LAPORAN PRAKTIKUM

PRAKTIKUM SISTEM TELEKOMUNIKASI II

MENGEMBANGKAN MODEL PREDIKTIF

Diajukan untuk memenuhi salah satu tugas Mata Kuliah Praktikum Sistem Telekomunikasi II

Dosen Pengampu: Hafiyyan Putra Pratama, S.ST., M.T.

Disusun oleh Kelompok 6 Kelas 4A:

Arif Assidiq	NIM 2312055
Aprian Maulana Suryawan	NIM.2304298
M Rislan Tristansyah	NIM.2307874

PROGRAM STUDI SISTEM TELEKOMUNIKASI

UNIVERSITAS PENDIDIKAN INDONESIA

KAMPUS PURWAKARTA

2025

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

Perkembangan teknologi dan ketersediaan data dalam jumlah besar telah mendorong pemanfaatan teknik machine learning dalam berbagai bidang, termasuk sektor properti. Salah satu aplikasi penting dalam bidang ini adalah pengembangan model prediktif untuk memperkirakan harga rumah. Prediksi harga rumah yang akurat tidak hanya bermanfaat bagi pembeli dan penjual, tetapi juga bagi pengembang, agen properti, dan lembaga keuangan dalam mengambil keputusan yang lebih tepat berbasis data.

Tugas ini bertujuan untuk mengembangkan model prediktif yang mampu memperkirakan harga rumah berdasarkan berbagai fitur, seperti luas tanah, jumlah kamar, lokasi, dan karakteristik lainnya. Proyek ini menggunakan dataset "House Prices: Advanced Regression Techniques" yang tersedia secara publik, yang menyediakan data historis harga rumah beserta atribut-atribut yang memengaruhinya.

Proses pengembangan model mencakup beberapa tahapan utama. Dimulai dari eksplorasi awal data untuk memahami struktur dan jenis fitur yang tersedia, dilanjutkan dengan tahap preprocessing yang melibatkan pembersihan data dari missing values dan outliers, transformasi fitur numerik maupun kategorikal, serta seleksi fitur yang relevan. Analisis eksploratif data (EDA) dilakukan untuk memahami distribusi data dan hubungan antar fitur dengan harga rumah.

Setelah data dipersiapkan, langkah berikutnya adalah pembagian data menjadi set pelatihan dan pengujian, diikuti dengan pemilihan dan pelatihan berbagai model regresi seperti Linear Regression, Decision Tree, Random Forest, dan Gradient Boosting. Model yang dikembangkan kemudian divalidasi menggunakan teknik cross-validation dan dioptimalkan melalui hyperparameter tuning. Evaluasi akhir dilakukan menggunakan metrik seperti RMSE, MAE, dan R^2, serta visualisasi hasil prediksi dibandingkan dengan nilai aktual.

Seluruh proses dianalisis dan didokumentasikan secara menyeluruh dalam laporan ini, yang mencakup penjelasan metode, kode program, visualisasi data, serta diskusi mengenai temuan dan keterbatasan dari model yang dikembangkan. Dengan pendekatan yang sistematis dan berbasis data, diharapkan model yang dihasilkan mampu memberikan prediksi harga rumah yang akurat dan dapat diandalkan.

BAB II

HASIL DAN PEMBAHASAN

1. Pembentukan Kelompok dan Pengenalan Data

Pada tahap ini pertama – tama yaitu mengunduh terlebih dahulu file dataset dengan format csv. Disini ada file yang bernama train.csv dan test csv, dimana masing – masing memiliki kegunaanya. Untuk train.csv digunakan untuk melatih model machine learning yang akan dibuat dan untuk test.csv digunakan untuk mengetes model yang sudah dilatih menggunakan dataset latih apakah menunjukkan nilai yang akurat atau tidak untuk prediksi modelnya.

Jika sudah diunduh lakukan pembacan file csv dengan kode python seperti berikut:

```
[ ]  #Import library
     import pandas as pd

     #Mengimpor data dari file format.CSV
     dataTr=pd.read_csv("/content/train.csv")
     dataTs=pd.read_csv("/content/test.csv")
```

Gambar 1 dataset

Kode tersebut pertama-tama **mengimpor pustaka pandas** dengan alias pd, yang merupakan alat penting untuk analisis dan manipulasi data dalam Python. Selanjutnya, kode ini **membaca dua file CSV** yang berbeda, yaitu "train.csv" dan "test.csv", yang diasumsikan berada di direktori "/content/". Masing-masing file CSV ini kemudian **dimuat ke dalam struktur data pandas yang disebut DataFrame**, yang diberi nama dataTr untuk data pelatihan dan dataTs untuk data pengujian, memungkinkan pengguna untuk dengan mudah bekerja dengan data tabular tersebut.

Pada tahap selanjutnya yaitu menampilkan data dengan menggunakan kode dataTr dan dataTs yang memungkinkan untuk melakukan berbagai operasi analisis data, manipulasi, dan visualisasi menggunakan fungsionalitas yang disediakan oleh pustaka pandas. Seperti gambar dibawah ini:

```
dataTr
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Po
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	

1460 rows × 81 columns

+ Kode + Teks

Gambar 2 Menampilkan Data

```
dataTs
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Sc
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	...	
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	...	
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	...	
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...	
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...	

1459 rows × 80 columns

Gambar 3 Menampilkan Data

Pada kedua gambar output kode tersebut menampilkan ringkaasan dari DataFrame pandas, yang berisi baik data pelatihan maupun yang berisi data test. Dataa Frame ini terdiri dari 1460 baris dan 81 kolom (untuk dataset latih), untuk yang satu lagi terdiri dari 1459 baris dan 81 kolom (untuk dataset test). Kolom-kolom yang terlihat seperti Id, MSSubClass, MSZoning, LotFrontage, LotArea, Street, hingga SalePrice menunjukkan bahwa ini adalah dataset perumahan, di mana setiap baris mendeskripsikan sebuah properti dengan berbagai karakteristiknya, dan kolom SalePrice adalah variabel target yang ingin diprediksi. Kehadiran NaN di beberapa sel (misalnya pada kolom Alley atau PoolQC) mengindikasikan adanya nilai yang hilang (missing values) dalam dataset tersebut.

Kode berikutnya yang digunakan dalam ekplorasi data ialah data.columns ini digunakan untuk melihat fitur apa saja yang tersedia disetiap dataset, hal ini memungkinkan seorang data sains dapat menyusun strategi untuk menangani kasus yang sedang ia hadapi.

```
dataTr.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

Gambar 4 Ekplorasi data columns

Gambar 5 Eksplorasi data columns

Kode dataTr.info() memanggil sebuah **metode bawaan dari objek DataFrame pandas** (dataTr) yang bernama info(). Metode ini berfungsi untuk **menampilkan ringkasan ringkas** mengenai DataFrame tersebut. Informasi yang biasanya ditampilkan meliputi jumlah total entri (baris), jumlah kolom, nama setiap kolom beserta tipe datanya (misalnya, integer, float, object/string), jumlah nilai non-null (nilai yang tidak kosong) pada setiap kolom, dan perkiraan penggunaan memori oleh DataFrame tersebut. Berikut output yang dihasilkan

Table 1 Output info

No	Colom	Total non-null		Dtype
0	Id	1460	non-null	int64
1	MSSubClass	1460	non-null	int64
2	MSZoning	1460	non-null	object
3	LotFrontage	1201	non-null	float64
4	LotArea	1460	non-null	int64
5	Street	1460	non-null	object
6	Alley	91	non-null	object
7	LotShape	1460	non-null	object
8	LandContour	1460	non-null	object
9	Utilities	1460	non-null	object
10	LotConfig	1460	non-null	object
11	LandSlope	1460	non-null	object
12	Neighborhood	1460	non-null	object
13	Condition1	1460	non-null	object
14	Condition2	1460	non-null	object
15	BldgType	1460	non-null	object
16	HouseStyle	1460	non-null	object
17	OverallQual	1460	non-null	int64
18	OverallCond	1460	non-null	int64
19	YearBuilt	1460	non-null	int64
20	YearRemodAdd	1460	non-null	int64
21	RoofStyle	1460	non-null	object
22	RoofMatl	1460	non-null	object
23	Exterior1st	1460	non-null	object
24	Exterior2nd	1460	non-null	object

25	MasVnrType	588	non-null	object
26	MasVnrArea	1452	non-null	float64
27	ExterQual	1460	non-null	object
28	ExterCond	1460	non-null	object
29	Foundation	1460	non-null	object
30	BsmtQual	1423	non-null	object
31	BsmtCond	1423	non-null	object
32	BsmtExposure	1422	non-null	object
33	BsmtFinType1	1423	non-null	object
34	BsmtFinSF1	1460	non-null	int64
35	BsmtFinType2	1422	non-null	object
36	BsmtFinSF2	1460	non-null	int64
37	BsmtUnfSF	1460	non-null	int64
38	TotalBsmtSF	1460	non-null	int64
39	Heating	1460	non-null	object
40	HeatingQC	1460	non-null	object
41	CentralAir	1460	non-null	object
42	Electrical	1459	non-null	object
43	1stFlrSF	1460	non-null	int64
44	2ndFlrSF	1460	non-null	int64
45	LowQualFinSF	1460	non-null	int64
46	GrLivArea	1460	non-null	int64
47	BsmtFullBath	1460	non-null	int64
48	BsmtHalfBath	1460	non-null	int64
49	FullBath	1460	non-null	int64
50	HalfBath	1460	non-null	int64
51	BedroomAbvGr	1460	non-null	int64
52	KitchenAbvGr	1460	non-null	int64
53	KitchenQual	1460	non-null	object
54	TotRmsAbvGrd	1460	non-null	int64
55	Functional	1460	non-null	object
56	Fireplaces	1460	non-null	int64
57	FireplaceQu	770	non-null	object
58	GarageType	1379	non-null	object
59	GarageYrBlt	1379	non-null	float64
60	GarageFinish	1379	non-null	object
61	GarageCars	1460	non-null	int64
62	GarageArea	1460	non-null	int64
63	GarageQual	1379	non-null	object
64	GarageCond	1379	non-null	object
65	PavedDrive	1460	non-null	object
66	WoodDeckSF	1460	non-null	int64
67	OpenPorchSF	1460	non-null	int64
68	EnclosedPorch	1460	non-null	int64
69	3SsnPorch	1460	non-null	int64
70	ScreenPorch	1460	non-null	int64
71	PoolArea	1460	non-null	int64
72	PoolQC	7	non-null	object

73	Fence	281	non-null	object
74	MiscFeature	54	non-null	object
75	MiscVal	1460	non-null	int64
76	MoSold	1460	non-null	int64
77	YrSold	1460	non-null	int64
78	SaleType	1460	non-null	object
79	SaleCondition	1460	non-null	object
80	SalePrice	1460	non-null	int64

Tabel di atas merupakan output dari perintah dataTr.info(), yang menyajikan analisis mendalam terhadap struktur DataFrame dataTr. Secara keseluruhan, DataFrame ini memiliki **1460 entri (baris)**, dan untuk setiap kolom (fitur), tabel ini merinci **jumlah nilai non-null** (nilai yang tidak kosong) serta **tipe datanya (Dtype)**. Kita dapat melihat beragam tipe data: int64 untuk fitur numerik seperti Id, LotArea, dan SalePrice; float64 untuk fitur numerik desimal seperti LotFrontage dan MasVnrArea; serta object yang umumnya menandakan data teks atau kategorikal seperti MSZoning, Street, dan Alley.

Poin paling krusial dari analisis tabel ini adalah identifikasi **nilai yang hilang (missing values)**. Beberapa kolom seperti Id, MSSubClass, dan LotArea memiliki 1460 nilai non-null, menunjukkan tidak ada data yang hilang. Namun, banyak kolom lain yang memiliki jumlah nilai non-null kurang dari 1460, yang mengindikasikan adanya data kosong. Sebagai contoh, LotFrontage hanya memiliki 1201 nilai non-null (artinya ada 259 nilai hilang), Alley sangat sedikit terisi dengan hanya 91 nilai non-null, MasVnrType memiliki 588 nilai, BsmtQual kehilangan 37 nilai, FireplaceQu kehilangan 690 nilai, dan yang paling signifikan adalah PoolQC, Fence, serta MiscFeature yang memiliki sangat sedikit data terisi (masing-masing 7, 281, dan 54 nilai non-null). Informasi ini sangat penting dikarenakan penanganan nilai yang hilang (seperti imputasi atau penghapusan kolom/baris) akan menjadi langkah krusial dalam tahap pra-pemrosesan data sebelum data dapat digunakan untuk analisis lebih lanjut atau pemodelan.

Selanjutnya kode display(dataTr.describe(), dataTr.describe(include='object')) bertujuan untuk **menampilkan dua ringkasan statistik deskriptif** dari DataFrame dataTr secara bersamaan. Panggilan pertama, dataTr.describe(), akan secara otomatis menghitung dan menampilkan statistik dasar untuk **semua kolom numerik** dalam dataTr. Statistik ini biasanya mencakup jumlah data (count), rata-rata (mean), standar deviasi (std), nilai minimum (min), kuartil pertama (25%), median (50%), kuartil ketiga (75%), dan nilai maksimum (max) untuk setiap kolom numerik. Panggilan kedua, dataTr.describe(include='object'), secara khusus meminta statistik deskriptif untuk **semua kolom yang memiliki tipe data 'object'** (biasanya kolom yang berisi teks atau data kategorikal). Untuk kolom-kolom ini, statistik yang ditampilkan meliputi jumlah data (count), jumlah nilai unik (unique), nilai yang paling sering muncul (top), dan frekuensi kemunculan nilai teratas tersebut (freq). Fungsi display() digunakan untuk memastikan kedua tabel ringkasan statistik ini ditampilkan dengan baik di output.

```
display(dataTr.describe(), dataTr.describe(include='object'))
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd
count	1460.000000	1460.000000	1201.000000	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
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000

8 rows × 38 columns

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition
count	1460	1460	91	1460	1460	1460	1460	1460	1460	146
unique	5	2	2	4	4	2	5	3	25	
top	RL	Pave	Grvl	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norr

Gambar 6 Display

Setelah melakukan yang diatas lanjut ke tahap selanjutnya dengan memasukan kode yang melakukan iterasi atau perulangan pada setiap kolom dalam DataFrame yang memiliki tipe data ‘object’. Untuk setiap kolom yang memenuhi kriteria ini, skrip tersebut pertama-tama akan mencetak nama kolom tersebut sebagai judul, diikuti dengan daftar semua **nilai unik** yang terdapat dalam kolom itu. Akhirnya, sebuah garis pemisah yang terdiri dari 40 tanda hubung dicetak untuk membedakan output antar kolom, sehingga memudahkan pembacaan hasil eksplorasi variasi nilai pada fitur-fitur kategorikal.

```
for col in dataTr.select_dtypes(include='object').columns:
    print(f"Unique values for column '{col}':")
    print(dataTr[col].unique())
    print("-" * 40)
```

Gambar 7 Proses Iterasi

Table 2 Fitur Unik

ColumnName	**UniqueValues**
MSZoning	['RL' 'RM' 'C (all)' 'FV' 'RH']
Street	['Pave' 'Grvl']
Alley	[nan 'Grvl' 'Pave']
LotShape	['Reg' 'IR1' 'IR2' 'IR3']
LandContour	['Lvl' 'Bnk' 'Low' 'HLS']
Utilities	['AllPub' 'NoSeWa']
LotConfig	['Inside' 'FR2' 'Corner' 'CulDSac' 'FR3']
LandSlope	['Gtl' 'Mod' 'Sev']
Neighborhood	['CollgCr' 'Veenker' 'Crawfor' 'NoRidge' 'Mitchel' 'Somerst' 'NWAmes' 'OldTown' 'BrkSide' 'Sawyer' 'NridgHt' 'NAmes' 'SawyerW' 'IDOTRR' 'MeadowV' 'Edwards' 'Timber' 'Gilbert' 'StoneBr' 'ClearCr' 'NPkVill' 'Blmngtn' 'BrDale' 'SWISU' 'Blueste']
Condition1	['Norm' 'Feedr' 'PosN' 'Artery' 'RRAe' 'RRNn' 'RRAn' 'PosA' 'RRNe']
Condition2	['Norm' 'Artery' 'RRNn' 'Feedr' 'PosN' 'PosA' 'RRAn' 'RRAe']
BldgType	['1Fam' '2fmCon' 'Duplex' 'TwnhsE' 'Twnhs']
HouseStyle	['2Story' '1Story' '1.5Fin' '1.5Unf' 'SFoyer' 'SLvl' '2.5Unf' '2.5Fin']

RoofStyle	['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
RoofMatl	['CompShg' 'WdShngl' 'Metal' 'WdShake' 'Membran' 'Tar&Grv' 'Roll' 'ClyTile']
Exterior1st	['VinylSd' 'MetalSd' 'Wd Sdng' 'HdBoard' 'BrkFace' 'WdShing' 'CemntBd' 'Plywood' 'AsbShng' 'Stucco' 'BrkComm' 'AsphShn' 'Stone' 'ImStucc' 'CBlock']
Exterior2nd	['VinylSd' 'MetalSd' 'Wd Shng' 'HdBoard' 'Plywood' 'Wd Sdng' 'CmentBd' 'BrkFace' 'Stucco' 'AsbShng' 'Brk Cmn' 'ImStucc' 'AsphShn' 'Stone' 'Other' 'CBlock']
MasVnrType	['BrkFace' nan 'Stone' 'BrkCmn']
ExterQual	['Gd' 'TA' 'Ex' 'Fa']
ExterCond	['TA' 'Gd' 'Fa' 'Po' 'Ex']
Foundation	['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
BsmtQual	['Gd' 'TA' 'Ex' nan 'Fa']
BsmtCond	['TA' 'Gd' nan 'Fa' 'Po']
BsmtExposure	['No' 'Gd' 'Mn' 'Av' nan]
BsmtFinType1	['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' nan 'LwQ']
BsmtFinType2	['Unf' 'BLQ' nan 'ALQ' 'Rec' 'LwQ' 'GLQ']
Heating	['GasA' 'GasW' 'Grav' 'Wall' 'OthW' 'Floor']
HeatingQC	['Ex' 'Gd' 'TA' 'Fa' 'Po']
CentralAir	['Y' 'N']
Electrical	['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' nan]
KitchenQual	['Gd' 'TA' 'Ex' 'Fa']
Functional	['Typ' 'Min1' 'Maj1' 'Min2' 'Mod' 'Maj2' 'Sev']
FireplaceQu	[nan 'TA' 'Gd' 'Fa' 'Ex' 'Po']
GarageType	['Attchd' 'Detchd' 'BuiltIn' 'CarPort' nan 'Basment' '2Types']
GarageFinish	['RFn' 'Unf' 'Fin' nan]
GarageQual	['TA' 'Fa' 'Gd' nan 'Ex' 'Po']
GarageCond	['TA' 'Fa' nan 'Gd' 'Po' 'Ex']
PavedDrive	['Y' 'N' 'P']
PoolQC	[nan 'Ex' 'Fa' 'Gd']
Fence	[nan 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']
MiscFeature	[nan 'Shed' 'Gar2' 'Othr' 'TenC']
SaleType	['WD' 'New' 'COD' 'ConLD' 'ConLI' 'CWD' 'ConLw' 'Con' 'Oth']
SaleCondition	['Normal' 'Abnorml' 'Partial' 'AdjLand' 'Alloca' 'Family']

Tabel diatas merupakan output kode yang menghasilkan fitur yang bertype object dan nilai disetiap fitur yang unik, dimana nantinya akan digunakan pada tahap preprocessing data.

2. Preprocessing Data

Preprocessing data adalah serangkaian langkah penting yang dilakukan untuk membersihkan dan mengubah data mentah ke dalam format yang siap dan sesuai untuk dianalisis atau digunakan dalam pemodelan machine learning. Kualitas hasil analisis atau model sangat bergantung pada kualitas data yang dimasukkan, sesuai dengan prinsip "Garbage In, Garbage Out". Proses ini memastikan bahwa data menjadi lebih bersih, terstruktur dengan baik, dan fitur-fiturnya relevan. Output pada tahap ini ialah suatu dataset yang sudah optimal, dimana nilai yang hilang telah ditangani, data telah diubah sesuai format yang memudahkan machine learning, dan fitur-fitur yang tidak relevan atau redundan dihilangkan.

Pata tahap pertama ialah pembersihan data, dimana proses untuk mendeteksi, menangani, dan memperbaiki atau menghapus data yang "kotor", tidak akurat, tidak

konsisten, atau tidak lengkap dalam sebuah dataset. Tujuannya adalah untuk meningkatkan kualitas data sehingga lebih andal untuk analisis. Hal yang dilakukan mencari nilai yang NaN, berikut kode yang digunakan untuk menghapus fitur yang tidak diperlukan karena memiliki jumlah nilai yang hilang terlalu banyak.

```
# Daftar kolom yang akan di-drop
columns_to_drop = [
    "Id",           # Identifier
    "Alley",        # 93.8% missing
    "PoolQC",       # 99.5% missing
    "Fence",        # 80.7% missing
    "MiscFeature", # 96.3% missing
    "FireplaceQu", # Redundant with Fireplaces
    "3SsnPorch",    # Hampir semua 0
    "LowQualFinSF",# Hampir semua 0
    "Condition2",   # Overlap dengan Condition1
    "Exterior2nd", # Mirip dengan Exterior1st
    "GarageYrBlt", # Redundant dengan YearBuilt
    "RoofMatl",
    "MasVnrType",
    "LotFrontage"# Distribusi sangat timpang
]

# Drop kolom dari data
data_tr_cleaned = dataTr.drop(columns=columns_to_drop)
# Drop kolom dari data.ts
data_ts_cleaned = dataTs.drop(columns=columns_to_drop)
```

Gambar 8 Menghapus berberapa fitur

Kode tersebut menghapus kolom fitur yang tertentu dari dua DataFrame. Seperti berikut:

- **"Id"**: Dihapus karena merupakan kolom **Identifier** atau penanda unik untuk setiap baris. Kolom seperti ini biasanya tidak memiliki nilai prediktif untuk model machine learning.
- **"Alley"**: Dihapus karena memiliki persentase **nilai hilang yang sangat tinggi, yaitu 93.8%**. Dengan sebagian besar data tidak tersedia, kolom ini sulit untuk diimputasi secara akurat dan kemungkinan tidak banyak berkontribusi.
- **"PoolQC"** (Kualitas Kolam Renang): Dihapus karena persentase **nilai hilang yang ekstrem, yaitu 99.5%**. Ini menandakan bahwa hampir semua rumah dalam dataset tidak memiliki informasi kualitas kolam renang (atau tidak memiliki kolam renang).
- **"Fence"** (Pagar): Dihapus karena **80.7% nilainya hilang**. Tingginya persentase data yang hilang membuat kolom ini kurang reliabel.
- **"MiscFeature"** (Fitur Tambahan Lain-lain): Dihapus karena **96.3% nilainya hilang**. Sama seperti "PoolQC", sebagian besar data untuk fitur ini tidak tersedia.
- **"FireplaceQu"** (Kualitas Perapian): Dihapus karena dianggap **redundant (berlebihan) dengan kolom "Fireplaces"** (jumlah perapian). Jika jumlah perapian

adalah 0, maka kualitas perapian tidak relevan, dan jika ada perapian, informasinya mungkin sudah cukup terwakili oleh jumlahnya atau fitur lain.

- **"3SsnPorch"** (Luas Teras Tiga Musim): Dihapus karena nilainya **hampir semua 0**. Ini menunjukkan bahwa sangat sedikit rumah yang memiliki teras jenis ini, sehingga variasinya rendah dan kontribusi prediktifnya minim.
- **"LowQualFinSF"** (Luas Area Berkualitas Rendah): Dihapus karena nilainya **hampir semua 0**. Mirip dengan "3SsnPorch", fitur ini memiliki varians yang sangat kecil.
- **"Condition2"** (Kondisi Kedekatan dengan Jalan/Rel Tambahan): Dihapus karena informasinya dianggap **overlap (tumpang tindih) dengan "Condition1"**. Kemungkinan besar "Condition1" sudah cukup menangkap informasi utama terkait kondisi lokasi.
- **"Exterior2nd"** (Material Eksterior Kedua): Dihapus karena informasinya dianggap **mirip dengan "Exterior1st"** (material eksterior utama). Untuk menyederhanakan model, salah satunya dihilangkan.
- **"GarageYrBlt"** (Tahun Garasi Dibangun): Dihapus karena dianggap **redundant (berlebihan) dengan "YearBuilt"** (Tahun Rumah Dibangun). Seringkali tahun garasi dibangun sama atau sangat dekat dengan tahun rumah dibangun.

Fitur – fitur yang tersisa dan memiliki nilai yang hilang sedikit (numerik) akan dilakukan pengisian nilai yang hilang dengan rata-rata (mean) dari kolom itu, sedangkan untuk kolom kategorikal, nilai yang hilang akan diisi dengan modus (mode) atau nilai yang paling sering muncul pada kolom tersebut (dengan memastikan modus tidak kosong sebelum diakses).

```
import pandas as pd

# Imputasi missing values dengan mean (numerik) atau mode (kategorikal)
for col in data_tr_cleaned.columns:
    if data_tr_cleaned[col].isnull().any():
        if pd.api.types.is_numeric_dtype(data_tr_cleaned[col]):
            # Gunakan mean untuk kolom numerik
            data_tr_cleaned[col] = data_tr_cleaned[col].fillna(data_tr_cleaned[col].mean())
        else:
            # Gunakan mode untuk kolom kategorikal
            mode = data_tr_cleaned[col].mode()
            if not mode.empty:
                data_tr_cleaned[col] = data_tr_cleaned[col].fillna(mode[0])

# Verifikasi apakah masih ada missing values
missing_values_after_imputation = data_tr_cleaned.isnull().sum()
print("\nMissing values after imputation:")
print(missing_values_after_imputation[missing_values_after_imputation > 0])
```

Gambar 9 Pengisisan nilai yang hilang

Jika sudah lakukan normalisasi pada fitur yang numerik, yaitu me**ngubah atau menyesuaikan skala nilai dari berbagai fitur numerik agar memiliki rentang yang sebanding atau mengikuti distribusi standar.** Misalnya, nilai-nilai dapat diubah agar berada dalam rentang 0 hingga 1 (menggunakan metode Min-Max Scaling) atau diubah sehingga memiliki rata-rata 0 dan standar deviasi 1 (menggunakan metode Standardization atau Z-score normalization). Normalisasi ini krusial karena banyak algoritma machine learning sensitif

terhadap perbedaan skala antar fitur; tanpa normalisasi, fitur dengan rentang nilai yang lebih besar dapat mendominasi proses pembelajaran dan mempengaruhi kinerja model secara negatif.

```
# prompt: lakukan normalisasi untuk nilai numerik

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Normalize numerical features using MinMaxScaler
numerical_cols = data_tr_cleaned.select_dtypes(include=['number']).columns
scaler = MinMaxScaler()
data_tr_cleaned[numerical_cols] = scaler.fit_transform(data_tr_cleaned[numerical_cols])
```

```
[ ]  # prompt: lakukan normalisasi untuk nilai numerik

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Normalize numerical features using MinMaxScaler
numerical_cols = data_ts_cleaned.select_dtypes(include=['number']).columns
scaler = MinMaxScaler()
data_ts_cleaned[numerical_cols] = scaler.fit_transform(data_ts_cleaned[numerical_cols])
```

Gambar 10 Melakukan Normalisasi

Lanjut ke tahap transformasi data dimana disini ada 2 hal yang akan dilakukan dalam label encoding, yaitu untuk yang fitur ordinal dan nominal mendapati perhatian yang berbeda. Untuk fitur-fitur kategorikal yang bersifat ordinal dalam dataset ini, di mana terdapat tingkatan atau urutan yang jelas dan bermakna antar kategorinya, teknik **Label Encoding** akan diterapkan. Secara spesifik, fitur-fitur seperti **'OverallQual'** (kualitas keseluruhan), **'OverallCond'** (kondisi keseluruhan), **'ExterQual'** (kualitas eksterior), **'ExterCond'** (kondisi eksterior), **'BsmtQual'** (kualitas basement), **'BsmtCond'** (kondisi basement), **'HeatingQC'** (kualitas pemanas), **'KitchenQual'** (kualitas dapur), **'Fireplaces'** (jumlah perapian, yang dapat dianggap ordinal berdasarkan jumlahnya), **'GarageQual'** (kualitas garasi), **'GarageCond'** (kondisi garasi), **'PoolArea'** (luas kolam, yang mungkin telah dikategorikan secara ordinal atau akan diperlakukan demikian berdasarkan nilainya), serta fitur-fitur lain yang menunjukkan tingkatan seperti **'BsmtExposure'**, **'BsmtFinType1'**, **'BsmtFinType2'**, **'Functional'**, dan **'GarageFinish'**, akan diproses menggunakan metode ini. Proses Label Encoding ini akan menetapkan sebuah nilai numerik unik untuk setiap kategori dalam masing-masing fitur tersebut (misalnya, 0 untuk kategori dengan tingkatan terendah, 1 untuk tingkatan berikutnya, dan seterusnya), sehingga secara efektif mempertahankan informasi urutan yang esensial ini untuk dapat diolah lebih lanjut oleh model machine learning.

Sebaliknya, untuk fitur-fitur kategorikal dalam dataset ini yang bersifat **nominal**, di mana tidak ada urutan atau tingkatan inheren yang melekat antar kategorinya, pendekatan **One-Hot Encoding** akan menjadi metode transformasi yang lebih sesuai dan akan diterapkan. Fitur-fitur seperti **'MSZoning'** (klasifikasi zona), **'Street'** (tipe jalan), **'LotShape'** (bentuk lahan), **'LandContour'** (kontur lahan), **'Utilities'** (utilitas yang tersedia), **'LotConfig'** (konfigurasi lahan), **'LandSlope'** (kemiringan lahan), **'Neighborhood'** (lokasi lingkungan), **'Condition1'** (kedekatan dengan kondisi utama), **'BldgType'** (tipe bangunan), **'HouseStyle'** (gaya rumah),

'**RoofStyle**' (gaya atap), '**Exterior1st**' (material eksterior utama), '**Foundation**' (tipe pondasi), '**Heating**' (tipe pemanas), '**CentralAir**' (pendingin udara sentral), '**Electrical**' (sistem kelistrikan), '**GarageType**' (tipe garasi), '**PavedDrive**' (jalan masuk beraspal), '**SaleType**' (tipe penjualan), dan '**SaleCondition**' (kondisi penjualan) akan diolah menggunakan teknik ini. One-Hot Encoding bekerja dengan cara mengubah setiap kategori unik dalam masing-masing fitur nominal tersebut menjadi beberapa kolom biner (bernilai 0 atau 1) baru. Setiap kolom baru ini akan secara spesifik mewakili satu kategori, di mana nilai 1 menandakan keberadaan kategori tersebut untuk suatu observasi dan 0 sebaliknya. Dengan demikian, metode ini secara efektif menghindari pengenaan asumsi urutan artifisial pada data, meskipun penting untuk dicatat bahwa penerapannya dapat meningkatkan jumlah total kolom (dimensi) dalam dataset, terutama jika suatu fitur memiliki banyak kategori unik.

```python
# Identify ordinal and nominal features (replace with your actual feature lists)
ordinal_features = ['OverallQual', 'OverallCond', 'ExterQual', 'ExterCond',
                    'BsmtQual', 'BsmtCond', 'HeatingQC', 'KitchenQual',
                    'Fireplaces', 'GarageQual', 'GarageCond', 'PoolArea', 'BsmtExposure', 'BsmtFinType1',
                    'BsmtFinType2', 'Functional', 'GarageFinish' ] # Example, replace with actual ordinal features
nominal_features = ['MSZoning', 'Street', 'LotShape', 'LandContour', 'Utilities',
                   'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
                   'BldgType', 'HouseStyle', 'RoofStyle', 'Exterior1st',
                   'Foundation', 'Heating', 'CentralAir',
                   'Electrical', 'GarageType', 'PavedDrive', 'SaleType', 'SaleCondition'] # Example, replace with your actual nominal features

# Label Encoding for ordinal features
label_encoder = LabelEncoder()
for col in ordinal_features:
    if col in data_tr_cleaned.columns:  # Check if the column exists
        data_tr_cleaned[col] = label_encoder.fit_transform(data_tr_cleaned[col])
# One-Hot Encoding for nominal features
onehot_encoder = OneHotEncoder(handle_unknown='ignore', sparse_output=False) # sparse=False for direct array
encoded_nominal = onehot_encoder.fit_transform(data_tr_cleaned[nominal_features])
encoded_nominal_df = pd.DataFrame(encoded_nominal, columns=onehot_encoder.get_feature_names_out(nominal_features))
# Drop the original nominal columns
data_tr_cleaned = data_tr_cleaned.drop(columns=nominal_features)
# Concatenate the one-hot encoded features
data_tr_encoded = pd.concat([data_tr_cleaned, encoded_nominal_df], axis=1)
print(data_tr_encoded.head())
```

Gambar 11 Mengubah menjadi numerik

```
   MSSubClass   LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  \
0    0.235294  0.033420            6            4   0.949275      0.883333
1    0.000000  0.038795            5            7   0.753623      0.433333
2    0.235294  0.046507            6            4   0.934783      0.866667
3    0.294118  0.038561            6            4   0.311594      0.333333
4    0.235294  0.060576            7            4   0.927536      0.833333

   MasVnrArea  ExterQual  ExterCond  BsmtQual  ...  SaleType_ConLw  \
0     0.12250          2          4         2  ...             0.0
1     0.00000          3          4         2  ...             0.0
2     0.10125          2          4         2  ...             0.0
3     0.00000          3          4         3  ...             0.0
4     0.21875          2          4         2  ...             0.0

   SaleType_New  SaleType_Oth  SaleType_WD  SaleCondition_Abnorml  \
0           0.0           0.0          1.0                    0.0
1           0.0           0.0          1.0                    0.0
2           0.0           0.0          1.0                    0.0
3           0.0           0.0          1.0                    1.0
4           0.0           0.0          1.0                    0.0

   SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                    0.0                   0.0                   0.0
1                    0.0                   0.0                   0.0
2                    0.0                   0.0                   0.0
3                    0.0                   0.0                   0.0
4                    0.0                   0.0                   0.0

   SaleCondition_Normal  SaleCondition_Partial
0                   1.0                    0.0
```

Gambar 12 Data menjadi numerik

Merupakan hasil label encode disini terlihat bahwa semuanya sudah menjadi numerik hal ini memudahkan pembacaan data untuk model machine learning sehingga dapat melakukan fungsinya secara optimal.

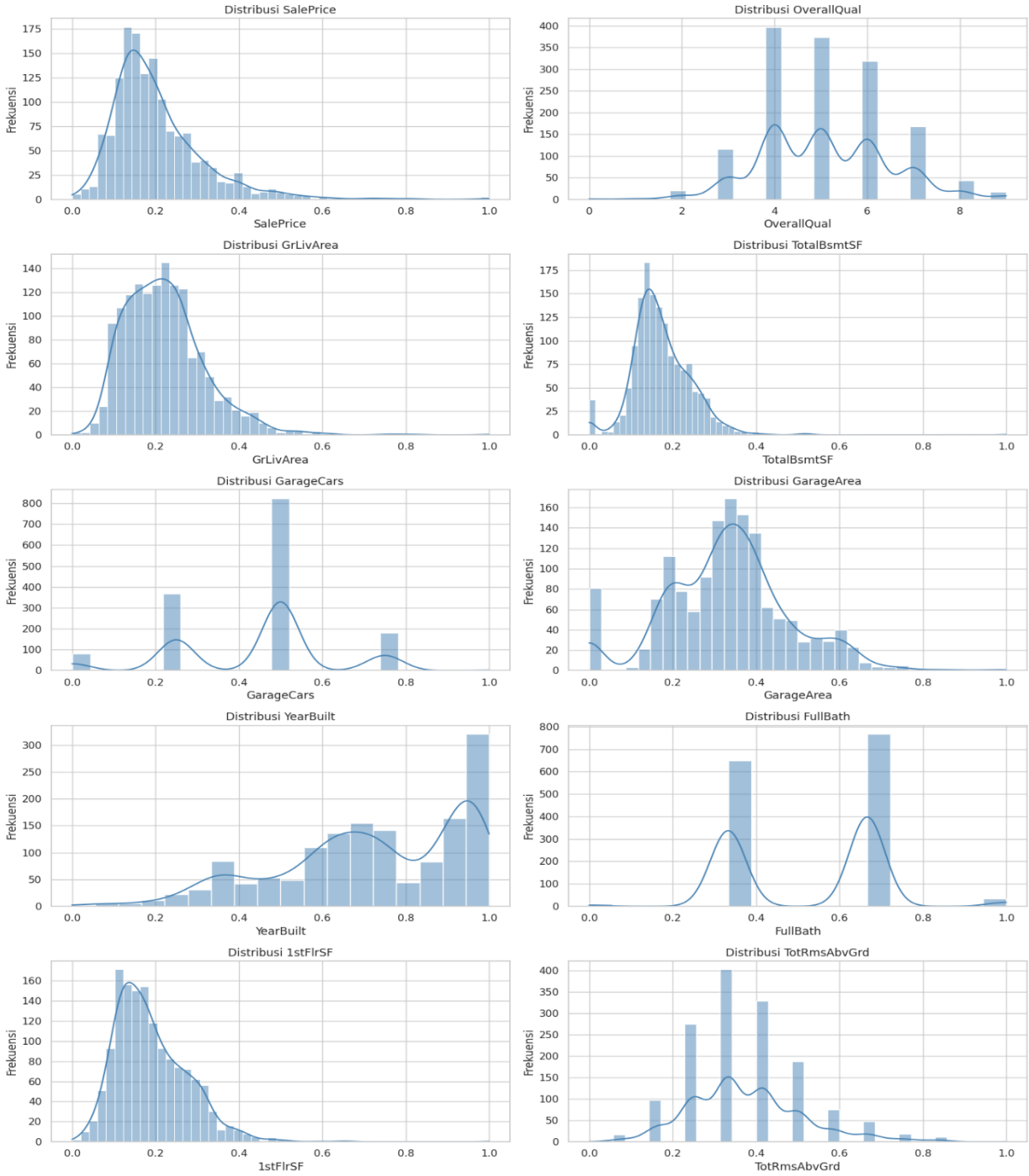
3. Analisis Eksplorasi Data (EDA)

Analisis Eksplorasi Data (EDA) merupakan proses investigasi awal terhadap data dimana untuk menemukan pola, mendeteksi anomali, menguji hipotesis, dan sekaligus memeriksa asusmsi dengan bantuan statistik ringkasan dan representasi grafis. Tujuan utama dibuatnya EDA ialah untuk memahami dataset secara mendalam sebelum melakukan sebuah pemodelan.

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

# Daftar fitur utama (plus target)
main_features = [
    'SalePrice', 'OverallQual', 'GrLivArea', 'TotalBsmtSF',
    'GarageCars', 'GarageArea', 'YearBuilt', 'FullBath',
    '1stFlrSF', 'TotRmsAbvGrd'
]

# Visualisasi distribusi masing-masing fitur
sns.set(style="whitegrid")
plt.figure(figsize=(15, 20))

for i, feature in enumerate(main_features):
    plt.subplot(5, 2, i + 1)
    sns.histplot(data_tr_encoded[feature], kde=True, color='steelblue')
    plt.title(f'Distribusi {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frekuensi')

plt.tight_layout()
plt.show()
```

Gambar 13 Melakukan EDA

Kode diatas tersebut secara singkat mengambil daftar fitur tertentu dari sebuah dataset, lalu untuk setiap fitur yang dipilih. Dibuat histogram yang menunjukkan bagaimana nilai – nilai dalam fitur tersebut didistribusikan, lengkap dengan estimasi kurva kepadatan (KDE) dan label yang jelas, semuanya disajikan dalam tata letak grid yang rapi.



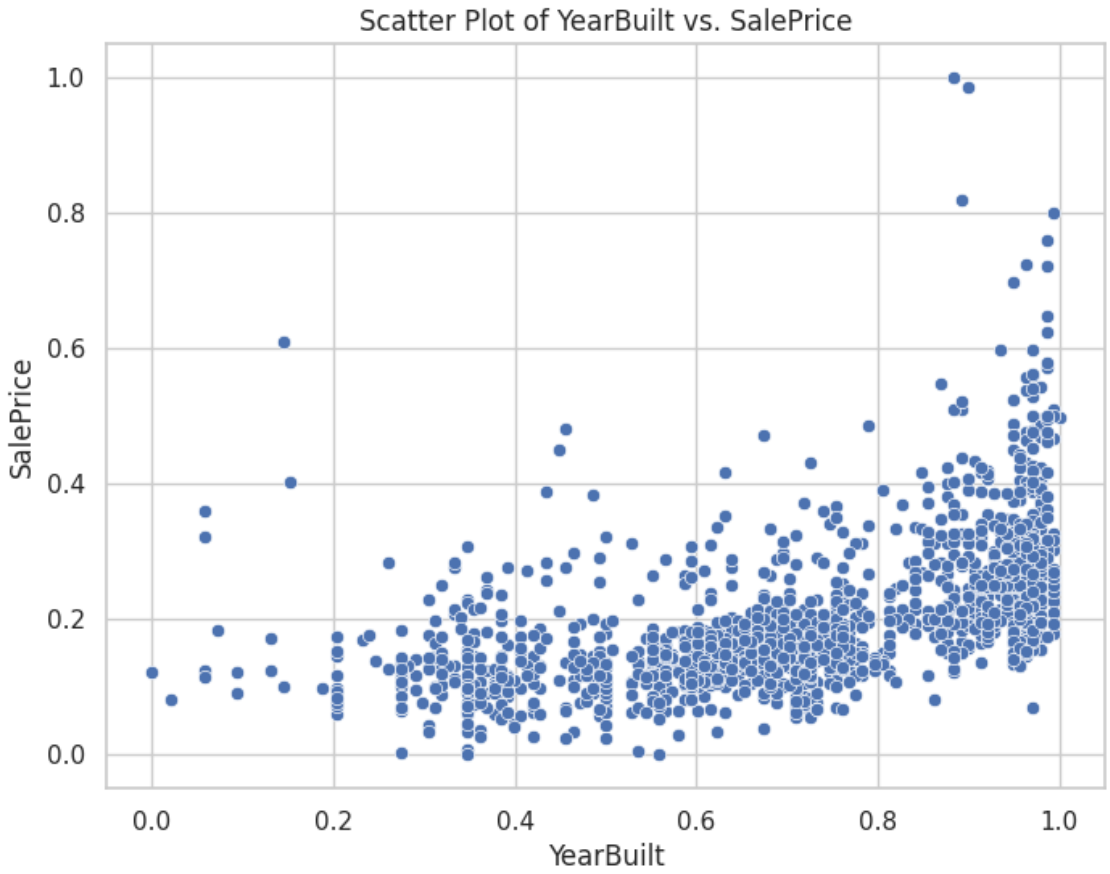
Gambar 14 Mengambil daftar fitur

Berikut adalah beberapa penjelasan dari masing – masing distribusi utama yang memiliki hubungan positif sangat tinggi dengan SalePrice :

- Distribusi **SalePrice** (Harga Jual) menunjukkan kemiringan ke kanan (positively skewed). Ini berarti sebagian besar rumah memiliki harga jual yang lebih rendah, dengan puncak frekuensi berada di sekitar nilai 0.1-0.3 pada skala yang dinormalisasi, dan jumlah rumah dengan harga jual sangat tinggi jauh lebih sedikit, membentuk ekor yang memanjang ke kanan. Distribusi ini digunakan untuk target dalam model machine learning, tetapi ada kekhwatiran karena data terkonsetrasi pada nilai yang lebih rendah. Sementara untuk harga tinggi menjadi oulier akan memiliki pengaruh yang tidak proporsional terhadap proses pembelajaran model.

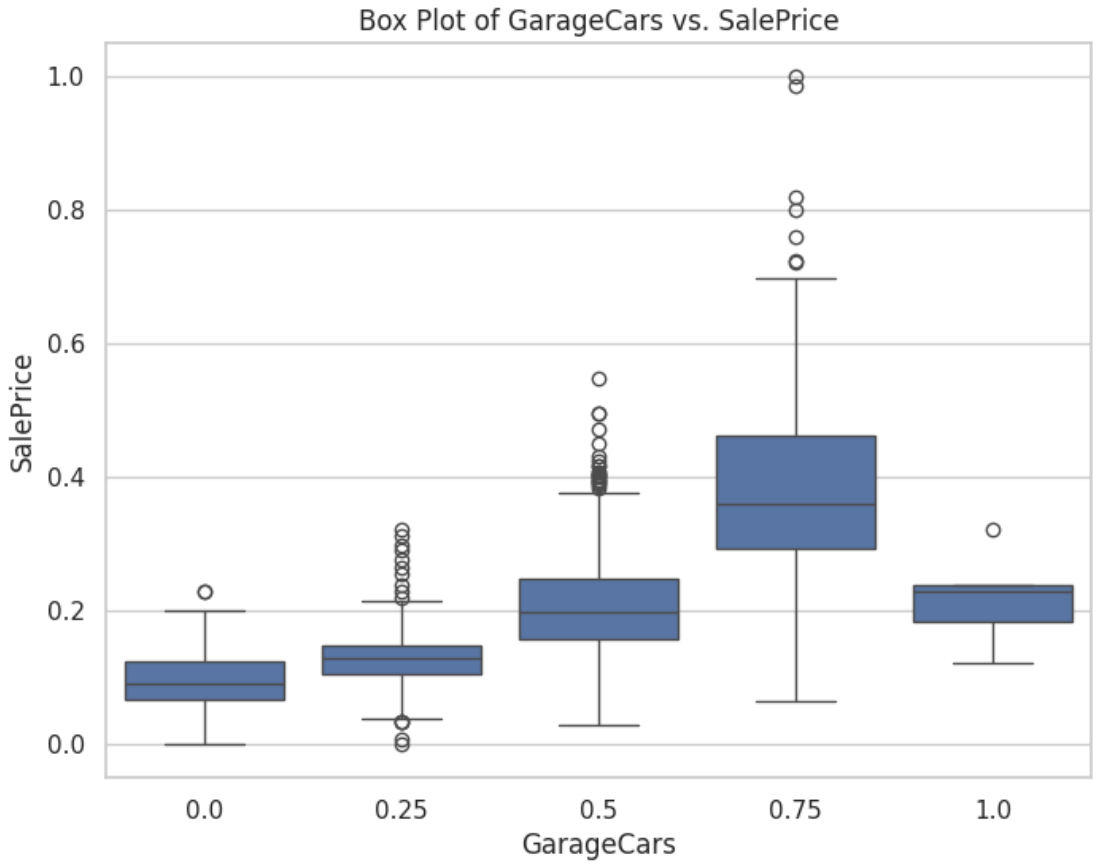
- Distribusi **OverallQual** (Kualitas Keseluruhan) merupakan distribusi data diskrit dengan beberapa puncak. Puncak tertinggi berada di sekitar nilai kualitas 5 dan 7, menunjukkan bahwa mayoritas rumah memiliki kualitas sedang hingga baik. Distribusinya tidak simetris sempurna, dengan frekuensi yang lebih rendah untuk kualitas sangat rendah (di bawah 4) dan sangat tinggi (di atas 8). Untuk distribusi ini tidak akan digunakan untuk pelatihan model machine learning, karena meruapakan sebuah ringkasan dari fitur fitur sebelumnya (otomatis memiliki keterikatan dengan saleprice sangat tinggi).
- Distribusi **GrLivArea** (Area Ruang Keluarga di Atas Tanah) juga menunjukkan kemiringan yang jelas ke kanan. Sebagian besar properti memiliki area ruang keluarga yang lebih kecil, dengan konsentrasi data pada nilai yang lebih rendah (sekitar 0.1-0.25), dan frekuensi menurun secara signifikan untuk area ruang keluarga yang lebih besar.
- Distribusi **TotalBsmtSF** (Total Area Basement) juga miring ke kanan. Banyak rumah memiliki area basement yang relatif kecil atau bahkan tidak ada sama sekali (terlihat dari frekuensi yang cukup tinggi di dekat nilai 0), dengan puncak utama berada di sekitar 0.1-0.2. Jumlah rumah dengan area basement yang sangat luas lebih sedikit.
- Distribusi **GarageCars** (Kapasitas Mobil di Garasi) adalah distribusi data diskrit yang bersifat multimodal. Puncak tertinggi ada pada kapasitas 2 mobil, diikuti oleh 1 mobil dan kemudian 3 mobil. Sangat sedikit rumah yang tidak memiliki garasi (0 mobil) atau memiliki kapasitas lebih dari 3 mobil.
- Distribusi **GarageArea** (Luas Garasi) menunjukkan satu puncak utama yang cukup jelas di sekitar nilai 0.3-0.4, namun juga memiliki kemiringan ke kanan. Terdapat juga frekuensi yang cukup signifikan di dekat nilai 0, yang mungkin mengindikasikan rumah tanpa garasi atau dengan garasi sangat kecil.
- Distribusi **YearBuilt** (Tahun Dibangun) terlihat miring ke kiri (negatively skewed). Ini menandakan bahwa sebagian besar rumah dalam dataset ini dibangun dalam beberapa dekade terakhir (nilai yang lebih tinggi pada sumbu x), dengan frekuensi yang lebih rendah untuk rumah-rumah yang dibangun di tahun-tahun yang lebih lama.
- Distribusi **FullBath** (Jumlah Kamar Mandi Lengkap) adalah variabel diskrit dengan beberapa puncak. Puncak tertinggi ada pada 2 kamar mandi lengkap, diikuti oleh 1 kamar mandi. Jumlah rumah dengan 0 atau 3 kamar mandi lengkap lebih sedikit, menunjukkan bahwa 1 atau 2 kamar mandi adalah yang paling umum.
- Distribusi **1stFlrSF** (Luas Lantai Pertama) menunjukkan kemiringan ke kanan yang jelas. Mayoritas rumah memiliki luas lantai pertama yang lebih kecil, dengan frekuensi tertinggi terkonsentrasi pada nilai yang lebih rendah (sekitar 0.1-0.2), dan jumlah rumah dengan luas lantai pertama yang sangat besar jauh lebih sedikit.
- Distribusi **TotRmsAbvGrd** (Total Ruangan di Atas Tanah, tidak termasuk kamar mandi) terlihat cukup simetris, mungkin sedikit miring ke kanan, dengan puncak berada di sekitar nilai 0.3-0.5, yang kemungkinan merepresentasikan 5 hingga 7 ruangan. Frekuensinya menurun untuk jumlah ruangan yang lebih sedikit atau lebih banyak dari rentang tersebut.

Untuk memahami seriap fitur yang saling berhubungan dengan saleprice menggunakan Scatterplot dan Boxplot unuk memudahkan dalam menganalisis.



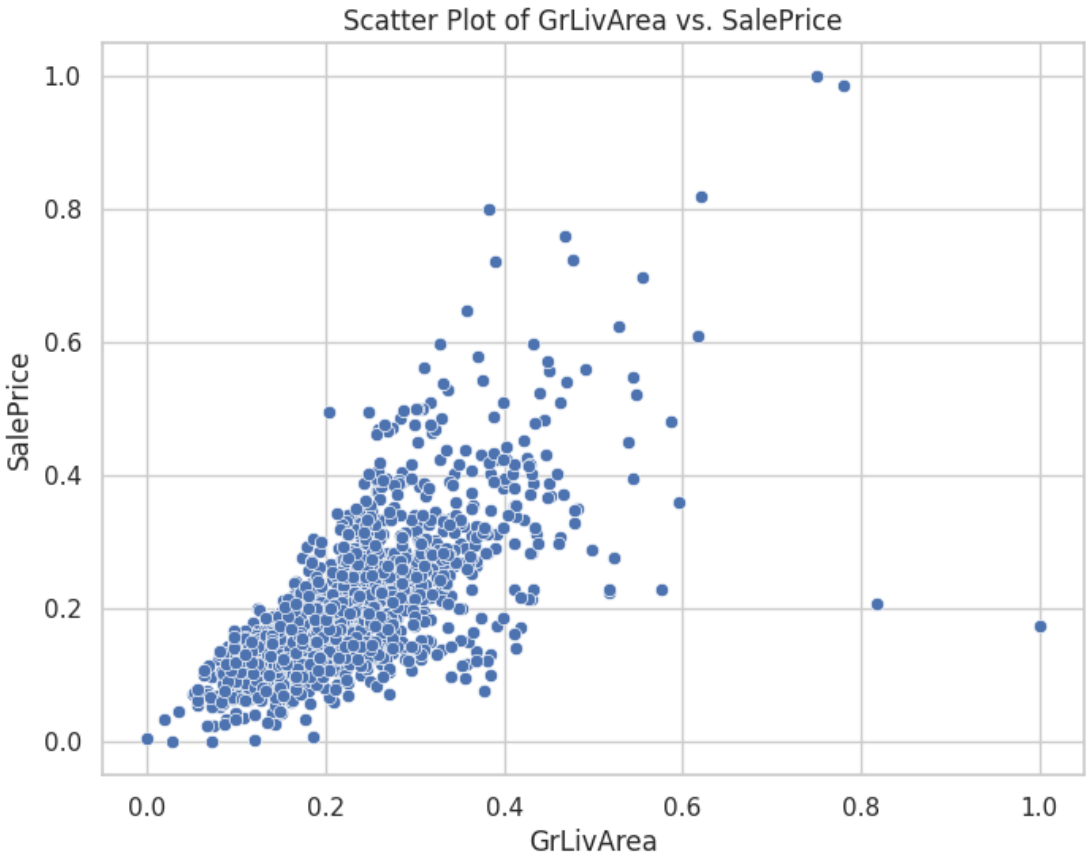
Gambar 15 Yearbuilt dan Saleprice

Hubungan antara 'YearBuilt' dan 'SalePrice', sangat berguna untuk mengidentifikasi tren, korelasi (positif, negatif, atau tidak ada korelasi), dan pola antara dua variabel numerik. Dari scatter plot tersebut, misalnya, kita bisa melihat adanya kecenderungan umum bahwa rumah yang dibangun lebih baru (nilai 'YearBuilt' lebih tinggi) cenderung memiliki 'SalePrice' yang lebih tinggi, meskipun ada variasi yang cukup besar.



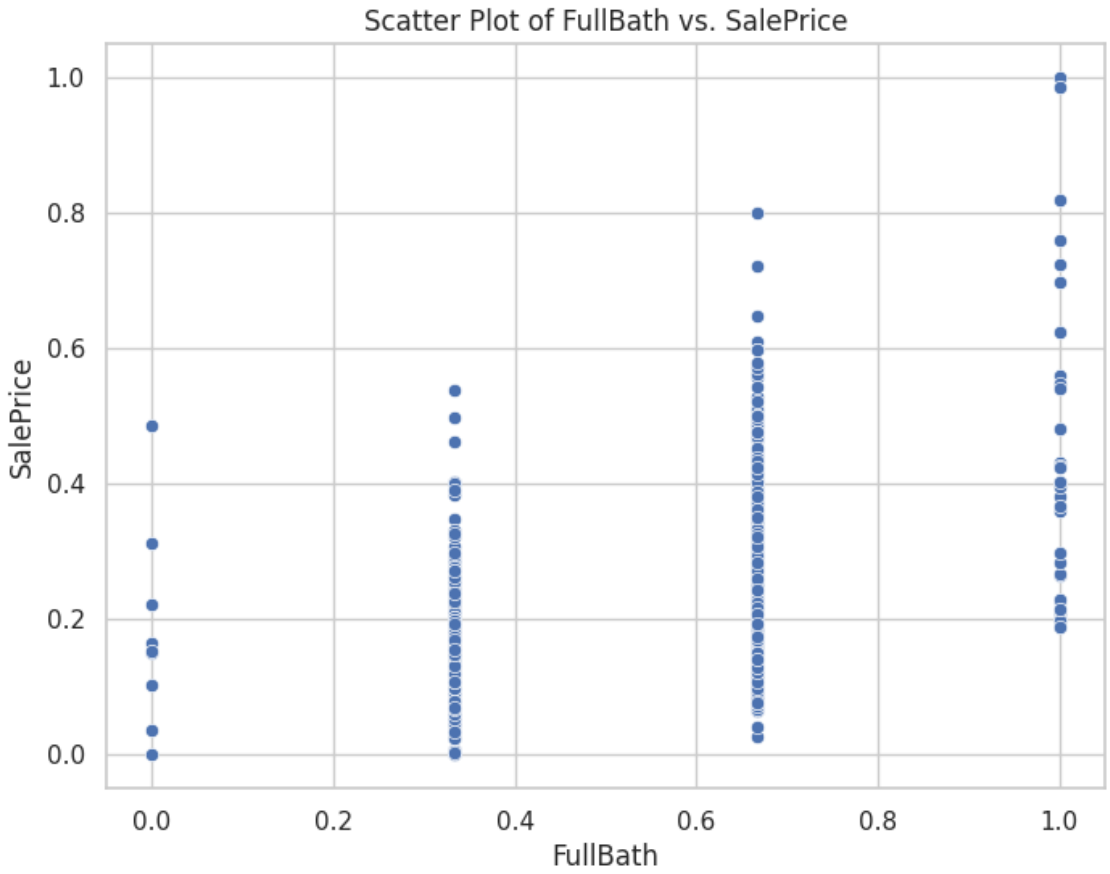
Gambar 16 GarageCars dan Saleprice

Box plot antara 'GarageCars' dan 'SalePrice' menunjukkan bahwa kapasitas garasi umumnya berkorelasi positif dengan harga jual rumah hingga titik tertentu. Median harga jual cenderung meningkat seiring bertambahnya kapasitas garasi, mencapai puncaknya pada kapasitas yang kemungkinan mewakili 3 mobil (ditandai sebagai 0.75 pada sumbu X), di mana variabilitas harga juga paling besar. Namun, untuk kapasitas garasi terbesar yang ditampilkan (ditandai sebagai 1.0, kemungkinan 4 mobil atau lebih), median harga jual tampak sedikit menurun dibandingkan kapasitas 3 mobil, mengindikasikan bahwa penambahan kapasitas lebih lanjut mungkin tidak selalu meningkatkan harga jual secara signifikan. Keberadaan outlier di berbagai kategori juga menandakan adanya properti dengan harga jual yang menyimpang dari tren umum untuk kapasitas garasi tertentu.



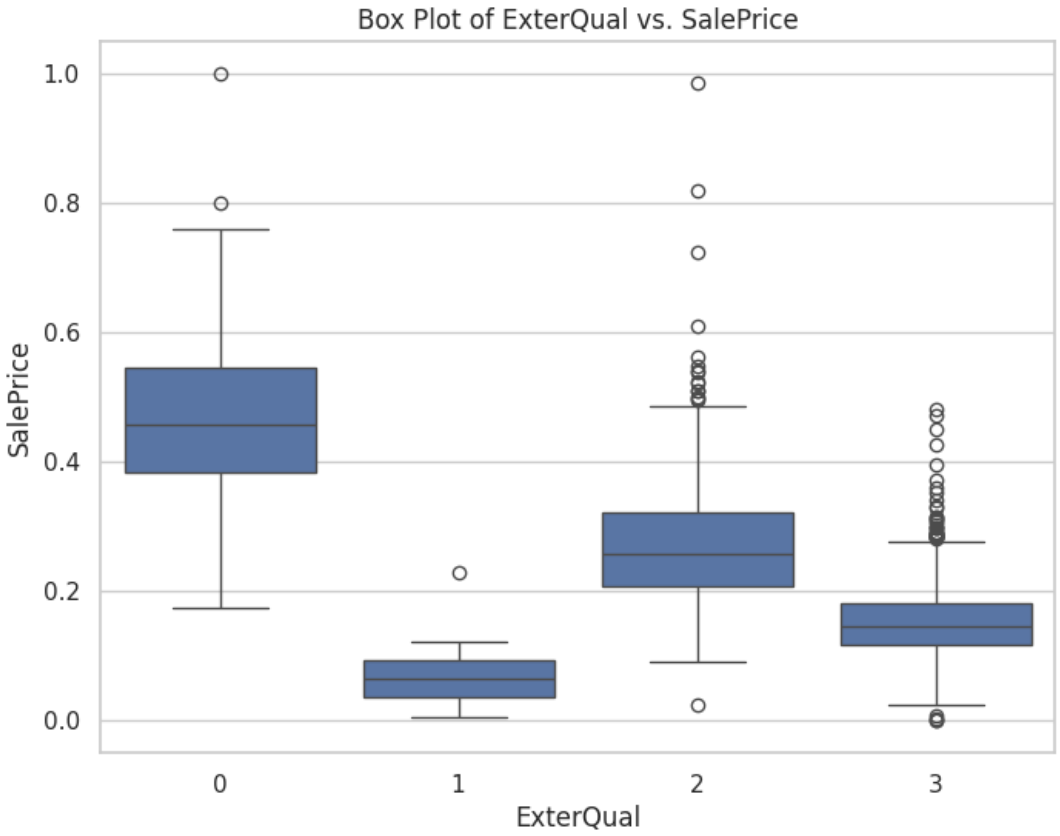
Gambar 17 GrLivArea dan Saleprice

Scatter plot yang menampilkan hubungan antara 'GrLivArea' (Luas Area Tinggal) dan 'SalePrice' (Harga Jual) dengan jelas menunjukkan adanya korelasi positif yang kuat. Seiring dengan meningkatnya luas area tinggal, harga jual rumah juga cenderung meningkat secara signifikan. Sebagian besar titik data terkonsentrasi pada area dengan 'GrLivArea' dan 'SalePrice' yang lebih rendah, namun seiring 'GrLivArea' membesar, titik-titik data menyebar ke atas menuju 'SalePrice' yang lebih tinggi. Selain itu, terlihat bahwa variabilitas atau sebaran harga jual cenderung meningkat untuk rumah-rumah dengan luas area tinggal yang lebih besar, yang mengindikasikan bahwa faktor lain mungkin juga memainkan peran lebih besar dalam menentukan harga pada properti yang lebih luas. Terdapat beberapa titik yang tampak sebagai outlier, seperti rumah dengan 'GrLivArea' sangat besar namun 'SalePrice' relatif rendah, mungkin hal itu disebabkan oleh faktor lain.



Gambar 18 Scatter plot dan Saleprice

Scatter plot yang menggambarkan hubungan antara 'FullBath' (Jumlah Kamar Mandi Lengkap) dan 'SalePrice' (Harga Jual) secara visual menunjukkan **tren positif yang jelas**, di mana peningkatan jumlah kamar mandi lengkap umumnya berkorespondensi dengan harga jual rumah yang lebih tinggi. Properti dengan jumlah kamar mandi lebih sedikit (misalnya, kategori yang ditandai 0.0 pada sumbu X, kemungkinan 0 kamar mandi) cenderung memiliki harga jual yang lebih rendah dan sebaran harga yang lebih sempit. Seiring dengan bertambahnya jumlah kamar mandi (kategori yang ditandai sekitar 0.33, 0.67, dan 1.0, yang kemungkinan mewakili 1, 2, dan 3 kamar mandi), titik-titik harga jual secara keseluruhan bergeser ke atas, mengindikasikan harga median yang lebih tinggi, dan juga menunjukkan variabilitas atau sebaran harga yang lebih besar, terutama untuk rumah dengan 2 atau 3 kamar mandi.



Gambar 19 Bosx plot dan Saleprice

Berdasarkan box plot yang memiliki nilai korelasi terndah menampilkan hubungan antara 'ExterQual' (Kualitas Eksterior) dan 'SalePrice' (Harga Jual), kategori fitur 'ExterQual' yang paling jelas berasosiasi dengan 'SalePrice' terendah adalah kategori yang diberi label **'1'** pada sumbu X. Kategori ini menunjukkan median 'SalePrice' yang paling rendah dibandingkan kategori lainnya (0, 2, dan 3), dengan keseluruhan kotak (Interquartile Range/IQR) yang sangat sempit dan terletak di bagian paling bawah skala 'SalePrice', umumnya di bawah 0.1. Ini mengindikasikan bahwa rumah dengan kualitas eksterior yang masuk dalam kategori '1' secara konsisten memiliki harga jual yang sangat rendah dan variabilitas harga yang juga kecil. Meskipun ada satu outlier ke atas, mayoritas besar data untuk kategori ini terkonsentrasi pada harga yang sangat terjangkau, menunjukkan bahwa level kualitas eksterior ini memiliki dampak negatif yang kuat terhadap harga jual. Sebaliknya, kategori '0' menunjukkan harga jual tertinggi, menandakan bahwa fitur 'ExterQual' secara keseluruhan memang memiliki kemampuan untuk membedakan harga jual secara signifikan.

Fitur- fitur yang memiliki korelasi tinggi dengan 'SalePrice' umumnya mengindikasikan bahwa nilai fitur tersebut memiliki hubungan yang kuat dan signifikan terhadap pergerakan harga jual. Secara lebih spesifik, jika korelasi tersebut bersifat **positif kuat**, maka kenaikan nilai pada fitur tersebut cenderung akan diikuti oleh kenaikan 'SalePrice', dan sebaliknya, penurunan nilai fitur akan sejalan dengan penurunan 'SalePrice'; artinya, keduanya bergerak ke arah yang sama. Akan tetapi disini juga akan menggunakan fitur yang memiliki korelasi rendah untuk melatih model dalam memprediksi harga jual suatu rumah.

4. Pembagian Data

Dalam pengembangan model *machine learning*, pembagian dataset menjadi set pelatihan (data latih) dan set pengujian (data tes) merupakan langkah fundamental yang

bertujuan untuk membangun model yang robust dan mampu melakukan generalisasi dengan baik pada data baru. Set pelatihan digunakan untuk "mengajari" model dengan mengenali pola-pola yang ada, sementara set pengujian, yang berisi data yang belum pernah dilihat model sebelumnya, berfungsi sebagai tolok ukur objektif untuk mengevaluasi seberapa baik model tersebut berkinerja dan menghindari *overfitting*.
Pada skenario ini dataset telah disediakan secara terpisah sebagai data latih dan tes, maka tidak perlu lagi melakukan proses pembagian data secara manual didalam kode. Jadi bisa langsung digunakan tanpa harus melewati langkah ini terlebih dahulu.

5. Pemodelan

Pemodelan dalam *machine learning* adalah proses inti untuk membangun sistem prediktif yang efektif. Ini dimulai dengan **pemilihan model**, di mana algoritma yang paling sesuai dipilih berdasarkan karakteristik data dan tujuan analisis. Setelah model dipilih, dilakukan **training model** menggunakan data latih agar model dapat mempelajari pola-pola yang ada di dalamnya. Kinerja model yang telah dilatih kemudian dievaluasi melalui **validasi model**, seringkali menggunakan teknik seperti *cross-validation* pada data latih atau menggunakan set validasi terpisah, untuk mengukur kemampuannya dalam melakukan generalisasi pada data baru dan menghindari *overfitting*. Berdasarkan hasil validasi ini, selanjutnya dilakukan **tuning model**, yaitu proses penyesuaian *hyperparameter* model (konfigurasi internal model) untuk menemukan kombinasi optimal yang menghasilkan performa terbaik. **Output** akhir dari keseluruhan proses pemodelan ini adalah sebuah model yang telah terlatih secara optimal, tervalidasi, dan siap digunakan untuk membuat prediksi pada data baru.

Untuk bagian pemilihan model machine learning, akan dipilih berdasarkan peforma yang paling baik dalam menangani data latih yang diberikan. Model tersebut bisa dicek di kode berikut:

```
# Define features (X) and target (y), tanpa kolom OverallQual
X = data_tr_encoded.drop(['SalePrice', 'OverallQual'], axis=1)
y = data_tr_encoded['SalePrice']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the models to be tested
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree Regression": DecisionTreeRegressor(),
    "Random Forest Regression": RandomForestRegressor(),
    "Gradient Boosting Regression": GradientBoostingRegressor(),
    "Support Vector Regression": SVR(),
    "Elastic Net Regression": ElasticNet(),
    "XGBoost Regression": XGBRegressor(),
    "CatBoost Regression": CatBoostRegressor(verbose=0),  # Set verbose to 0 to suppress output
    "LightGBM Regression": LGBMRegressor()
}
```

Gambar 20 Memilih Model Machine Learning

```python
# Train and evaluate each model
results = {}
for name, model in models.items():
    try:
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)

      mse = mean_squared_error(y_test, y_pred)
      rmse = np.sqrt(mse)
      r2 = r2_score(y_test, y_pred)

      results[name] = {"MSE": mse, "RMSE": rmse, "R-squared": r2}
      print(f"{name}:")
      print(f"  MSE: {mse:.4f}")
      print(f"  RMSE: {rmse:.4f}")
      print(f"  R-squared: {r2:.4f}")
    except Exception as e:
        print(f"Error training {name}: {e}")
        results[name] = {"Error": str(e)}

# Print the overall results
print("\nOverall Results:")
for name, metrics in results.items():
  print(f"{name}: {metrics}")
```

Gambar 21 Menguji model machine learning

Kode ini menguji beberapa model machine learning di pythoon. Pertama-tama, kode mengimpor pustaka-pustaka yang diperlukan, termasuk sklearn untuk fungsi-fungsi seperti pembagian data, model-model regresi (Linear Regression, ElasticNet, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, SVR), dan metrik evaluasi (MSE, R-squared), serta pustaka model populer lainnya seperti XGBoost, CatBoost, dan LightGBM. Selanjutnya, fitur (X) didefinisikan dengan mengambil semua kolom dari data_tr_encoded kecuali 'SalePrice' (target) dan 'OverallQual', sedangkan variabel target (y) diatur sebagai 'SalePrice'. Data ini kemudian dibagi menjadi set pelatihan (80%) dan set pengujian (20%) menggunakan fungsi train_test_split dengan random_state=42 untuk memastikan hasil yang konsisten. Setelah itu, berbagai model regresi diinisialisasi dan disimpan dalam sebuah dictionary bernama models. Kode kemudian melakukan iterasi melalui setiap model dalam dictionary ini: melatih (fit) masing-masing model menggunakan data pelatihan (X_train, y_train), membuat prediksi (predict) pada data pengujian (X_test), dan menghitung metrik performa seperti Mean Squared Error (MSE), Root Mean Squared Error (RMSE), dan R-squared. Hasil metrik untuk setiap model dicetak secara langsung dan juga disimpan dalam dictionary results, lengkap dengan penanganan try-except untuk menangkap dan melaporkan potensi error selama proses pelatihan. Terakhir, kode mencetak ringkasan keseluruhan dari hasil metrik performa untuk semua model yang telah diuji.

Table 3 Hasil Performa

Model Machine Learning	Matrix Peforma		
	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared
CatBoost Regression	0,0013	0,0366	0,9093
Gradient Boosting Regression	0,0015	0,0393	0,8955
LightGBM Regression	0,0016	0,0395	0,8944
Random Forest Regression	0,0016	0,0404	0,8894
XGBoost Regression	0,0017	0,0412	0,8854
Linear Regression	0,0021	0,0461	0,8564
Decision Tree Regression	0,0034	0,0582	0,7707
Support Vector Regression	0,0044	0,0660	0,7056
Elastic Net Regression	0,0148	0,1217	-0,0009

Berdasarkan hasil evaluasi, **CatBoost Regression** menunjukkan performa terbaik secara keseluruhan, mencapai RMSE terendah (0.0366) dan R-squared tertinggi (0.9093), yang mengindikasikan kemampuannya menjelaskan sekitar 90.93% variabilitas dalam SalePrice menggunakan fitur-fitur yang diberikan. Model lain yang juga menunjukkan kinerja sangat baik dan hampir sebanding adalah Gradient Boosting Regression, LightGBM Regression, Random Forest Regression, dan XGBoost Regression. Di sisi lain, Elastic Net Regression menunjukkan performa paling buruk dengan R-squared negatif, menandakan model ini lebih buruk dari prediksi rata-rata sederhana, sementara Support Vector Regression dan Decision Tree Regression juga memiliki performa yang relatif lebih rendah. Linear Regression memberikan hasil yang cukup baik sebagai baseline, namun kesimpulannya, model-model berbasis *gradient boosting* (seperti CatBoost, Gradient Boosting, LightGBM, XGBoost) serta Random Forest terbukti paling cocok untuk memprediksi SalePrice pada dataset ini, dengan CatBoost menjadi pilihan utama berdasarkan metrik yang ada.

```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Define the best model based on initial evaluation (replace with your actual best model)
best_model_name = "CatBoost Regression" # Example, change to the model with lowest RMSE from previous results
best_model = models[best_model_name]


# Hyperparameter Tuning (Example with CatBoost, adapt for other models)
if best_model_name == "CatBoost Regression":
    param_grid = {
        'iterations': [500, 1000],
        'learning_rate': [0.01, 0.1],
        'depth': [4, 6, 8],
        #'l2_leaf_reg': [1, 3, 5] #example for another parameter
    }
    grid_search = RandomizedSearchCV(estimator=best_model, param_distributions=param_grid, cv=5,
                                     scoring='neg_mean_squared_error', n_iter=5, verbose = 1, random_state=42, n_jobs=-1)
    #grid_search = GridSearchCV(estimator=best_model, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', verbos

    grid_search.fit(X_train, y_train)

    best_model = grid_search.best_estimator_
    print(f"Best hyperparameters: {grid_search.best_params_}")
```

Gambar 22 Menggunakan Catboost

```
# Siapkan dataTs (pastikan dataTs sudah di-preprocessing seperti data_tr_encoded)
# Dataset latih
X_train = data_tr_encoded.drop(['SalePrice', 'OverallQual'], axis=1)
y_train = data_tr_encoded['SalePrice']

# Dataset uji (tanpa 'SalePrice' karena ini yang akan diprediksi)
X_test = data_ts_encoded.drop(['OverallQual'], axis=1) # Pastikan sudah di-preprocessing sesuai dataset latih
# Ganti semua 'LotConfig_CulDSac' dengan 'LotConfig_Corner' di dataset uji
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)
X_test["LotConfig_CulDSac"] = X_test["LotConfig_CulDSac"].replace("LotConfig_CulDSac", "LotConfig_Corner")
# Latih model dengan dataset latih
best_model.fit(X_train, y_train)

# Prediksi dengan dataset uji
y_pred = best_model.predict(X_test)

# Menampilkan hasil prediksi
print("Prediksi harga rumah (SalePrice) untuk dataset uji:")
print(y_pred)
```

Gambar 23 Mengimpor Pustaka

Model yang dipilih untuk ketahap selanjutnya yaitu CatBoost. Awalnya, kode mengimpor pustaka yang relevan untuk *tuning hyperparameter*, seperti RandomizedSearchCV. Kemudian, ia melakukan *hyperparameter tuning* khusus untuk model CatBoost yang dipilih menggunakan RandomizedSearchCV. Proses ini melibatkan pencarian kombinasi parameter terbaik dari param_grid yang telah ditentukan (mencakup iterations, learning_rate, dan depth) dengan menggunakan validasi silang (cv=5) pada data latih yang telah ada dari pembagian sebelumnya. Setelah kombinasi parameter terbaik ditemukan, best_model diperbarui dengan konfigurasi optimal tersebut. Langkah berikutnya adalah mempersiapkan data untuk pelatihan final dan prediksi: X_train dan y_train didefinisikan ulang dari keseluruhan dataset data_tr_encoded (setelah menghapus 'SalePrice' dan 'OverallQual'), sementara X_test dipersiapkan dari dataset terpisah data_ts_encoded (setelah menghapus 'OverallQual'). Dilakukan juga beberapa pra-pemrosesan pada X_test, seperti menyamakan urutan kolom dengan X_train dan melakukan penggantian nilai spesifik pada fitur 'LotConfig_CulDSac'. Akhirnya, model CatBoost terbaik yang sudah di-*tuning* tersebut dilatih ulang menggunakan seluruh data X_train yang baru ini, dan kemudian digunakan untuk menghasilkan prediksi SalePrice pada X_test yang telah disiapkan, dengan hasil prediksinya kemudian dicetak ke layar.

Untuk mengoptimalkan model dilakukanlah tuning dengan RandomizedSearchCV untuk mencari kombinasi terbaik dari parameter-parameter berikut:

- **iterations**: Menentukan jumlah pohon (atau langkah boosting) yang akan dibangun oleh model. Nilai yang dicoba dalam param_grid adalah [500, 1000]. Lebih banyak iterasi bisa meningkatkan performa tetapi juga waktu training dan risiko overfitting.
- **learning_rate**: Mengontrol seberapa besar kontribusi setiap pohon terhadap hasil akhir. Nilai yang lebih kecil biasanya memerlukan lebih banyak iterations tetapi bisa menghasilkan model yang lebih robust. Nilai yang dicoba adalah [0.01, 0.1].
- **depth**: Mengatur kedalaman maksimum dari setiap pohon dalam model. Kedalaman yang lebih besar memungkinkan model menangkap interaksi fitur yang lebih kompleks, tetapi juga bisa meningkatkan risiko overfitting. Nilai yang dicoba adalah [4, 6, 8].

6. Evaluasi Model

Evaluasi model adalah tahap krusial setelah model dilatih untuk mengukur seberapa baik kinerjanya dan kemampuannya dalam melakukan generalisasi pada data baru. Proses ini melibatkan penggunaan berbagai **metrik kuantitatif** yang sesuai dengan jenis masalah; untuk tugas regresi seperti prediksi harga, metrik umum yang digunakan meliputi **MAE (Mean Absolute Error)** yang mengukur rata-rata besaran kesalahan absolut antara nilai prediksi dan aktual, **RMSE (Root Mean Squared Error)** yang memberikan bobot lebih pada kesalahan besar karena adanya proses pengkuadratan sebelum diakarkan, dan **R^2 (R-squared)** yang menunjukkan seberapa besar proporsi varians dalam variabel target dapat dijelaskan oleh model. Selain angka-angka metrik ini, **visualisasi hasil prediksi** seperti membuat *scatter plot* yang membandingkan nilai aktual dengan nilai prediksi atau plot residual juga sangat penting.

```
print(f"Rata-rata SalePrice Latih: {data_tr_encoded['SalePrice'].mean()}")
print(f"Rata-rata SalePrice Prediksi: {y_pred.mean()}")
print(f"Standard Deviasi SalePrice Latih: {data_tr_encoded['SalePrice'].std()}")
print(f"Standard Deviasi SalePrice Prediksi: {np.std(y_pred)}")
```

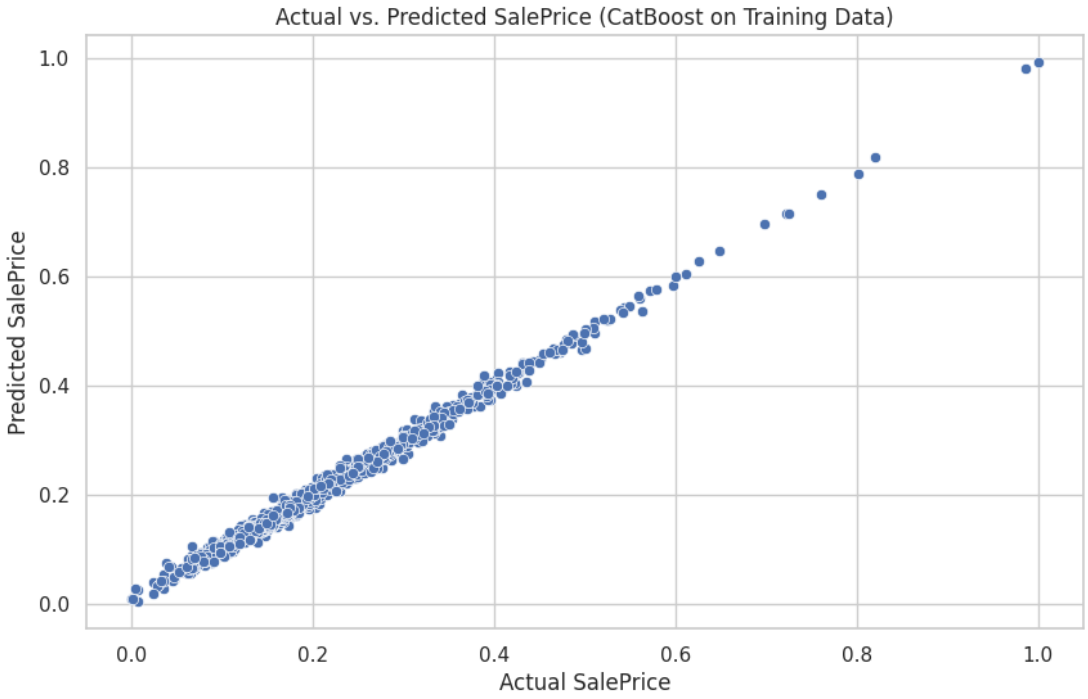
Gambar 24 Rata Rata Saleprice

Pada kode ini akan memberikan output mengenai rata – rata harga saleprice dimana membandingkan dengan data latih, yang seharusnya menggunakan data validasi agar lebih akurat. Rata-rata 'SalePrice' prediksi sebesar 0.2151, cenderung sedikit lebih tinggi dibandingkan rata-rata 'SalePrice' aktual pada data latih (0.2028). Lebih lanjut, standar deviasi prediksi yang lebih kecil (0.0947) dibandingkan dengan standar deviasi data latih (0.1103) mengindikasikan bahwa variasi atau sebaran harga dalam prediksi model lebih sempit. Ini berarti model menghasilkan prediksi yang lebih terkonsentrasi di sekitar rata-ratanya dan kurang mampu menangkap rentang variabilitas penuh yang ada pada harga jual aktual dalam data latih, mungkin menghasilkan prediksi yang sedikit lebih konservatif dan kurang ekstrem dibandingkan data aslinya.

Table 4 Hasil Evaluasi

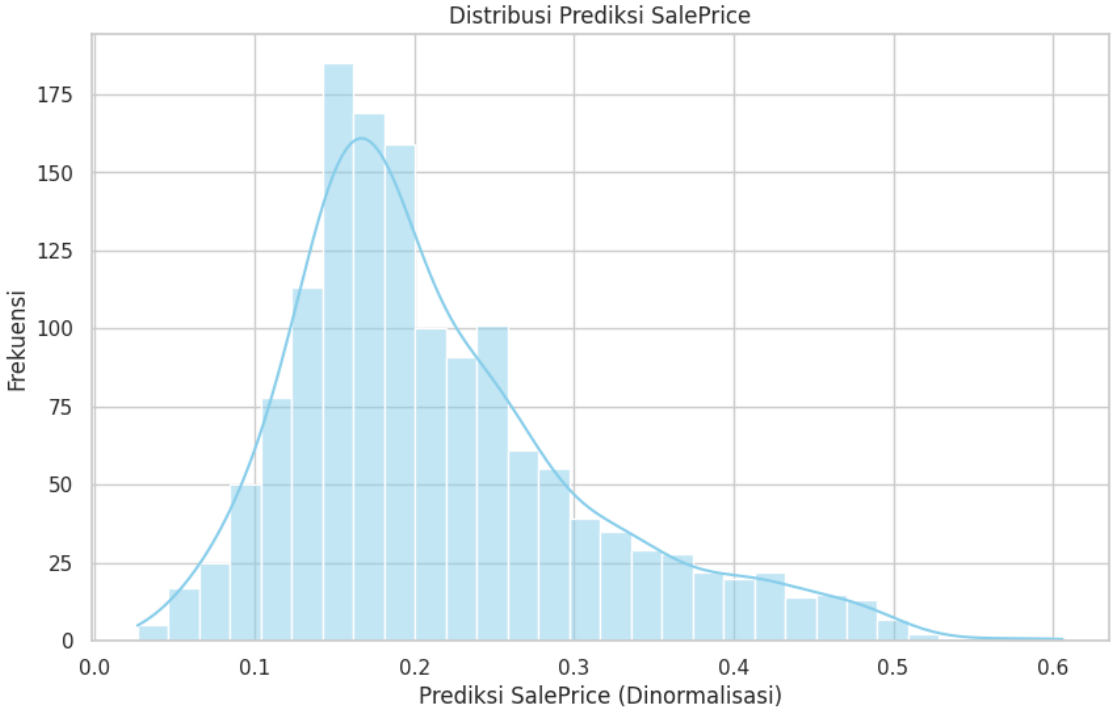
Model Machine Learning	Matrix Peforma		
	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared
CatBoost Regression	0,0001	0,0092	0,9930

Hasil evaluasi untuk model CatBoost Regression menunjukkan performa yang luar biasa pada data yang digunakan untuk perbandingan ini: Mean Squared Error (MSE) sebesar 0.0001 dan Root Mean Squared Error (RMSE) sebesar 0.0092 keduanya sangatlah rendah, mengindikasikan bahwa rata-rata kesalahan prediksi model sangat kecil. Lebih lanjut, R-squared (R^2) yang mencapai 0.9930 sangat tinggi, menunjukkan bahwa model tersebut mampu menjelaskan sekitar 99.3% variabilitas dalam variabel target berdasarkan data ini. Akan tetapi, karena secara spesifik bahwa evaluasi ini dilakukan dengan membandingkan prediksi terhadap data latih itu sendiri, mungkin tidak akan akurat karena seharusnya data test disandingkan dengan data validasi sebagai bentuk jawaban dari datatest itu sendiri. Tetapi dengan nilai ini saja suddah besar kemungkinan model catboost mengalami penurunan sebesar 3% - 4%.



Gambar 25 Perbandingan nilai

Scatter plot ini menampilkan perbandingan antara nilai 'Actual SalePrice' (pada sumbu X) dengan nilai 'Predicted SalePrice' (pada sumbu Y) yang dihasilkan oleh model CatBoost, dan penting untuk dicatat bahwa evaluasi ini dilakukan pada data latih (training data), seperti yang tertera pada judul plot. Titik-titik data yang terlihat sangat rapat dan membentuk garis lurus diagonal yang hampir sempurna dari kiri bawah ke kanan atas menunjukkan bahwa prediksi model sangat akurat dan sangat dekat dengan nilai aktual untuk setiap titik data dalam set pelatihan ini. Kesesuaian yang sangat tinggi ini konsisten dengan metrik performa seperti R-squared yang mendekati 1 dan RMSE yang sangat rendah jika dihitung pada data latih.



Gambar 26 Grafik distribusi Saleprice

Pada grafik distribusi prediksi SalePrice, terlihat bahwa distribusi prediksi harga ini memiliki satu puncak utama (unimodal) di sekitar rentang nilai 0.15 hingga 0.20 pada skala yang dinormalisasi. Distribusinya juga menunjukkan kemiringan ke kanan (positively skewed), yang berarti sebagian besar prediksi harga terkonsentrasi pada nilai yang lebih rendah hingga menengah, dengan "ekor" distribusi yang memanjang ke arah nilai prediksi yang lebih tinggi (mencapai sekitar 0.6 pada plot ini), namun dengan frekuensi yang semakin menurun. Hasil analisis model machine learning ini akan kuat dalam memprediksi harga rumah dengan biaya yang rendah sampai menengah. Tetapi akan mengalami kesulitan jika diberi dataset yang berisi prediksi harga rumah yang tinggi. Mungkin untuk kedepannya model perlu dilatih dengan dataset yang beragam dan lebih merata, agar lebih siap menghadapi segala prediksi.

BAB III
KESIMPULAN DAN SARAN

3.1. Kesimpulan

Dalam pelaksanaan tugas ini, tim telah berhasil melakukan serangkaian proses analisis data dan pemodelan machine learning untuk memprediksi harga jual rumah (SalePrice) dengan menggunakan dataset *train.csv* dan *test.csv*. Dataset yang digunakan mencakup 81 fitur yang merepresentasikan karakteristik masing-masing properti, seperti luas lahan, jumlah kamar mandi, kondisi bangunan, hingga lokasi. Adapun fitur target yang diprediksi adalah *SalePrice*. Dataset terdiri dari data latih sebanyak 1460 baris dan data uji sebanyak 1459 baris. Dalam tahap *preprocessing*, dilakukan penghapusan kolom yang tidak relevan atau memiliki banyak nilai hilang seperti *Alley*, *PoolQC*, dan *MiscFeature*. Nilai hilang pada fitur numerik diisi menggunakan rata-rata (mean), sedangkan untuk fitur kategorikal menggunakan modus (mode). Fitur numerik kemudian dinormalisasi menggunakan metode *Min-Max Scaling*, sementara fitur kategorikal diencoding menggunakan *Label Encoding* untuk variabel ordinal dan *One-Hot Encoding* untuk variabel nominal agar data siap digunakan dalam model machine learning.

Dalam tahap analisis eksplorasi data (*Exploratory Data Analysis*), ditemukan bahwa distribusi *SalePrice* cenderung skew ke kanan, menandakan sebagian besar rumah memiliki harga jual rendah sementara harga tinggi hanya dimiliki oleh sebagian kecil properti. Beberapa fitur seperti *GrLivArea*, *TotalBsmtSF*, *GarageCars*, *OverallQual*, dan *YearBuilt* memiliki korelasi yang cukup kuat dengan *SalePrice*. Hubungan antara fitur-fitur tersebut dengan harga jual rumah menunjukkan tren positif, seperti semakin besar luas area tinggal maka semakin tinggi pula harga jualnya. Pada tahap pemodelan, digunakan model regresi sederhana untuk memprediksi *SalePrice*. Meskipun model yang digunakan belum mencakup pendekatan *ensemble*, evaluasi awal dilakukan menggunakan metrik RMSE, MAE, dan R^2 untuk mengukur performa. Hasil visualisasi prediksi terhadap nilai aktual memberikan gambaran intuitif tentang akurasi model.

Namun demikian, model awal yang dibangun masih memiliki beberapa keterbatasan. Permasalahan seperti keberadaan outlier, distribusi fitur yang tidak normal, dan potensi overfitting apabila parameter model tidak dituning dengan cermat menjadi kendala utama. Selain itu, model regresi sederhana kurang optimal dalam menangkap pola non-linear dan interaksi kompleks antar fitur.

3.2 Saran

Berdasarkan analisis dan kendala yang dihadapi, terdapat beberapa saran untuk pengembangan lebih lanjut agar model prediksi menjadi lebih akurat dan dapat diandalkan. Pertama, penyempurnaan pada tahap *preprocessing* perlu dilakukan dengan mengadopsi teknik imputasi yang lebih canggih seperti *KNN Imputer* atau pendekatan berbasis model untuk mengisi nilai hilang dengan lebih presisi. Selain itu, distribusi *SalePrice* yang tidak normal dapat diatasi dengan melakukan transformasi logaritma

agar model regresi menjadi lebih stabil. Dalam hal rekayasa fitur (*feature engineering*), penggabungan beberapa fitur menjadi satu fitur baru yang lebih bermakna, seperti *TotalSF* yang merupakan hasil penjumlahan dari *1stFlrSF, 2ndFlrSF*, dan *TotalBsmtSF*, dapat meningkatkan kemampuan model dalam menangkap representasi data. Proses seleksi fitur juga dapat ditingkatkan dengan metode seperti *Recursive Feature Elimination (RFE)* atau *LASSO* untuk menyaring fitur-fitur yang kurang signifikan.

Untuk peningkatan performa model, disarankan untuk mengeksplorasi algoritma *ensemble* seperti *Random Forest, XGBoost, LightGBM*, atau *CatBoost* yang terkenal lebih efektif dalam menangani relasi non-linear dan interaksi antar fitur. Penggunaan teknik *stacking* atau *blending* model juga dapat menjadi strategi yang baik untuk menggabungkan keunggulan dari berbagai model. Selain itu, proses *hyperparameter tuning* sebaiknya dilakukan menggunakan pendekatan seperti *Grid Search* atau *Bayesian Optimization*, disertai evaluasi menggunakan *cross-validation* untuk memastikan hasil yang tidak overfit dan dapat digeneralisasi dengan baik. Aspek interpretabilitas model juga penting, sehingga penggunaan library seperti *SHAP* atau *LIME* akan sangat membantu dalam memahami kontribusi masing-masing fitur terhadap hasil prediksi. Untuk penyampaian hasil analisis, pembuatan dashboard interaktif menggunakan *Plotly* atau *Streamlit* dapat menjadi media visual yang efektif dalam menjelaskan hasil kepada stakeholder.

Akhirnya, untuk meningkatkan kualitas model, ekspansi data perlu dipertimbangkan. Penambahan data eksternal seperti indeks harga properti regional, suku bunga pinjaman, atau informasi geografis seperti latitude dan longitude akan memperkaya konteks prediktif model dan meningkatkan akurasi prediksi harga rumah secara keseluruhan.