tinggi. Kolom PoolQC memiliki nilai hilang sebesar 99,52%, disusul oleh MiscFeature sebesar 96,30%, Alley sebesar 93,77%, dan Fence sebesar 80,75%. Nilai-nilai ini menunjukkan bahwa sebagian besar baris dalam dataset tidak memiliki informasi pada kolom-kolom tersebut, sehingga fitur-fitur ini kemungkinan besar tidak memberikan kontribusi informasi yang signifikan dan dapat dipertimbangkan untuk dihapus dari analisis lanjutan. Selain itu, terdapat juga kolom dengan tingkat missing value sedang hingga rendah seperti MasVnrType (59,73%), FireplaceQu (47,26%), dan LotFrontage (17,74%) yang masih memungkinkan untuk dilakukan imputasi, baik dengan nilai rata-rata, modus, maupun strategi khusus tergantung jenis data.

Pada dataset *test*, pola missing value serupa juga ditemukan. Kolom PoolQC, MiscFeature, Alley, dan Fence memiliki persentase nilai hilang yang tinggi, masing-masing sebesar 99,79%, 96,50%, 92,67%, dan 80,12%. Selain itu, beberapa kolom lain seperti GarageQual, BsmtQual, MasVnrType, dan FireplaceQu juga menunjukkan missing value dengan persentase antara 3% hingga 61%. Terdapat pula beberapa fitur yang hanya memiliki satu atau dua nilai hilang, seperti KitchenQual, SaleType, MSZoning, dan Exterior1st, yang dapat ditangani dengan metode imputasi sederhana tanpa mempengaruhi kualitas data secara signifikan

BAB 3 DATA PREPARATION

3.1 Data Selection

Pada tahap ini dilakukan proses *data selection* dengan memilih fitur-fitur yang memiliki korelasi tinggi terhadap variabel target, yaitu SalePrice.

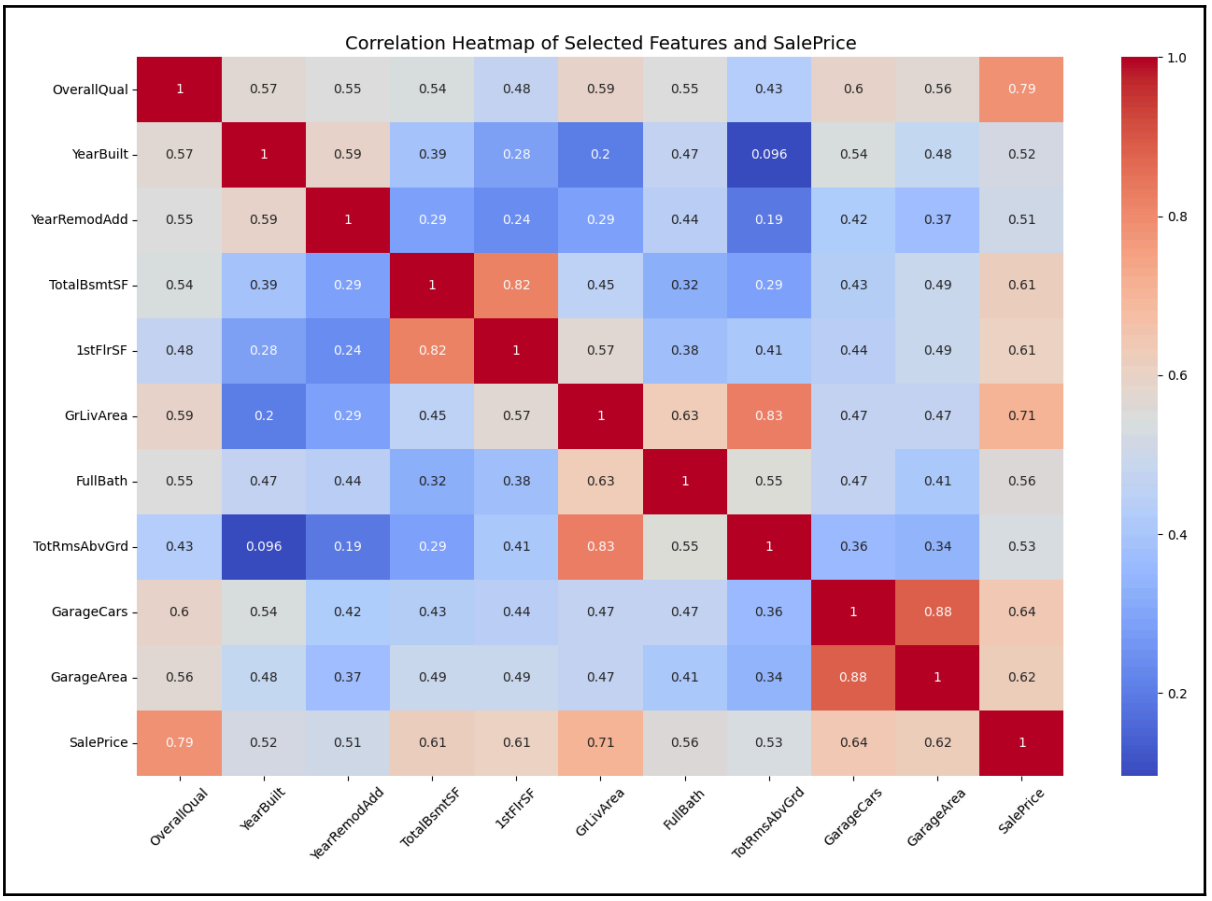
Code:

```
def select_features_by_correlation(train_data, test_data, target='SalePrice', threshold=0.5):
    """
    Memilih fitur berdasarkan korelasi dengan target,
    memastikan fitur ada di data training dan testing.
    """

    # 1. Ambil fitur numerik dari kedua data
    train_numeric = train_data.select_dtypes(include=['number'])
    test_numeric = test_data.select_dtypes(include=['number'])

    # 2. Hitung korelasi di data training
    corr = train_numeric.corr()[target].abs()

    # 3. Filter korelasi (opsional, seperti di kode awal)
    filtered_corr = corr[~((corr.index != target) & (corr.index.str.contains(target, case=False)))]

    # 4. Pilih fitur dengan korelasi di atas threshold dan ada di kedua data
    selected = filtered_corr[
```

```
        (filtered_corr >= threshold) &

        (filtered_corr.index != target) &

          (filtered_corr.index.isin(test_numeric.columns)) # Memastikan
fitur ada di test data

    ].index.tolist()



    return selected
```

```
selected_features = select_features_by_correlation(train,
target='SalePrice', threshold=0.5)

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(16, 10))
sns.heatmap(train[selected_features + ['SalePrice']].corr(),
annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap of Selected Features and
SalePrice", fontsize=14)
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```

Output:



Interpretasi:

Pada tahap ini dilakukan proses *data selection* dengan memilih fitur-fitur yang memiliki korelasi tinggi terhadap variabel target, yaitu SalePrice. Pemilihan fitur dilakukan menggunakan fungsi select_features_by_correlation() dengan ambang batas korelasi sebesar 0.5. Tujuan dari pemilihan ini adalah untuk menyaring fitur-fitur yang paling relevan terhadap harga penjualan rumah, sehingga dapat mengurangi kompleksitas data dan meningkatkan performa model.

Selanjutnya, korelasi antar fitur terpilih dan SalePrice divisualisasikan menggunakan heatmap. Dari visualisasi ini terlihat bahwa fitur seperti OverallQual, GrLivArea, GarageCars, dan TotalBsmtSF memiliki korelasi yang cukup kuat terhadap SalePrice, sehingga layak dipertimbangkan dalam pembangunan model prediktif.

3.2 Data Cleaning

3.2.1 Penanganan Missing Values

Fungsi fill_missing digunakan untuk menangani *missing values* (nilai kosong) dalam dataset, yang penting untuk menjaga kualitas data sebelum pemodelan. Fungsi ini diterapkan pada kolom-kolom terpilih (selected_features) dalam data train dan test.

Code:

```
def fill_missing(df, selected_cols):
    for col in selected_cols:
        if df[col].dtype in ['float64', 'int64']:
            df[col].fillna(df[col].median(), inplace=True)
        else:
            df[col].fillna(df[col].mode()[0], inplace=True)
    return df

train = fill_missing(train, selected_features)
test = fill_missing(test, selected_features)
```

Interpretasi:

Untuk kolom numerik, *missing values* diisi dengan **median** karena lebih tahan terhadap outlier. Untuk kolom kategorikal, digunakan **modus** (nilai paling sering muncul). Pendekatan ini mencegah kehilangan data penting dan menjaga representasi data tetap akurat tanpa menghapus baris atau kolom.

3.2.2 Penanganan Outlier

Fungsi remove_outliers digunakan untuk membersihkan data dari outlier (nilai ekstrem) dengan menggunakan metode Z-score. Fungsi ini diterapkan pada kolom-kolom numerik yang dipilih (selected_features) dalam data train.

Code:

```
def remove_outliers(df, features, z_thresh=3):
    for col in features:
        z = (df[col] - df[col].mean()) / df[col].std()
        df = df[(z > -z_thresh) & (z < z_thresh)]
    return df

train = remove_outliers(train, selected_features)
```

Outlier diidentifikasi jika nilai Z-score melebihi ambang batas (default = 3), lalu dihapus dari dataset. Pendekatan ini membantu meningkatkan kualitas data dan performa model, karena outlier dapat mengganggu hasil analisis, terutama pada model regresi.

3.3 Data Construct

Mengubah data agar distribusi lebih baik dengan fungsi fix_skewness digunakan untuk memperbaiki distribusi data yang skewed (miring) dengan menerapkan transformasi logaritma pada fitur-fitur numerik yang memiliki skewness tinggi.

```
def fix_skewness(df, threshold=0.5):
    skewed_feats = df[selected_features].apply(lambda x:
skew(x.dropna())).sort_values(ascending=False)
    skewed_feats = skewed_feats[skewed_feats > threshold]
    for feat in skewed_feats.index:
        df[feat] = np.log1p(df[feat])
    return df

train = fix_skewness(train)
test = fix_skewness(test)
```

Interpretasi:

Fitur dengan skewness > 0.5 diidentifikasi, lalu dilakukan transformasi log1p untuk mengurangi kemiringan distribusinya. Hal ini penting untuk meningkatkan performa model yang sensitif terhadap distribusi data, seperti regresi linear, serta membantu data lebih memenuhi asumsi normalitas.

3.4 Labeling Data

Dalam proyek ini, proses labeling tidak diperlukan karena dataset yang digunakan berasal dari kompetisi di Kaggle dan sudah mencakup label target yaitu SalePrice. Labeling umumnya dibutuhkan pada pendekatan unsupervised learning atau ketika bekerja dengan data mentah tanpa target. Karena ini adalah kasus supervised learning, dan label sudah tersedia, maka tahap ini dapat dilewati.

3.5 Data Integration

Proyek ini memanfaatkan dua dataset utama yang berasal dari sumber yang sama, yaitu platform Kaggle, terdiri atas data pelatihan (train) dan data pengujian (test). Dengan demikian, proses data integration atau penggabungan data dari berbagai sumber tidak diperlukan. Umumnya, integrasi data dibutuhkan ketika data diperoleh dari berbagai sistem atau file yang berbeda. Namun, dalam konteks proyek ini, seluruh data telah disediakan secara terstruktur dan konsisten dalam satu paket kompetisi, sehingga tahap integrasi dapat dikesampingkan.

BAB 4 MODELLING

Bagian ini adalah tahapan pembangunan dan pelatihan model machine learning yang digunakan dalam analisis data, pembangunan model dilakukan dengan menggunakan tiga algoritma berbeda: LightGBM, XGBoost, dan CatBoost. Selain itu, stacking model digunakan untuk meningkatkan akurasi model dengan menggabungkan prediksi dari ketiga model dasar tersebut.

4.1 Build Model

4.1.1 Melatih Model Dasar

Melatih tiga model dasar, yaitu LightGBM, XGBoost, dan CatBoost, dengan Fungsi train_models(), langkah-langkah melatih model:

4.1.1.1 Inisialisasi Model

Model-model ini diinisialisasi dengan beberapa parameter dasar untuk memastikan bahwa model dapat dilatih dan diuji dengan parameter awal yang konsisten. Setiap model diinisialisasi dengan menggunakan parameter random_state=42 untuk memastikan bahwa hasilnya dapat direproduksi. Selain itu, untuk model LightGBM dan XGBoost, parameter regulasi seperti reg_alpha dan reg_lambda disertakan untuk menambahkan regularisasi L1 dan L2 pada model untuk mencegah overfitting.

```
# LightGBM
lgb_model = lgb.LGBMRegressor(random_state=42, verbose=-1,
reg_alpha=1.0, reg_lambda=1.0)

# XGBoost
xgb_model = xgb.XGBRegressor(random_state=42, reg_alpha=1.0,
reg_lambda=1.0)

# CatBoost
cat_model = cb.CatBoostRegressor(silent=True, random_state=42)
```

4.1.1.2 Penentuan Hyperparameter

Setiap model memiliki hyperparameter grid yang berisi rentang nilai yang akan diuji selama proses tuning.
Untuk LightGBM, contohnya, n_estimators, learning_rate, num_leaves, dan max_depth dipilih sebagai parameter yang akan diuji.
Untuk XGBoost dan CatBoost, parameter yang diuji juga termasuk n_estimators, learning_rate, max_depth, dan parameter terkait lainnya.

```
# LightGBM params
lgb_params = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.05],
    'num_leaves': [15, 31],
    'max_depth': [1, 2, 3],
}

# XGBoost params
xgb_params = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.05],
    'max_depth': [1, 2],
}

# CatBoost params
cat_params = {
    'iterations': [200, 300],
    'learning_rate': [0.01, 0.05],
    'depth': [3, 4],
    'l2_leaf_reg': [5, 10],
}
```

4.1.1.3 Tuning Hyperparameter

Proses tuning dilakukan menggunakan fungsi tune_model() yang memanfaatkan GridSearchCV dengan 10-fold cross-validation (cv=10). Metode evaluasi yang digunakan adalah *negative root mean squared error* (neg-RMSE) untuk mendapatkan model dengan generalisasi terbaik.

```
grid = GridSearchCV(model, param_grid,
scoring='neg_root_mean_squared_error', cv=10, verbose=0)
grid.fit(X_train, y_train)
```

Output:

```
Tuning LightGBM...
Best params for LightGBM: {'learning_rate': 0.05, 'max_depth': 3,
'n_estimators': 200, 'num_leaves': 15}
Tuning XGBoost...
Best params for XGBoost: {'learning_rate': 0.05, 'max_depth': 2,
'n_estimators': 200}
Tuning CatBoost...
Best params for CatBoost: {'depth': 4, 'iterations': 300,
'l2_leaf_reg': 5, 'learning_rate': 0.05}
```

4.1.1.2 Pelatihan Model Final

Setelah menemukan parameter terbaik, model dilatih menggunakan data pelatihan (X_train, y_train).

Model yang terlatih disimpan dalam dictionary models, yang memungkinkan kita untuk mengaksesnya secara mudah untuk digunakan pada tahap berikutnya.

```
models['lgb'] = tune_model(lgb_model, lgb_params, X_train,
y_train, "LightGBM")
models['xgb'] = tune_model(xgb_model, xgb_params, X_train,
y_train, "XGBoost")
models['cat'] = tune_model(cat_model, cat_params, X_train,
y_train, "CatBoost")
```

4.2.1 Penggabungan Model dengan Stacking

Stacking adalah teknik ensemble di mana beberapa model dasar (base models) digabungkan untuk menghasilkan prediksi akhir. Berikut proses stacking:

4.2.1.1 Prediksi oleh Base Models

Prediksi dari model-model dasar (LightGBM, XGBoost, dan CatBoost) dihasilkan untuk data pelatihan (X_train) dan data validasi (X_val).

Prediksi dari setiap model disusun dalam bentuk array 2D menggunakan np.column_stack(), di mana setiap kolom berisi prediksi dari satu model.

```
train_preds = np.column_stack([models[m].predict(X_train) for m in
models])
val_preds = np.column_stack([models[m].predict(X_val) for m in
models])
```

4.2.1.2 Pelatihan Meta-model

Sebuah meta-model (dalam hal ini menggunakan Ridge Regression) dilatih menggunakan prediksi dari model dasar sebagai input dan variabel target (y_train) sebagai output.

Meta-model ini bertujuan untuk menggabungkan hasil prediksi dari model-model dasar dengan cara yang lebih optimal.

```
meta_model = Ridge(alpha=1.0)
meta_model.fit(train_preds, y_train)
```

4.2.1.3 Prediksi Meta-model

Setelah meta-model dilatih, digunakan untuk menghasilkan prediksi pada data pelatihan (train_preds) dan data validasi (val_preds).

Prediksi dari meta-model disimpan dalam train_meta_pred dan val_meta_pred.

```
train_meta_pred = meta_model.predict(train_preds)
val_meta_pred = meta_model.predict(val_preds)
```

BAB 5 EVALUATION

Evaluasi model dilakukan untuk menilai seberapa baik model yang dibangun mampu memprediksi nilai target pada data pelatihan dan data validasi. Pada eksperimen ini, evaluasi dilakukan menggunakan fungsi evaluate_predictions().

5.1 Evaluasi Prediction

Fungsi evaluate_predictions() digunakan untuk menghitung metrik-metrik evaluasi yang umum digunakan dalam regresi:

- Input:
  - y_train dan y_val adalah nilai target aktual dari data pelatihan dan validasi.
  - train_stack_pred dan val_stack_pred adalah prediksi yang dihasilkan oleh meta-model untuk data pelatihan dan validasi.
  - name="Stacked Model" → hanya label untuk identifikasi model saat menampilkan hasil

- Metrik Evaluasi:
  - RMSE (Root Mean Squared Error) mengukur rata-rata selisih kuadrat antara nilai prediksi dan nilai aktual. Metrik ini memberikan informasi tentang seberapa jauh prediksi dari nilai sebenarnya.
  - MAE (Mean Absolute Error) mengukur rata-rata selisih absolut antara prediksi dan nilai aktual. MAE memberikan gambaran yang lebih sederhana mengenai kesalahan model.
  - R-squared (R^2) mengukur seberapa besar proporsi variansi dalam data yang dapat dijelaskan oleh model. Nilai R-squared yang lebih tinggi menunjukkan bahwa model dapat menjelaskan lebih banyak variansi dari data target.

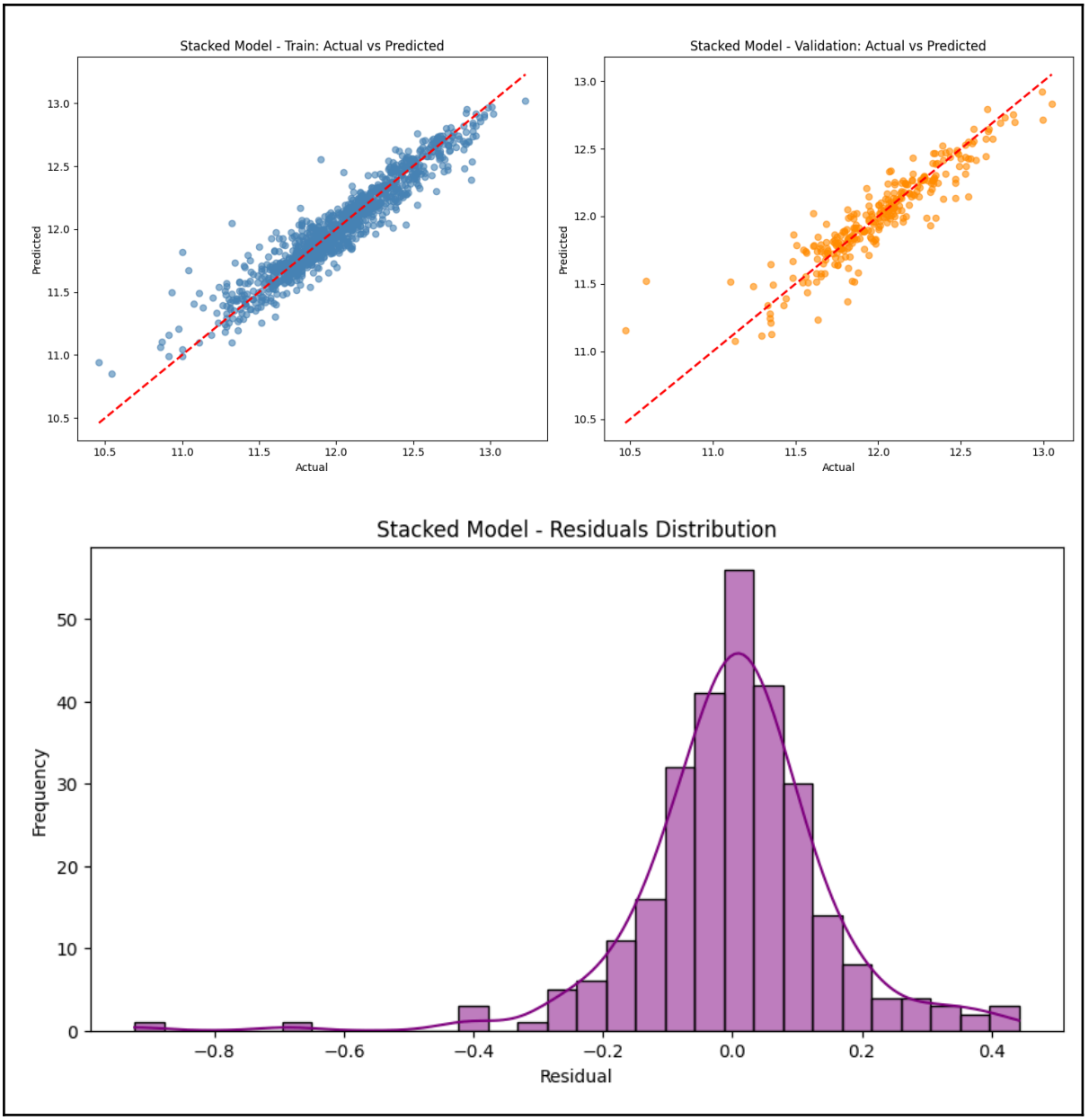
Code:

```
# Fungsi evaluasi untuk menghitung metrik performa model

def evaluate_predictions(y_train, train_pred, y_val, val_pred,
name="Model"):

    print(f"\n{name} Performance:")



    print("\nTrain Set:")
```

```
    print(f"  RMSE: {np.sqrt(mean_squared_error(y_train,
train_pred)):.4f}")  # Root Mean Squared Error

    print(f"  MAE:  {mean_absolute_error(y_train, train_pred):.4f}")  #
Mean Absolute Error

    print(f"  R²:   {r2_score(y_train, train_pred):.4f}")  # R-squared
(koefisien determinasi)



    print("\nValidation Set:")

    print(f"  RMSE: {np.sqrt(mean_squared_error(y_val,
val_pred)):.4f}")

    print(f"  MAE:  {mean_absolute_error(y_val, val_pred):.4f}")

    print(f"  R²:   {r2_score(y_val, val_pred):.4f}")
```

Output:

```
Tuning LightGBM...
Best params for LightGBM: {'learning_rate': 0.05, 'max_depth': 3,
'n_estimators': 200, 'num_leaves': 15}
Tuning XGBoost...
Best params for XGBoost: {'learning_rate': 0.05, 'max_depth': 2,
'n_estimators': 200}
Tuning CatBoost...
Best params for CatBoost: {'depth': 4, 'iterations': 300,
'l2_leaf_reg': 5, 'learning_rate': 0.05}

Stacked Model Performance:

Train Set:
  RMSE: 0.1213
  MAE:  0.0873
  R²:   0.9001

Validation Set:
  RMSE: 0.1465
  MAE:  0.0996
  R²:   0.8367
```



Interpretasi Hasil Model Ensemble (Stacking):

Model ini menggunakan pendekatan stacking ensemble yang menggabungkan tiga model terbaik:

- LightGBM
- XGBoost
- CatBoost

Dengan Ridge Regression sebagai meta-model.

Hyperparameter Terbaik

Model	Hyperparameter Terbaik
LightGBM	learning_rate: 0.05, max_depth: 3, n_estimators: 200, num_leaves: 15
XGBoost	learning_rate: 0.05, max_depth: 2, n_estimators: 200

CatBoost learning_rate: 0.05, depth: 4, iterations: 300, l2_leaf_reg: 5

Evaluasi Kinerja Model

Training Set

- RMSE (Root Mean Squared Error): 0.1213
- MAE (Mean Absolute Error): 0.0873
- R^2 (R-squared): 0.9001

Validation Set

- RMSE: 0.1465
- MAE: 0.0996
- R^2: 0.8367

Interpretasi

1. Akurasi Tinggi
   Nilai R^2 yang tinggi menunjukkan bahwa model sangat baik dalam menjelaskan variansi data.
2. Tidak Overfitting
   Perbedaan metrik antara training dan validation kecil, artinya model mampu melakukan generalisasi dengan baik.
3. Kesalahan Prediksi Rendah
   Nilai RMSE dan MAE yang kecil menandakan bahwa kesalahan rata-rata dalam prediksi cukup rendah.

Analisis Visual

- Plot Prediksi vs Aktual:
  Titik-titik prediksi mendekati garis diagonal (garis ideal), menunjukkan prediksi yang akurat.
- Distribusi Residuals:
  Residual tersebar merata dan simetris di sekitar nol. Ini mengindikasikan tidak adanya bias sistematis.

Model stacking ensemble yang menggabungkan LightGBM, XGBoost, dan CatBoost dengan Ridge Regression sebagai meta-model menunjukkan performa yang sangat baik. Hasil evaluasi menunjukkan nilai akurasi tinggi pada data pelatihan maupun validasi, dengan nilai R^2 sebesar 0.9001 pada training dan 0.8367 pada validasi. Hal ini menandakan bahwa model memiliki kemampuan generalisasi yang baik dan tidak mengalami overfitting.

Nilai error yang rendah (RMSE dan MAE) menunjukkan bahwa model mampu memprediksi harga rumah secara akurat dan konsisten. Distribusi residual yang simetris dan grafik prediksi yang mendekati garis aktual memperkuat bukti bahwa model bekerja secara optimal.

Secara keseluruhan, model ini sangat layak untuk digunakan dalam memprediksi harga rumah pada data nyata. Namun, untuk meningkatkan akurasi lebih lanjut, disarankan melakukan eksplorasi lanjutan pada feature engineering, penambahan data, atau pengujian meta-model lain sebagai pembanding.

BAB 6 DEPLOYMENT

Deployment merupakan tahapan akhir dalam siklus proyek House Pricing, yang bertujuan untuk mengintegrasikan model yang telah dibangun ke dalam suatu sistem yang dapat diakses dan digunakan oleh pengguna akhir (end-user). Dalam proyek ini, deployment masih dilakukan secara lokal menggunakan aplikasi web berbasis Flask, yang memungkinkan pengguna untuk melakukan prediksi harga rumah berdasarkan input fitur yang diberikan secara langsung melalui antarmuka web.

6.1 Persiapan Deployment

6.1.1 Evaluasi Prediction

Model yang telah dibangun terdiri dari tiga komponen utama:

- meta_model: model utama yang menggabungkan output dari base models.
- base_models: kumpulan model dasar yang digunakan dalam ensemble stacking.
- scaler: objek standar scaler untuk transformasi fitur numerik.

Ketiga komponen ini disimpan dalam sebuah dictionary model_bundle, kemudian diserialisasi menggunakan modul pickle dan disimpan dalam file .pkl. Berikut adalah potongan kode untuk proses penyimpanan model:

```
import pickle

# Bungkus komponen
model_bundle = {
    'meta_model': meta_model,
    'base_models': models,
    'scaler': scaler
}

with
open('/content/drive/MyDrive/Dami/housepricesadvancedregressiontechniques/stacked_model.pkl', 'wb') as f:
    pickle.dump(model_bundle, f)

print("Model berhasil disimpan ke Google Drive.")
```

```
import pickle
```

```
# Load kembali model dari file
with
open('/content/drive/MyDrive/Dami/housepricesadvancedregressiontechniques/stacked_model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

# Akses komponen
meta_model_loaded = loaded_model['meta_model']
base_models_loaded = loaded_model['base_models']
scaler_loaded = loaded_model['scaler']

print("Model berhasil dimuat dari Google Drive.")
```

6.1.2 Struktur Folder Aplikasi

Struktur folder aplikasi deployment disusun sebagai berikut:

```
HousePrice_K10/
│
├── app.py
├── stacked_model.pkl
├── templates/
│   └── index.html
├── static/
   └── style.css
```

6.2 Implementasi Aplikasi Web

6.2.1 Framework Flask

Flask dipilih sebagai framework aplikasi web karena sifatnya yang ringan, fleksibel, dan mudah digunakan untuk keperluan prototyping maupun deployment aplikasi machine learning. Flask juga mendukung integrasi yang baik dengan template HTML dan routing yang sederhana.

6.2.1 File app.py – Backend dengan Flask

Pada File app.py merupakan inti dari aplikasi Flask, di dalamnya memiliki Inisialisasi Flask dan pemanggilan model, Routing ke halaman utama (/) untuk form input, Routing ke /predict untuk memproses prediksi. Berikut code di app.py:

```
from flask import Flask, render_template, request
import joblib
import numpy as np
import pandas as pd

app = Flask(__name__)

MODEL_PATH = r'D:\Adeeee\New folder\Dami\HousePriceProyek\stacked_model.pkl'
loaded_model = joblib.load(MODEL_PATH)  # loaded_model

# Fungsi prediksi harga rumah
def predict_house_price(features):
    feature_names = [
        'OverallQual', 'YearBuilt', 'YearRemodAdd', 'TotalBsmtSF',
        '1stFlrSF', 'GrLivArea', 'FullBath', 'TotRmsAbvGrd',
        'GarageCars', 'GarageArea'
    ]

    # Load model & scaler dari dictionary
    meta_model = loaded_model['meta_model']
    base_models = loaded_model['base_models']
    scaler = loaded_model['scaler']

    # Konversi input ke DataFrame
    features_df = pd.DataFrame([features], columns=feature_names)

    # Scaling
    features_scaled = scaler.transform(features_df)

    # Prediksi base models
    meta_features = np.column_stack([
        base_models[m].predict(features_scaled) for m in base_models
    ])

    # Prediksi akhir
    final_prediction = meta_model.predict(meta_features)[0]
    return np.expm1(final_prediction)  # Transformasi balik log1p

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
```

```
def predict():
    if request.method == 'POST':
        # Ambil input dari form (tanpa GarageYrBlt)
        overall_quality = int(request.form['overall_quality'])
        year_built = int(request.form['year_built'])
        year_remod_add = int(request.form['year_remod_add'])
        total_bsmt_sf = int(request.form['total_bsmt_sf'])
        first_flr_sf = int(request.form['first_flr_sf'])
        gr_liv_area = int(request.form['gr_liv_area'])
        full_bath = int(request.form['full_bath'])
        total_rooms_abv_grd = int(request.form['total_rooms_abv_grd'])
        garage_cars = int(request.form['garage_cars'])
        garage_area = int(request.form['garage_area'])

        # Susun dictionary fitur
        features = {
            'OverallQual': overall_quality,
            'YearBuilt': year_built,
            'YearRemodAdd': year_remod_add,
            'TotalBsmtSF': total_bsmt_sf,
            '1stFlrSF': first_flr_sf,
            'GrLivArea': gr_liv_area,
            'FullBath': full_bath,
            'TotRmsAbvGrd': total_rooms_abv_grd,
            'GarageCars': garage_cars,
            'GarageArea': garage_area
        }

        predicted_price = predict_house_price(features)
        return render_template('index.html', predicted_price=predicted_price)

#standar untuk memulai aplikasi web di Flask.
if __name__ == '__main__':
    app.run(debug=True)
```

Pada app.py di di lakukan inisialisasi aplikasi Flask dan pemanggilan model hasil pelatihan yang telah disimpan dalam bentuk file stacked_model.pkl. File tersebut berisi meta model, base models, serta objek scaler yang digunakan untuk standardisasi data. Aplikasi memiliki dua routing utama, yaitu ke halaman utama (/) yang menampilkan form input, serta ke endpoint /predict yang menangani proses prediksi berdasarkan input pengguna.

Pada saat pengguna mengisi form dan mengirimkannya, data input akan diambil melalui metode POST dan dikonversi ke dalam bentuk dictionary. Dictionary ini kemudian diubah menjadi objek DataFrame dan distandardisasi menggunakan scaler yang sebelumnya telah dilatih bersama data pelatihan. Setelah data distandardisasi, masing-masing base model akan menghasilkan prediksi, dan hasil-hasil ini digabungkan menjadi fitur meta. Meta model kemudian menggunakan fitur meta ini untuk menghasilkan prediksi akhir harga rumah.

Karena target SalePrice pada saat pelatihan sebelumnya telah ditransformasi dengan logaritma natural (log1p), maka hasil prediksi dikembalikan ke skala aslinya menggunakan fungsi np.expm1().

Jika prediksi berhasil diproses, hasilnya akan ditampilkan kembali ke halaman index.html. Namun, jika terjadi kesalahan selama proses prediksi, maka aplikasi akan menampilkan pesan error yang menjelaskan kesalahan tersebut. Saat ini, aplikasi masih dijalankan secara lokal dalam mode debug (debug=True) untuk keperluan pengujian dan pengembangan.

6.2.1 File index.html – Form Input dan Tampilan Prediksi

Template HTML ini menyediakan antarmuka pengguna untuk memasukkan data dan melihat hasil prediksi.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>House Price Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>

<header>
    <h1>Development of a Predictive Regression Model for House Prices</h1>
    <p>Using Ensemble Stacking Techniques</p>
</header>

<main class="main-content">
    <div class="container">
        <h2>Business Understanding</h2>
        <p>Industri properti sangat dinamis dan memiliki dampak besar terhadap
ekonomi. Proyek ini bertujuan untuk menyediakan sistem prediksi harga rumah
berbasis machine learning untuk membantu pemangku kepentingan dalam pengambilan
keputusan yang lebih cepat dan tepat.</p>

        <h2>Data Understanding</h2>
        <h3>Deskripsi Data</h3>
        <p>Dataset diambil dari kompetisi "House Prices - Advanced Regression
Techniques" di Kaggle. Fitur numerik dan kategorikal dipilih untuk model
prediksi, dengan target utama adalah harga jual rumah (SalePrice).</p>

        <h3>Distribusi Harga Rumah</h3>
        <p>Distribusi harga rumah menunjukkan pola positively skewed, dengan
harga mayoritas di kisaran rendah dan beberapa harga tinggi yang jauh lebih
tinggi (outlier).</p>

        <h2>Prediction Input</h2>
```

```
        <p>Masukkan data fitur rumah untuk prediksi harga:</p>

        <form action="/predict" method="post">
            <label for="overall_quality">Overall Quality (1-10):</label>
            <input type="number" id="overall_quality" name="overall_quality"
required>

            <label for="year_built">Year Built:</label>
            <input type="number" id="year_built" name="year_built" required>

            <label for="year_remod_add">Year Remodeled:</label>
            <input type="number" id="year_remod_add" name="year_remod_add"
required>

            <label for="total_bsmt_sf">Total Basement SF:</label>
            <input type="number" id="total_bsmt_sf" name="total_bsmt_sf"
required>

            <label for="first_flr_sf">1st Floor SF:</label>
            <input type="number" id="first_flr_sf" name="first_flr_sf" required>

            <label for="gr_liv_area">Above Ground Living Area
(GrLivArea):</label>
            <input type="number" id="gr_liv_area" name="gr_liv_area" required>

            <label for="full_bath">Full Bathrooms:</label>
            <input type="number" id="full_bath" name="full_bath" required>

            <label for="total_rooms_abv_grd">Total Rooms Above Ground:</label>
            <input type="number" id="total_rooms_abv_grd"
name="total_rooms_abv_grd" required>

            <label for="garage_cars">Garage Capacity (Cars):</label>
            <input type="number" id="garage_cars" name="garage_cars" required>

            <label for="garage_area">Garage Area (sqft):</label>
            <input type="number" id="garage_area" name="garage_area" required>

            <input type="submit" value="Predict Price">
        </form>

        {% if predicted_price %}
        <h2>Prediction Result</h2>
        <p>The predicted price for the house is: <strong>${{ predicted_price |
round(2) }}</strong></p>
        {% endif %}
    </div>
</main>

</body>
```

```
</html>
```

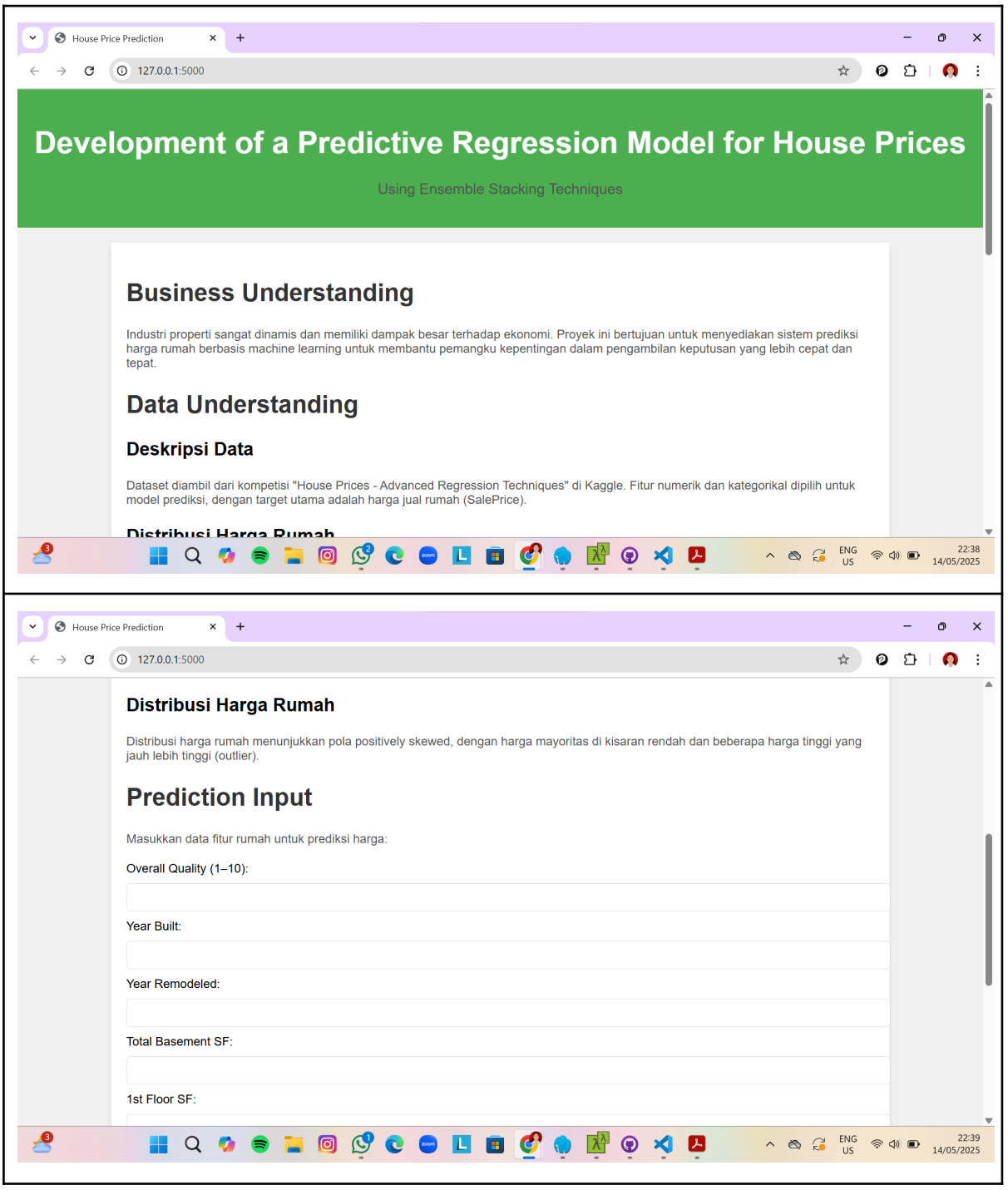
6.2.1 File style.css – Desain Antarmuka

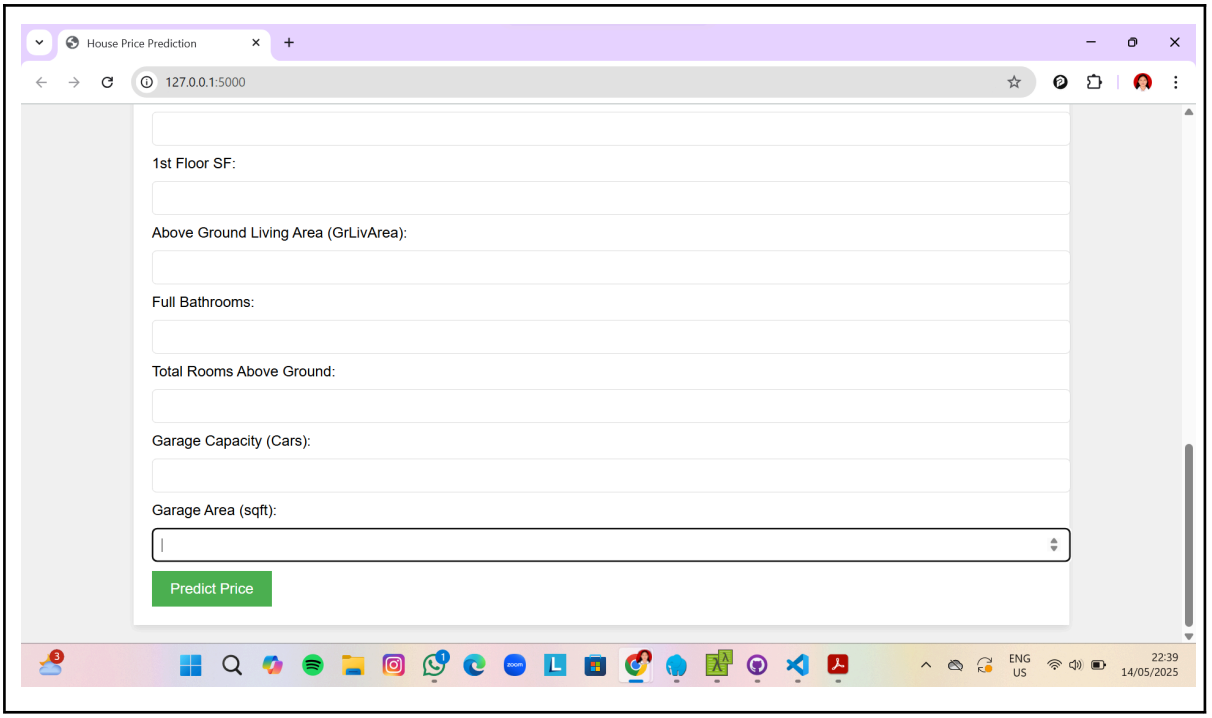
File CSS disimpan di folder static/ dan bertugas membuat tampilan lebih menarik.

```
/* General Styles */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

/* Header */
header {
    background-color: #4CAF50;
    color: white;
    padding: 20px;
    text-align: center;
}

header h1 {
    font-size: 2.5em;
}

header p {
    font-size: 1.2em;
}

/* Main Content */
.main-content {
    display: flex;
    justify-content: center;
    align-items: flex-start;
    padding: 20px;
}

.container {
    width: 80%;
    max-width: 1200px;
    background-color: white;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 20px;
}
```

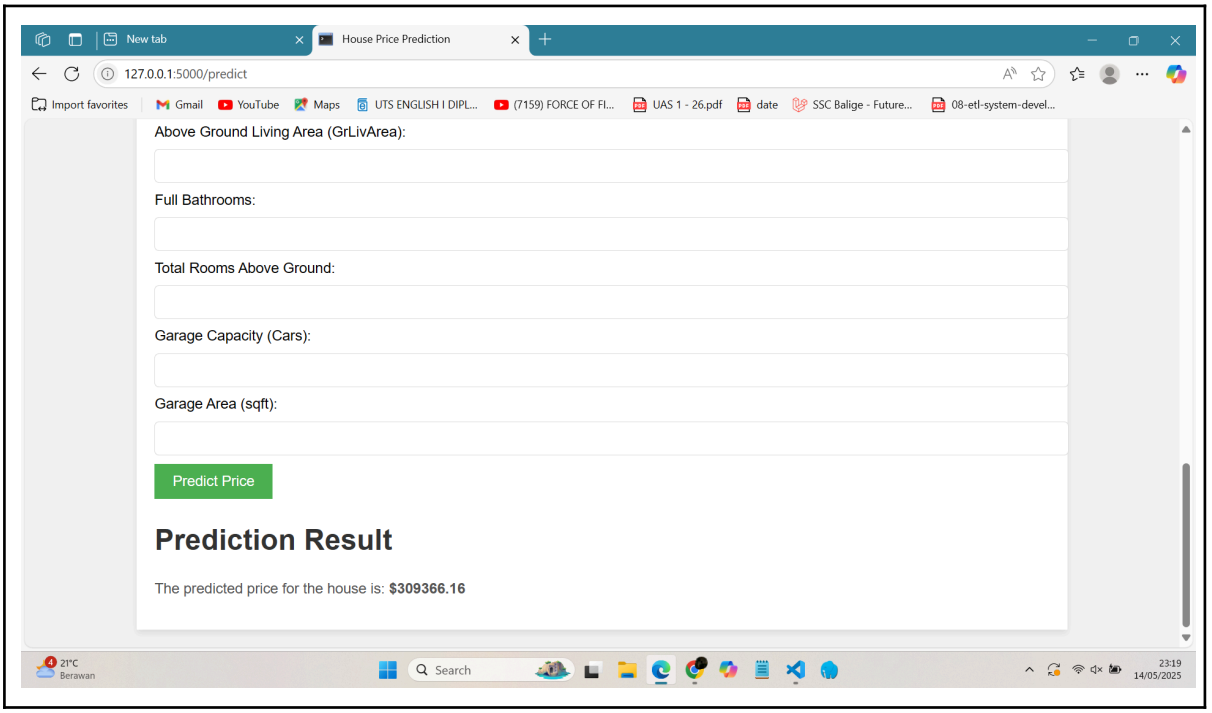
```css
h2 {
    color: #333;
    font-size: 2em;
}

h3 {
    font-size: 1.5em;
}

p {
    font-size: 1em;
    color: #555;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    font-size: 1em;
}

button:hover {
    background-color: #45a049;
}

/* Form Styling */
form {
    margin-top: 20px;
}

input[type="text"], input[type="number"], textarea {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ddd;
    border-radius: 5px;
}

input[type="submit"] {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    font-size: 1em;
}
```

Tampilan antarmuka pengguna:





Tampilan sudah di prediksi price:



DAFTAR PUSTAKA

[1] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[2] D. H. Wolpert, "Stacked Generalization," *Neural Networks*, vol. 5, no. 2, pp. 241-259, 1992.

[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785-794.

[4] J. Zhang, Z. Xu, and W. Liu, "Predicting House Prices Using Machine Learning Algorithms: A Review," Computational Intelligence and Neuroscience, vol. 2020, Article ID 4532859, 2020, pp. 1-15.