BY ALFIAN ABDUL GHAFFAR

# DAEGU APARTMENT PRICE PREDICTION

# Background

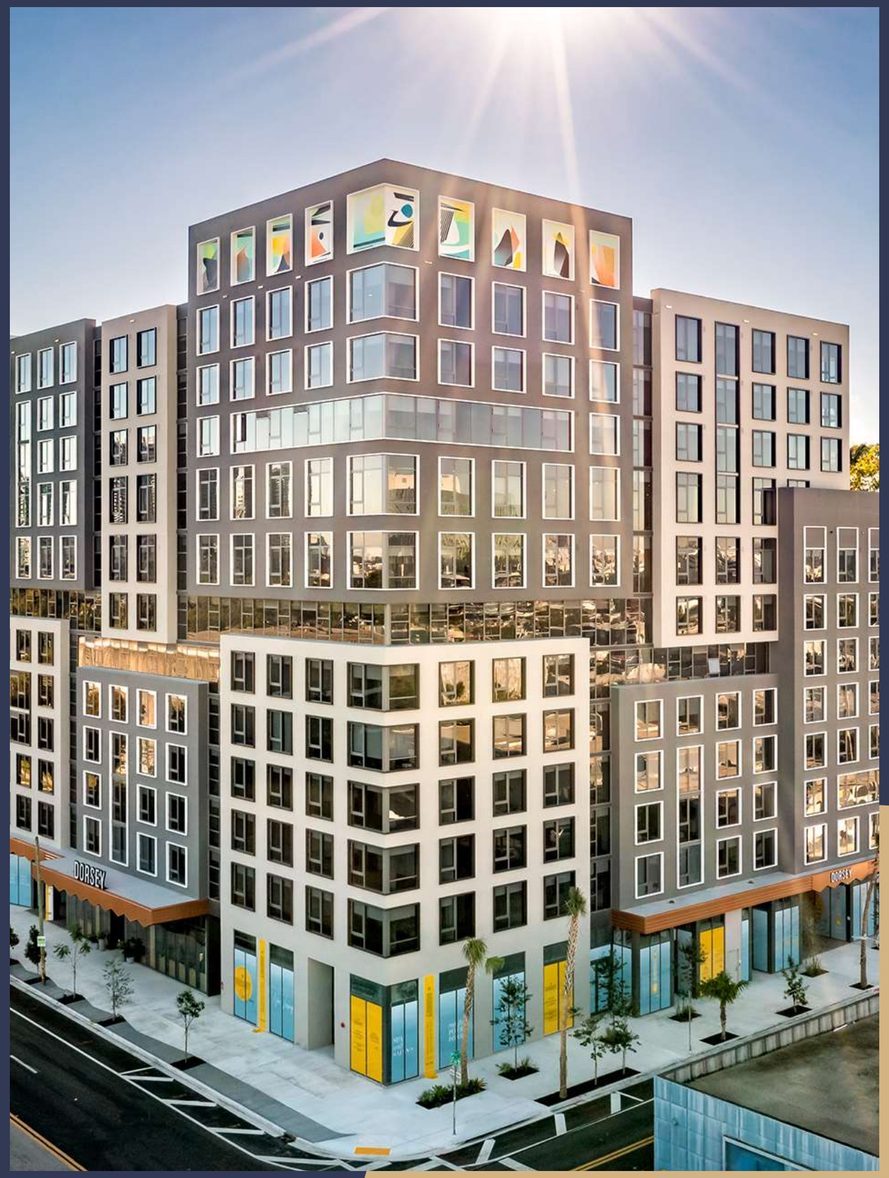
Daegu adalah sebuah kota besar di Korea Selatan. Kota ini merupakan kota terbesar keempat di Korea Selatan. Daegu memiliki populasi yang besar dan merupakan pusat ekonomi, budaya, metropolitan dan pendidikan. Hal tersebut membuat pertumbuhan penduduk dan ekonomi meningkat dengan pesat. Menurut worldpopulationreview.com populasi Daegu pada tahun 2023 diperkirakan mencapai 2.180.997 jiwa. Akibat padatnya jumlah penduduk namun dengan lahan tempat tinggal yang terbatas, membuat penduduk mengandalkan apartment sebagai pilihan untuk tempat tinggal. Pada tahun 2023 sejumlah 36000 keluarga memutuskan untuk mencari dan tinggal di apartment daerah Daegu. Karena perkembangan Daegu yang pesat, maka permintaan akan sewa apartemen pun meningkat.

# Problem Statement

Permintaan yang tinggi terhadap sewa apartment di kota Daegu menjadi tantangan bagi sebuah agen properti agar dapat bersaing dengan agen properti lainnya. Agar tetap dapat memperoleh customer yang loyal. Agen properti harus jeli dalam menentukan harga sewa yang sesuai dengan fasilitas dan fitur yang ditawarkan oleh sebuah unit apartment. Oleh karena itu dibutuhkan sebuah model yang dapat menjadi solusi untuk permasalahan tersebut, agar sebuah agen properti dapat menentukan harga sewa yang lebih kompetitif dan memudahkan untuk mencapai sales goal dan target.

# Goal

Perlu dibangun sebuah model yang dapat memprediksi harga sewa apartemen berdasarkan data yang disediakan, yaitu model yang layak dengan tingkat pembelajaran model terbaik dengan tingkat kesalahan seminimal mungkin.



# Project Stakeholder

Pada project ini pembuatan model prediksi ditujukan kepada stakeholder agen properti. Dimana tugas agen properti adalah membantu klien dalam proses penyewaan, pembelian atau penjualan properti, dalam hal ini adalah properti jenis apartment. Riset terhadap detail data apartment yang akan menjadi acuan untuk sewa apartment, seperti mencari apartment yang sesuai dengan kriteria klien agar customer mendapatkan harga competitive dalam melakukan sewa apartment.

# Analytics Approach

Untuk dapat menentukan harga sewa apartment yang tepat di kota Daegu. Kita akan melakukan analisa data untuk dapat menemukan pola dari fitur-fitur yang ada. Kita akan menggunakan semua fitur (kecuali harga jual) untuk memprediksi harga sewa setiap apartemen dengan menggunakan algoritma machine learning. Lalu, membangun sebuah model regresi yang akan membantu agen properti dalam menentukan harga sewa unit apartment.

# Problem Limitation

* Pada project Machine Learning kali ini merupakan project pertama, dimana menggunakan laptop sebagai media pemrosesan dan penyimpanan lokal, sehingga dengan perangkat memiliki kapasitas terbatas tersebut, maka harus berhati-hati dalam memilih model.
* Oleh karena itu model dengan tipe standalone akan diprioritaskan untuk dilakukan tuning dan optimisasi terlebih dahulu.

# Metric Evaluation

01 **RMSE (Root Mean Square Error)**

RMSE adalah metrik yang mengukur sejauh mana selisih antara nilai prediksi dan nilai aktual dalam bentuk akar rata-rata dari kuadrat selisih tersebut

02 **MAE (Mean Absolute Error)**

MAE adalah metrik yang mengukur rata-rata dari selisih absolut antara nilai prediksi dan nilai aktual.

03 **MAPE (Mean Absolute Percentage Error)**

MAPE adalah metrik yang mengukur rata-rata persentase selisih antara nilai prediksi dan nilai aktual.

# DATA UNDERSTANDING

# Dataset Information

| Attribute | Data Type, Length | Description |
|---|---|---|
| Hallway Type | Text | Tipe Apartment |
| TimeToSubway | Text | Waktu yang dibutuhkan ke stasiun subway terdekat |
| SubwayStation | Text | Nama stasiun subway terdekat |
| N_FacilitiesNearBy(ETC) | Float | Jumlah fasilitas terdekat |
| N_FacilitiesNearBy(PublicOffice) | Float | Jumlah fasilitas kantor publik terdekat |
| N_SchoolNearBy(University) | Float | Jumlah universitas terdekat |
| N_Parkinglot(Basement) | Float | Jumlah tempat parkir |
| YearBuilt | Integer | Tahun apartemen dibangun |
| N_FacilitiesInApt | Integer | Jumlah fasilitas di apartemen |
| Size(sqft) | Integer | Ukuran apartemen (dalam square feet) |
| SalePrice | Integer | Harga apartemen (Won) |

# Information Detail Dataset

| | feature | data_type | null_value | neg_value | n_unique | sample_unique |
|---|---|---|---|---|---|---|
| 0 | HallwayType | object | 0.0 | False | 3 | [terraced, mixed, corridor] |
| 1 | TimeToSubway | object | 0.0 | False | 5 | [0-5min, 10min~15min, 15min~20min, 5min~10min,... |
| 2 | SubwayStation | object | 0.0 | False | 8 | [Kyungbuk_uni_hospital, Chil-sung-market, Bang... |
| 3 | N_FacilitiesNearBy(ETC) | float64 | 0.0 | False | 4 | [0.0, 1.0, 5.0, 2.0] |
| 4 | N_FacilitiesNearBy(PublicOffice) | float64 | 0.0 | False | 8 | [3.0, 5.0, 7.0, 1.0, 4.0, 2.0, 6.0, 0.0] |
| 5 | N_SchoolNearBy(University) | float64 | 0.0 | False | 6 | [2.0, 1.0, 3.0, 4.0, 5.0, 0.0] |
| 6 | N_Parkinglot(Basement) | float64 | 0.0 | False | 20 | [1270.0, 0.0, 56.0, 798.0, 536.0, 605.0, 203.0... |
| 7 | YearBuilt | int64 | 0.0 | False | 16 | [2007, 1986, 1997, 2005, 2006, 2009, 2014, 199... |
| 8 | N_FacilitiesInApt | int64 | 0.0 | False | 9 | [10, 4, 5, 7, 2, 9, 8, 1, 3] |
| 9 | Size(sqf) | int64 | 0.0 | False | 89 | [1387, 914, 558, 1743, 1334, 572, 910, 288, 11... |
| 10 | SalePrice | int64 | 0.0 | False | 838 | [346017, 150442, 61946, 165486, 311504, 118584... |

# Change Column Name

| Old Column Name | New Column Name |
|---|---|
| Hallway Type | Hallway_Type |
| TimeToSubway | Time_To_Subway |
| SubwayStation | Subway_Station |
| N_FacilitiesNearBy(ETC) | ETCFacilities_Nearby |
| N_FacilitiesNearBy(PublicOffice) | PublicOffice_Nearby |
| N_SchoolNearBy(University) | University_Nearby |
| N_Parkinglot(Basement) | Basement_Parking |
| YearBuilt | Year_Built |
| N_FacilitiesInApt | Apt_Facilities |
| Size(sqf) | Size_Sqf |
| SalePrice | Sale_Price |

# Change Column Value

**Before**

```
# Kolom Time_To_Subway Before

df['Time_To_Subway'].value_counts()
```

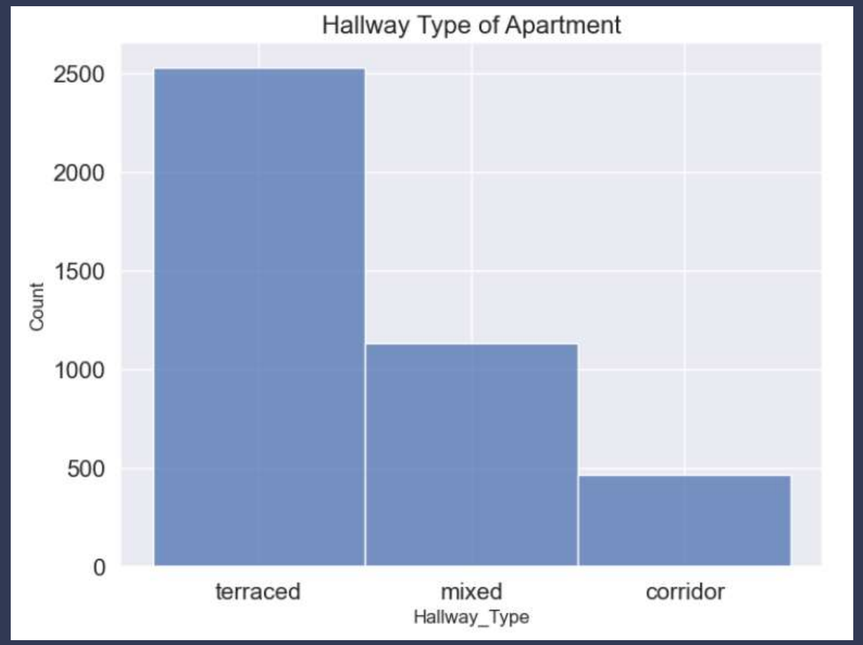
```
0-5min                1953
5min~10min             787
15min~20min            629
10min~15min            583
no_bus_stop_nearby     171
Name: Time_To_Subway, dtype: int64
```

**After**

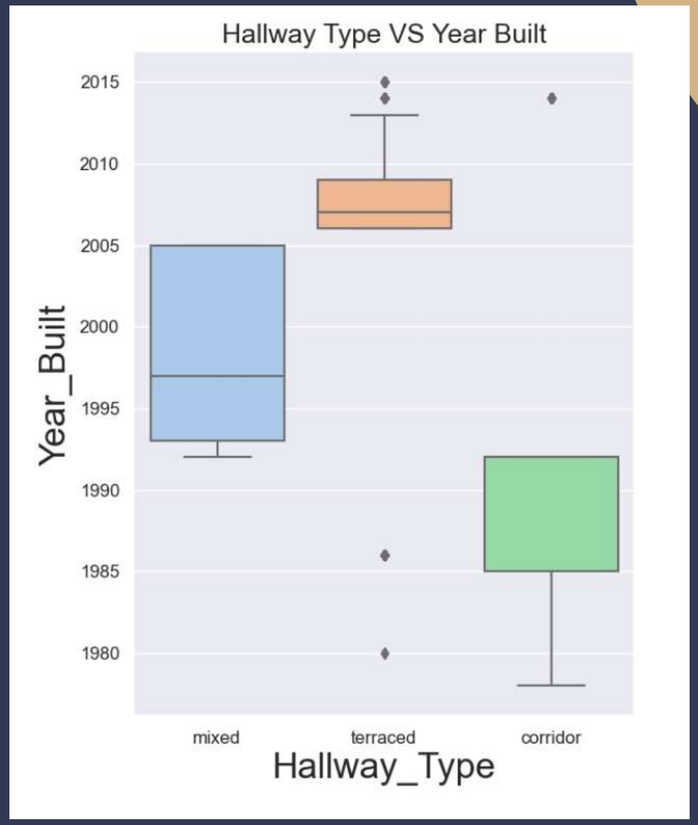
```
# # Kolom Time_To_Subway After

df['Time_To_Subway'].value_counts()
```

```
0-5min                1953
5min-10min             787
15min-20min            629
10min-15min            583
no_bus_stop_nearby     171
Name: Time_To_Subway, dtype: int64
```

# Hallway Type

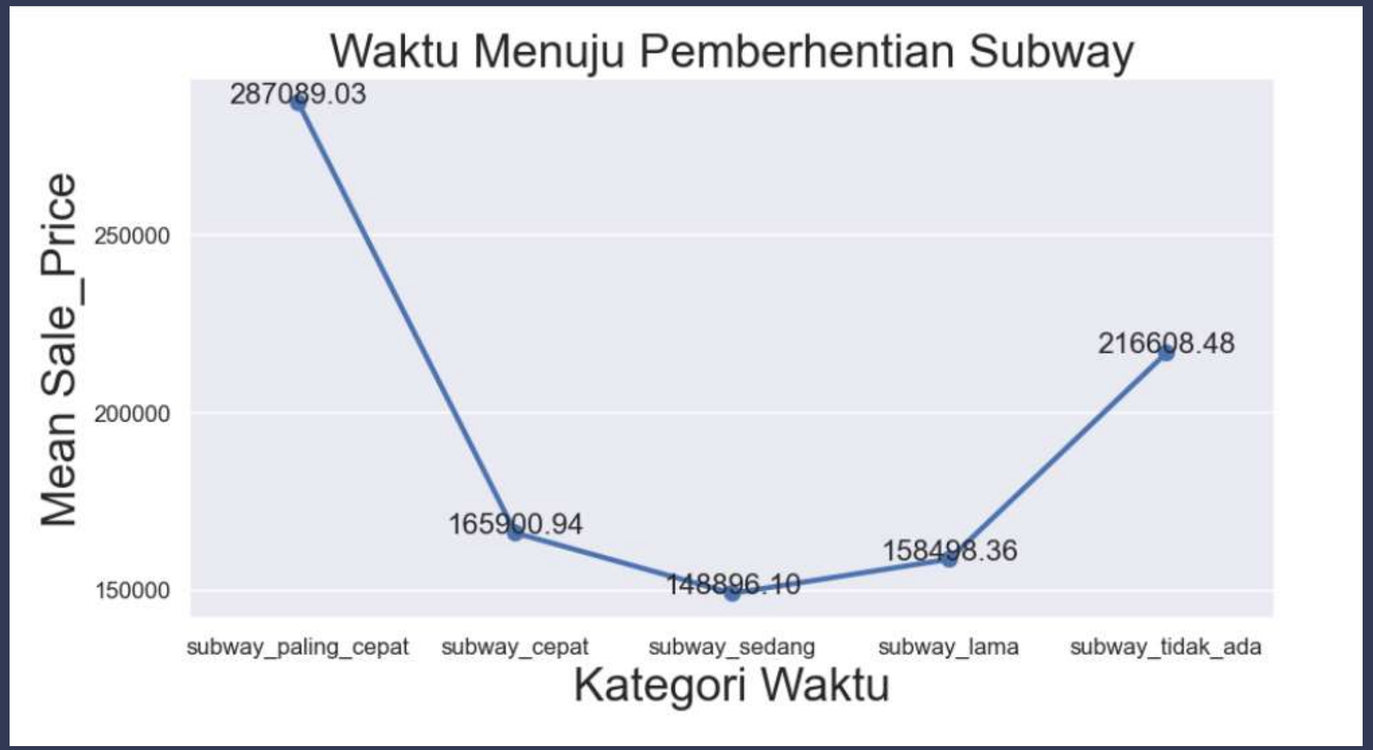



| Hallway_Type | corridor | mixed | terraced |
|---|---|---|---|
| Hallway_Type | 464 | 1131 | 2528 |



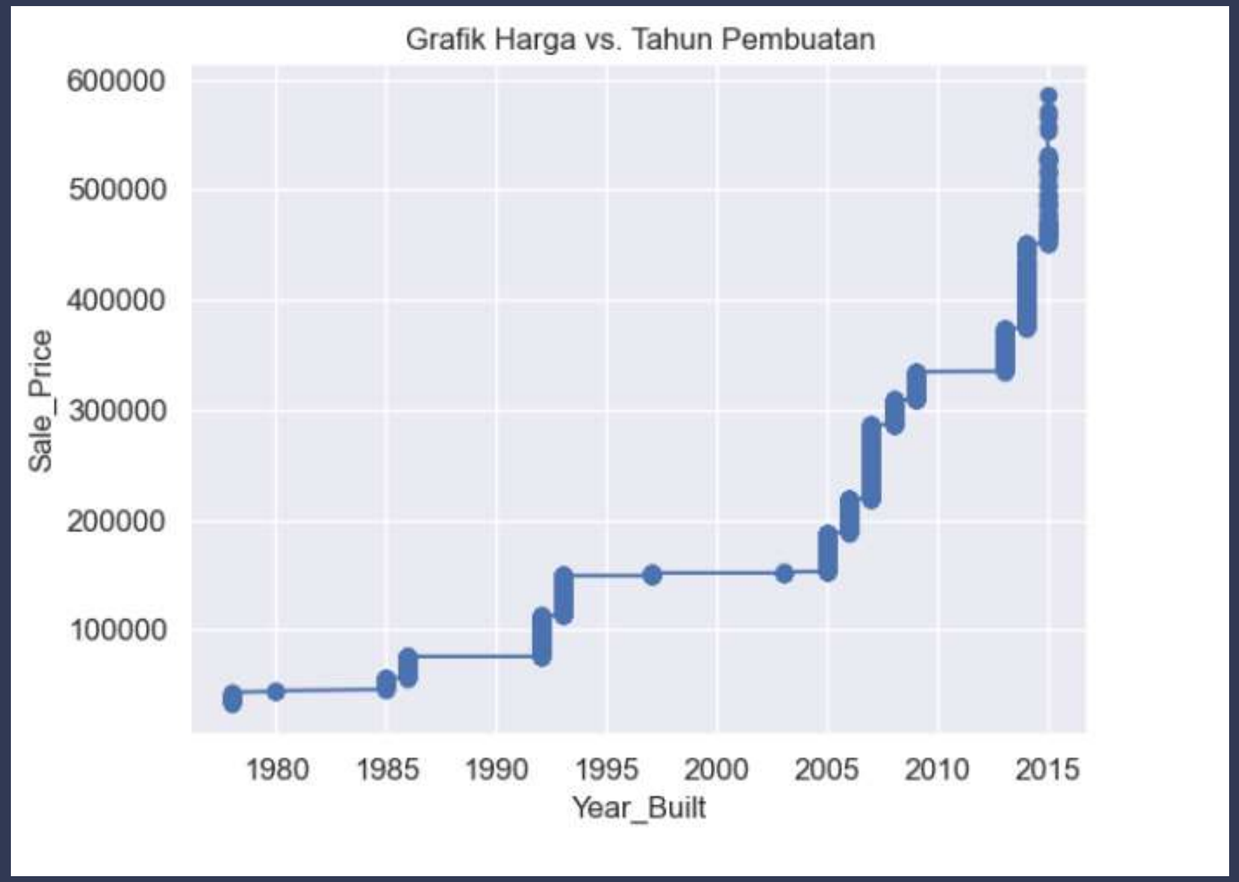

* Apartment bertipe Hallway Corridor banyak dibangun pada tahun 1995 kebawah.
* Apartment bertipe Hallway Mixed banyak dibangun pada tahun 1995 sampai dengan 2005.
* Apartment bertipe Hallway Terraced banyak dibangun pada tahun 2005 keatas.
* Untuk tahun terbaru pada 2005 keatas apartment tipe Hallway Mixed dan Corridor sudah jarang dibangun.

# Time To the Subway



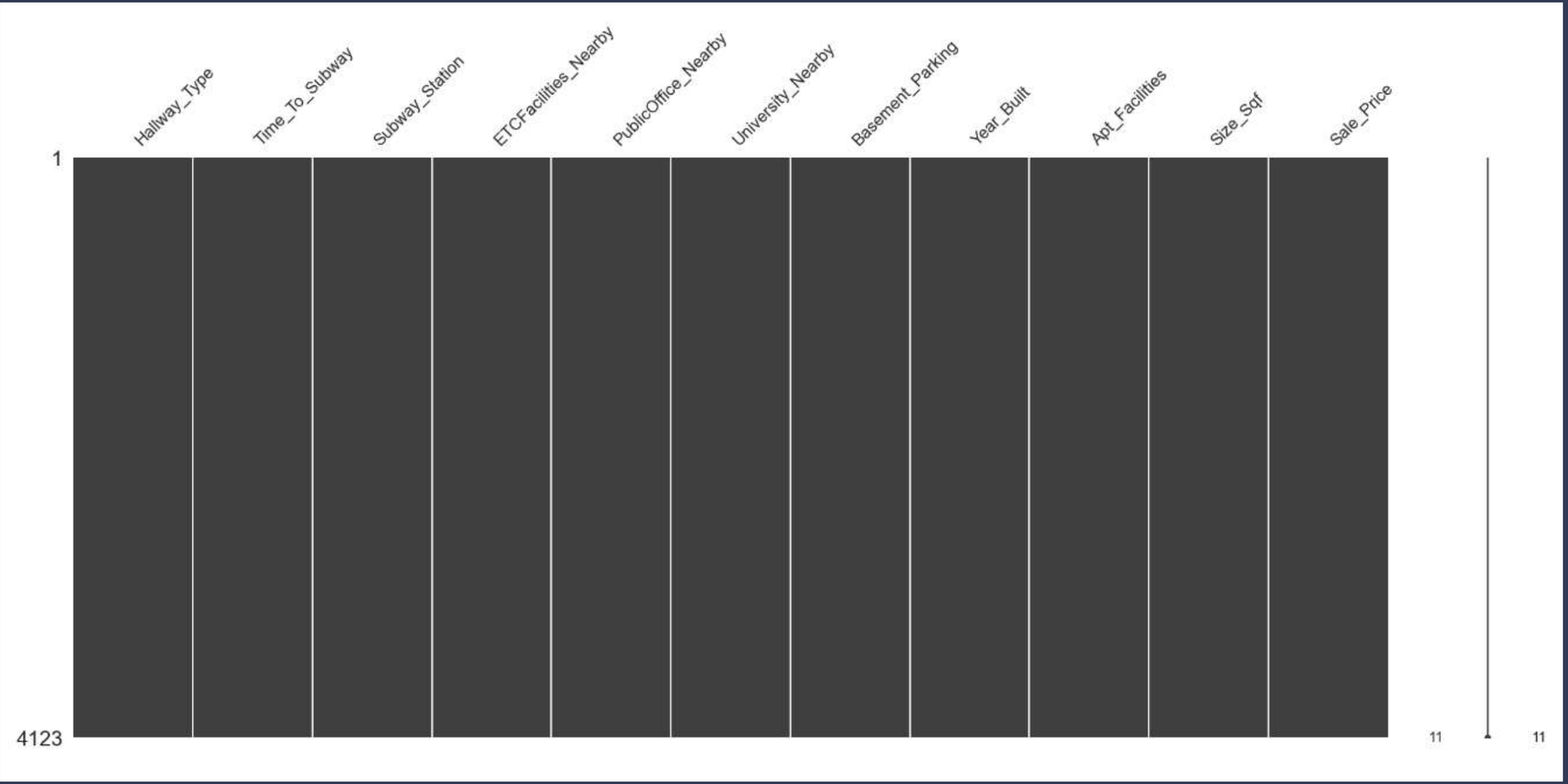

subway_paling_cepat = 0-5min
subway_cepat = 5min-10min
subway_sedang = 15min-20min
subway_lama = 10min-15min
subway_tidak_ada = no_bus_stop_nearby

# Sale_Price vs Year_Built



# DATA PREPOCESSING

# Check & Handling Missing Value




```
# Cheking if any missing value from the database

msno.matrix(df)
```

✓ 1.0s

# Check & Handling Duplicate Data

```
# check jumlah dari duplicated values
df.duplicated().sum()
```

✓ 0.0s

```
1422
```

```
# drop duplicated values
df.drop_duplicates(keep='last', inplace=True, ignore_index=True)
df
```

✓ 0.0s

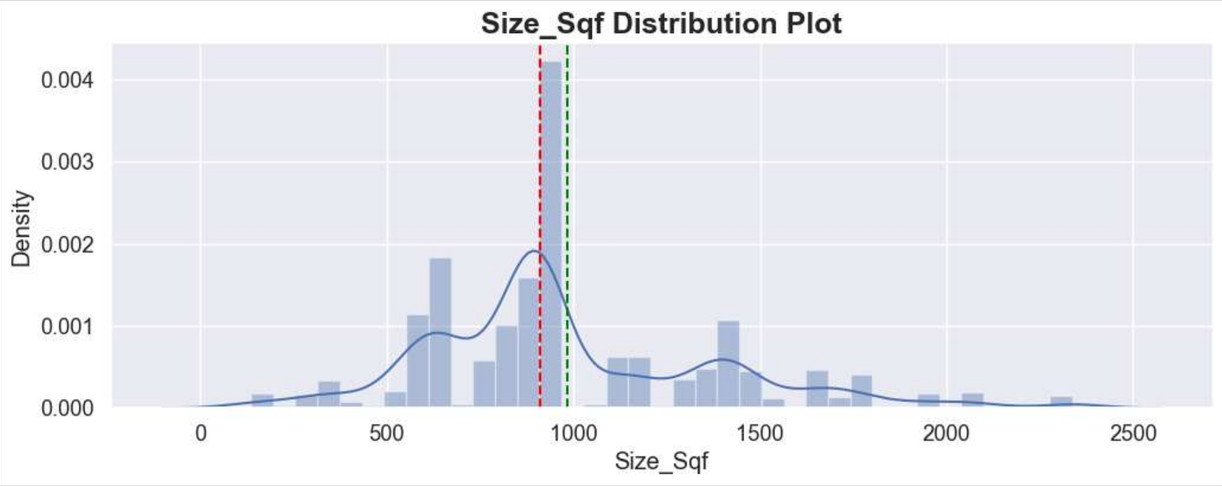
```
# Rechecking number of duplicated values
duplicateVal = df.duplicated().sum()
print (f'Setelah dilakukan proses delete duplicate, saat ini terdapat {duplicateVal} data terduplikasi.')
```

✓ 0.0s

```
Setelah dilakukan proses delete duplicate, saat ini terdapat 0 data terduplikasi.
```

# Check & Handling Rare Label

## Handling Outlier Pada Kolom Size_Sqf




```
Q1 quantile of the Size_Sqf : 743.0
Q2 quantile of the Size_Sqf : 910.0
Q3 quantile of the Size_Sqf : 1167.0
IQR of Size_Sqf : 424.0
Lower Outliers Limit of Size_Sqf : 107.0
Upper Outliers Limit of Size_Sqf : 1803.0
Amount of outliers Size_Sqf: 84
Percentage of outliers Size_Sqf: 3.11%
```

```
# Drop data outlier pada 'Sale_Price'
df = df[df['Size_Sqf'] <= upperbound_size]
```

✓ 0.0s

```
# Check the data of Income column that have value higher that upperbound

sizeOutliers = df.loc[df['Size_Sqf'] >= upperbound_size].sort_values(["Size_Sqf"],ascending=False)
sizeOutliers
```

[201] ✓ 0.0s

| Hallway_Type | Time_To_Subway | Subway_Station | ETCFacilities_Nearby | PublicOffice_Nearby | University_Nearby | Basement_Parking | Year_Built | Apt_Facilities | Size_Sqf | Sale_Price |
|---|---|---|---|---|---|---|---|---|---|---|

Sudah tidak terdapat nilai dari Size_Sqf yang mempunyai nilai Outliers.

# Check Correlations


Pada correlation dan signification matrix diatas menunjukkan bahwa semua fitur memiliki korelasi tinggi yaitu **Hallway_Type, Time_To_Subway, Subway_Station, ETCFacilities_Nearby, PublicOffice_Nearby, University_Nearby, Basement_Parking, Year_Built, Apt_Facilities, Size_Sqf, Sale_Price.** Dimana korelasinya adalah korelasi positif.

# FEATURE ENGINEERING

# Data Splitting

```
## Memisahkan Data Variabel Independen dari target

x = df_model.drop(['Sale_Price'], axis=1)
y = df_model['Sale_Price']

# Membagi training dan test data dengan proporsi 80:20
x_train, x_test, y_train, y_test = train_test_split(
    x,
    y,
    test_size=0.20,
    random_state=99)
```

✓ 0.0s

Membagi data latih dan data uji dengan proporsi 80:20 berarti membagi data menjadi dua bagian, 80% untuk data training dan 20% untuk data test.

# Data Encoding

```
# Convert fitur kategorikal ke numerikal

ordinal_mapping = [
    {'col':'Time_To_Subway',
     'mapping':{'no_bus_stop_nearby':0, '15min-20min':1, '10min-15min':2, '5min-10min':3, '0-5min':4 }}
    ]

ordinal_encoder = ce.OrdinalEncoder(cols=['Time_To_Subway'], mapping=ordinal_mapping)

transformer = ColumnTransformer([
                                ('one_hot_encoder', OneHotEncoder(drop='first'), ['Hallway_Type']),
                                ('binary_encoder', ce.BinaryEncoder(), ['Subway_Station']),
                                ('ordinal_encoder', ce.OrdinalEncoder(), ['Time_To_Subway'])
], remainder='passthrough')
```

✓ 0.0s

- Hallway_Type : Mengunakan encoding dengan tipe One-Hot Encoding karena terdapat 3 kategori
- Subway_Station : Mengunakan encoding dengan tipe Binary Encoding karena terdapat 8 kategori
- Time_To_Subway : Mengunakan encoding dengan tipe Ordinal Encoding karena bedasar kelompok waktunya akan diurutkan dari yang terendah (jauh dari subway) ke yang tertinggi (dekat dari subway).

# MODELING

# Choose Benchmark Model

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **KNN Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **XGBoost (Extreme Gradient Boosting) Regression**

| | Model | Mean_R2 | Std_R2 | Mean_Adjusted_R2 | Mean_RMSE | Std_RMSE | Mean_MAE | Std_MAE | Mean_MAPE | Std_MAPE | Validation_Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | KNN Regressor | 0.753858 | 0.020753 | 0.752438 | -49987.262203 | 2189.002246 | -38949.649886 | 1929.325288 | -0.200605 | 0.010254 | 2.922827 |
| 4 | XGBoost Regressor | 0.746293 | 0.013821 | 0.744829 | -50772.598081 | 1321.250570 | -39757.254136 | 1283.767272 | -0.201905 | 0.005973 | 10.001927 |
| 3 | Random Forest Regressor | 0.741941 | 0.013426 | 0.740452 | -51216.110596 | 1533.192856 | -39592.629240 | 1684.627072 | -0.202859 | 0.009076 | 45.100877 |
| 5 | Ridge | 0.714582 | 0.009831 | 0.712936 | -53885.932867 | 1712.318340 | -43358.877623 | 1629.375290 | -0.217295 | 0.004783 | 1.271828 |
| 0 | Linear Regression | 0.714565 | 0.009963 | 0.712918 | -53887.603773 | 1726.058418 | -43352.423005 | 1630.511347 | -0.217277 | 0.004853 | 2.762269 |
| 2 | DecisionTree Regressor | 0.614411 | 0.024882 | 0.612187 | -62599.464140 | 2356.466836 | -47165.799130 | 2089.197696 | -0.241714 | 0.009814 | 2.221051 |
| 6 | Lasso | -0.067290 | 0.028292 | -0.073447 | -104304.214084 | 5491.289682 | -83782.053160 | 4978.679825 | -0.505391 | 0.019450 | 1.348422 |

Berdasarkan rata-rata score **Mean_RMSE, Mean_MAE dan Mean_MAPE** kita akan mencoba melakukan tuning dan dilakukan prediksi pada test set pada dua model standalone terbaik, yaitu **KNN** dan **Ridge**. Namun jika improvement yang didapat masih kurang selanjutnya akan dilakukan tuning dan dilakukan prediksi pada test set model bebasis ensemble terbaik yaitu **XGBoost**

# Tuning Parameter Model Ridge

BEFORE TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| Ridge | 0.781122 | 0.775982 | 49515.062198 | 40086.287828 | 0.204492 |

AFTER TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| Ridge | 0.781155 | 0.776016 | 49511.32421 | 40081.900241 | 0.204412 |

# Tuning Parameter Model KNN

BEFORE TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| KNN | 0.776511 | 0.771262 | 50033.925425 | 39457.543893 | 0.198427 |

AFTER TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| KNN | 0.781064 | 0.775923 | 49521.564052 | 39112.346613 | 0.197193 |

# Tuning Parameter Model XGBoost

BEFORE TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| XGBoost | 0.771865 | 0.766507 | 50551.302893 | 39335.443918 | 0.20198 |

AFTER TUNING:

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| XGBoost | 0.818233 | 0.813964 | 45122.573696 | 36019.151501 | 0.19069 |

# Model Comparison After Tuning

```
display(score_after_tuning_ridge, score_after_tuning_knn, score_after_tuning_xgboost)
```

✓ 0.0s

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| Ridge | 0.781155 | 0.776016 | 49511.32421 | 40081.900241 | 0.204412 |

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| KNN | 0.781064 | 0.775923 | 49521.564052 | 39112.346613 | 0.197193 |

| | R-squared | Adjusted R-squared | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| XGBoost | 0.818233 | 0.813964 | 45122.573696 | 36019.151501 | 0.19069 |

# Choose Benchmark Model

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **KNN Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **XGBoost (Extreme Gradient Boosting) Regression**

| | Model | Mean_R2 | Std_R2 | Mean_Adjusted_R2 | Mean_RMSE | Std_RMSE | Mean_MAE | Std_MAE | Mean_MAPE | Std_MAPE | Validation_Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | KNN Regressor | 0.753858 | 0.020753 | 0.752438 | -49987.262203 | 2189.002246 | -38949.649886 | 1929.325288 | -0.200605 | 0.010254 | 2.922827 |
| 4 | XGBoost Regressor | 0.746293 | 0.013821 | 0.744829 | -50772.598081 | 1321.250570 | -39757.254136 | 1283.767272 | -0.201905 | 0.005973 | 10.001927 |
| 3 | Random Forest Regressor | 0.741941 | 0.013426 | 0.740452 | -51216.110596 | 1533.192856 | -39592.629240 | 1684.627072 | -0.202859 | 0.009076 | 45.100877 |
| 5 | Ridge | 0.714582 | 0.009831 | 0.712936 | -53885.932867 | 1712.318340 | -43358.877623 | 1629.375290 | -0.217295 | 0.004783 | 1.271828 |
| 0 | Linear Regression | 0.714565 | 0.009963 | 0.712918 | -53887.603773 | 1726.058418 | -43352.423005 | 1630.511347 | -0.217277 | 0.004853 | 2.762269 |
| 2 | DecisionTree Regressor | 0.614411 | 0.024882 | 0.612187 | -62599.464140 | 2356.466836 | -47165.799130 | 2089.197696 | -0.241714 | 0.009814 | 2.221051 |
| 6 | Lasso | -0.067290 | 0.028292 | -0.073447 | -104304.214084 | 5491.289682 | -83782.053160 | 4978.679825 | -0.505391 | 0.019450 | 1.348422 |

Berdasarkan rata-rata score **Mean_RMSE, Mean_MAE dan Mean_MAPE** kita akan mencoba melakukan tuning dan dilakukan prediksi pada test set pada dua model standalone terbaik, yaitu **KNN** dan **Ridge**. Namun jika improvement yang didapat masih kurang selanjutnya akan dilakukan tuning dan dilakukan prediksi pada test set model bebasis ensemble terbaik yaitu **XGBoost**

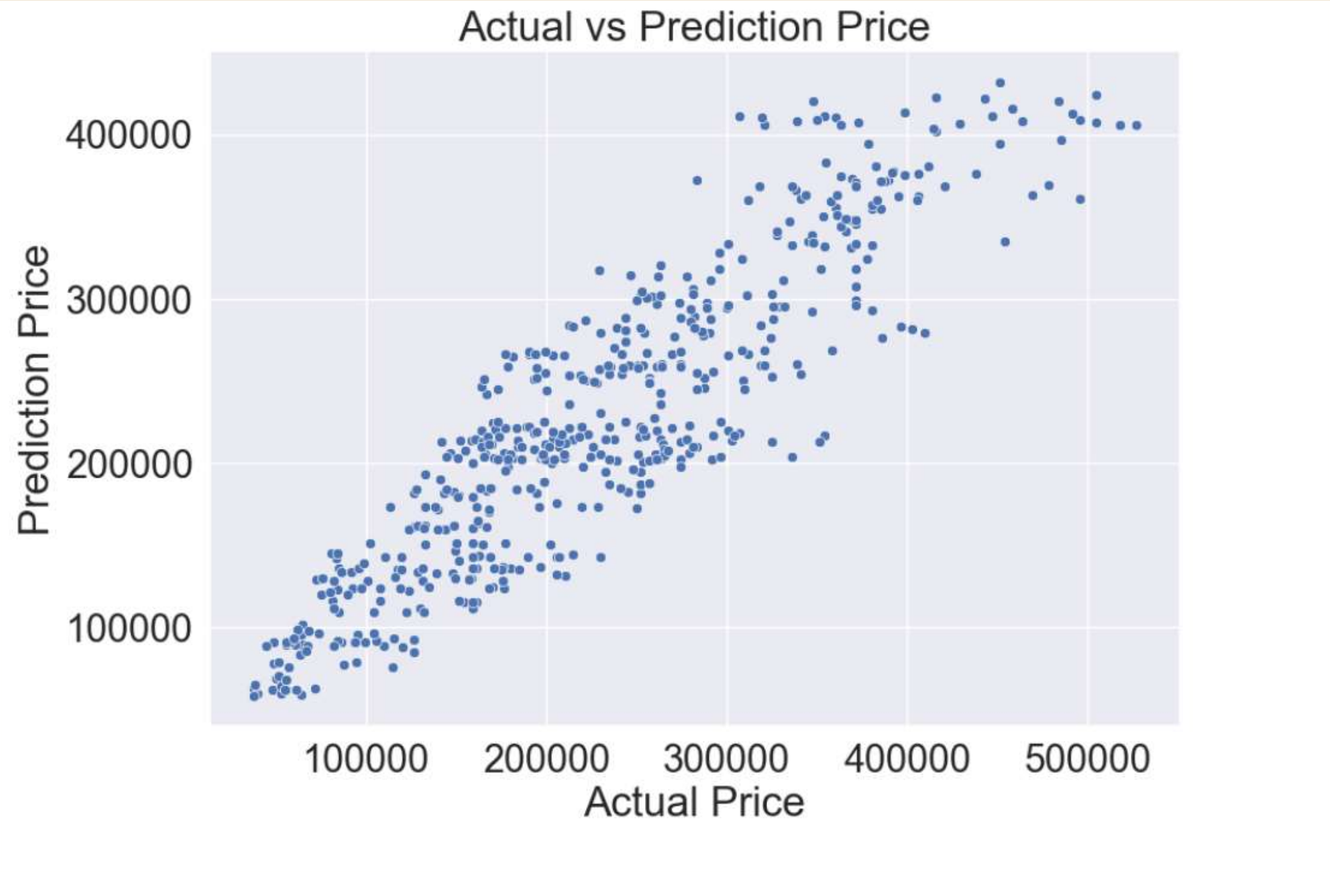
# Choose Benchmark Model

- **Linear Regression**
- **Lasso Regression**
- **Ridge Regression**
- **KNN Regression**
- **Decision Tree Regression**
- **Random Forest Regression**
- **XGBoost (Extreme Gradient Boosting) Regression**

| | Model | Mean_R2 | Std_R2 | Mean_Adjusted_R2 | Mean_RMSE | Std_RMSE | Mean_MAE | Std_MAE | Mean_MAPE | Std_MAPE | Validation_Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | KNN Regressor | 0.753858 | 0.020753 | 0.752438 | -49987.262203 | 2189.002246 | -38949.649886 | 1929.325288 | -0.200605 | 0.010254 | 2.922827 |
| 4 | XGBoost Regressor | 0.746293 | 0.013821 | 0.744829 | -50772.598081 | 1321.250570 | -39757.254136 | 1283.767272 | -0.201905 | 0.005973 | 10.001927 |
| 3 | Random Forest Regressor | 0.741941 | 0.013426 | 0.740452 | -51216.110596 | 1533.192856 | -39592.629240 | 1684.627072 | -0.202859 | 0.009076 | 45.100877 |
| 5 | Ridge | 0.714582 | 0.009831 | 0.712936 | -53885.932867 | 1712.318340 | -43358.877623 | 1629.375290 | -0.217295 | 0.004783 | 1.271828 |
| 0 | Linear Regression | 0.714565 | 0.009963 | 0.712918 | -53887.603773 | 1726.058418 | -43352.423005 | 1630.511347 | -0.217277 | 0.004853 | 2.762269 |
| 2 | DecisionTree Regressor | 0.614411 | 0.024882 | 0.612187 | -62599.464140 | 2356.466836 | -47165.799130 | 2089.197696 | -0.241714 | 0.009814 | 2.221051 |
| 6 | Lasso | -0.067290 | 0.028292 | -0.073447 | -104304.214084 | 5491.289682 | -83782.053160 | 4978.679825 | -0.505391 | 0.019450 | 1.348422 |

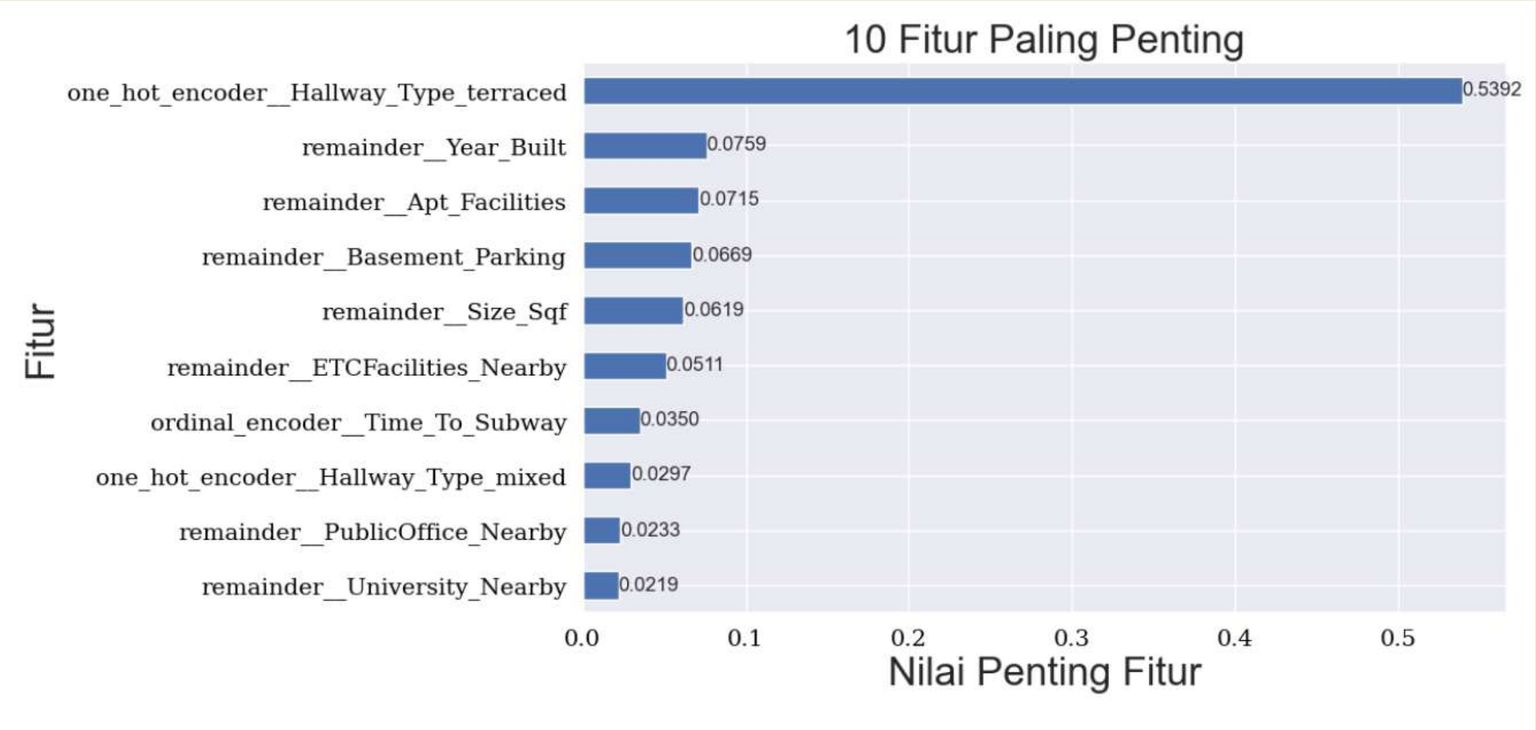
Berdasarkan rata-rata score **Mean_RMSE, Mean_MAE dan Mean_MAPE** kita akan mencoba melakukan tuning dan dilakukan prediksi pada test set pada dua model standalone terbaik, yaitu **KNN** dan **Ridge**. Namun jika improvement yang didapat masih kurang selanjutnya akan dilakukan tuning dan dilakukan prediksi pada test set model bebasis ensemble terbaik yaitu **XGBoost**

# Actual VS Predition




Terdapatnya nilai-nilai error yang membuat perbedaan yang cukup signifikan antara nilau RMSE dan MAE. Hal ini dapat terlihat pada grafik scatter plot diatas, dimana terdapat harga aktual yang tinggi akan tetapi di prediksikan lebih rendah (underestimation) ataupun sebaliknya (overestimation). Namun, jika dilihat dari nilai MAPE yaitu 19.069% menjadikan model masuk kedalam kategori "good forecasting"

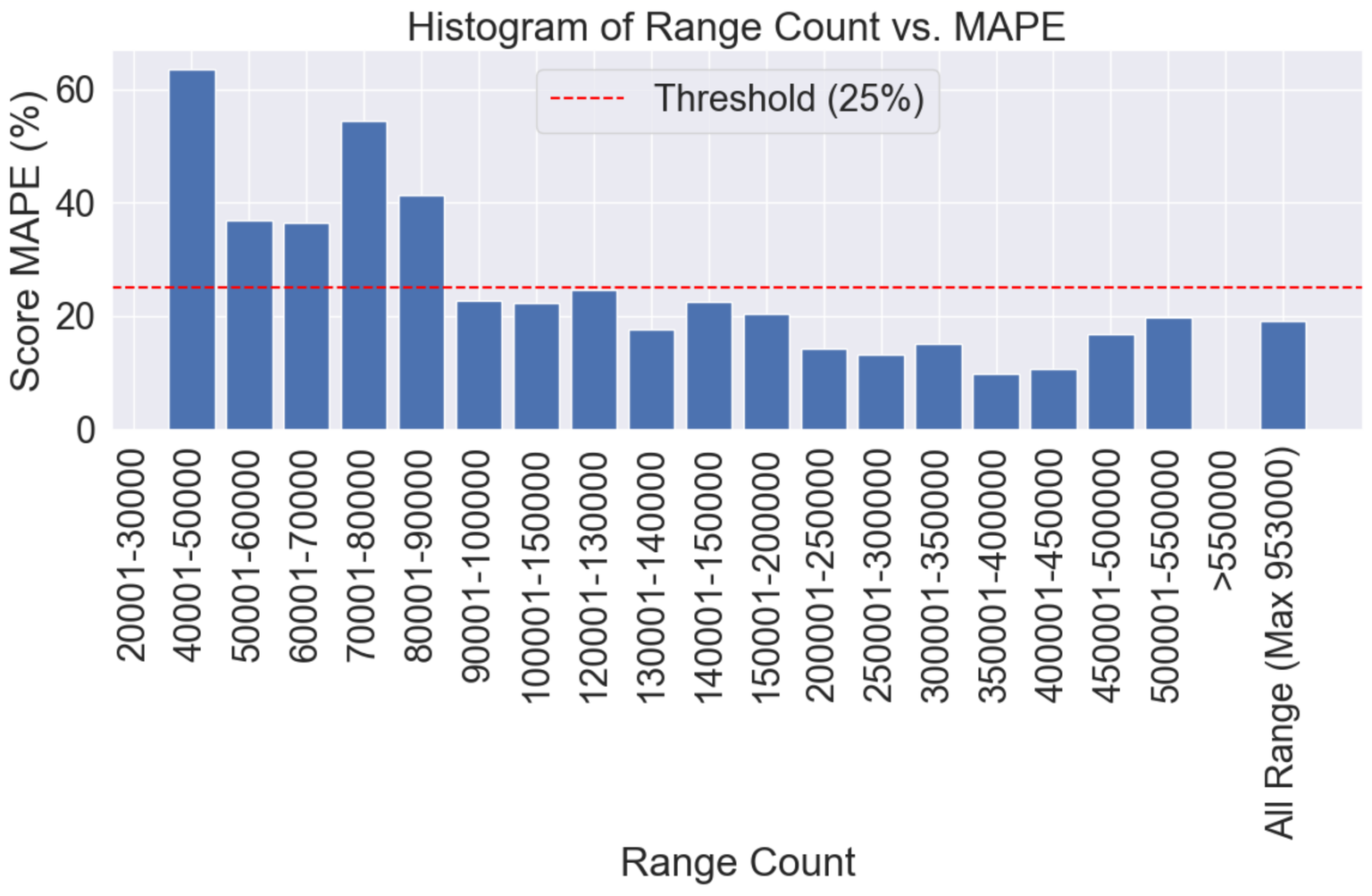
# Model Explanation




Berdasarkan model final (model XGBoost), feature yang paling berpengaruh terhadap harga sewa apartment

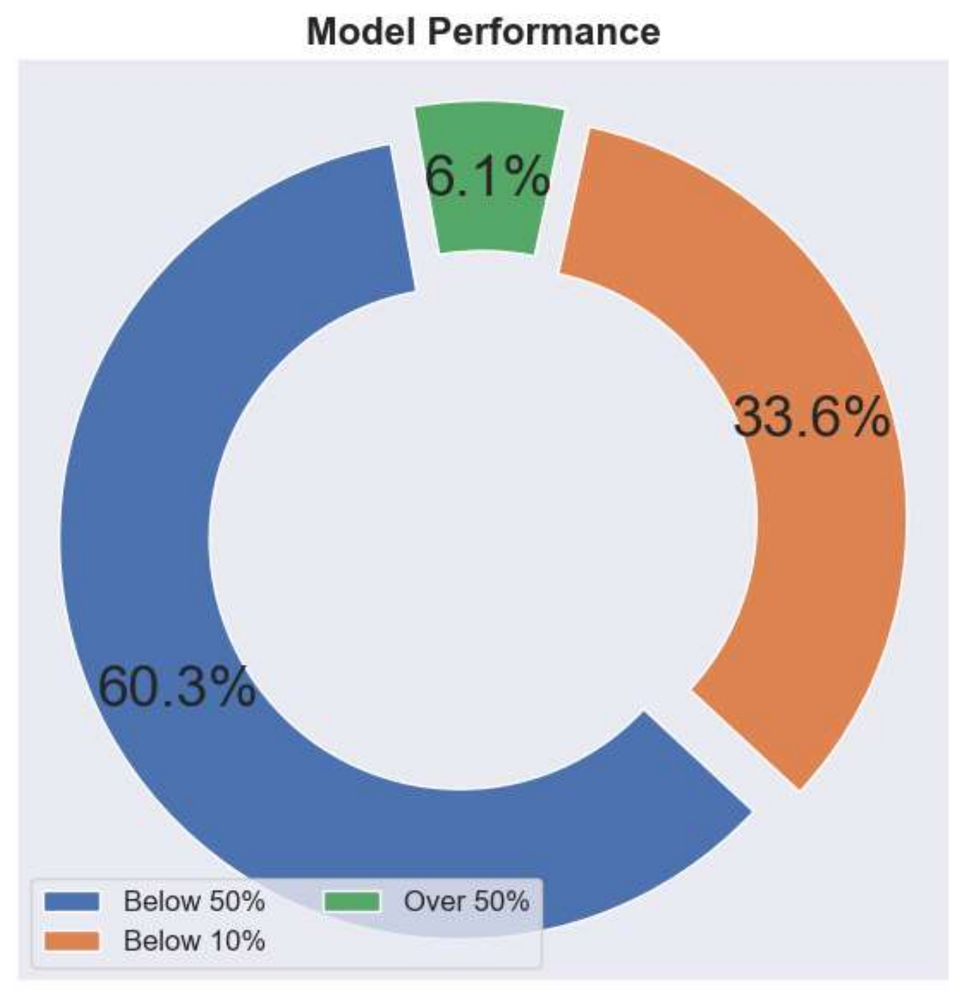
1. Hallway_Type (0.5392)
2. Year_Built (tahun pembuatan) (0.0759)
3. Apt_Facilities (Fasilitas Apartment) (0.0715)

# Model Limitation




Dapat dijelaskan limitasi untuk model berikut yaitu model XGboost pada dataset data_daegu_apartment dapat memprediksi harga dengan range mulai dari **90001 won dan seterusnya sampai dengan diatas 500000**. Dimana pada rentang harga ini nilai MAPE bernilai dibawah 25 %

# Error Rate Prediction




1. **Prediksi dengan tingkat kesalahan di atas 50% (buruk):** Terlihat bahwa hanya sekitar 6.1% dari prediksi yang termasuk dalam kategori ini. Persentase kesalahan yang relatif rendah ini dapat dimengerti, mengingat bahwa tidak ada model yang benar-benar sempurna.

2. **Prediksi dengan tingkat kesalahan di bawah 50% (baik):** Sekitar 60.3% dari total prediksi masuk ke dalam kategori ini, menunjukkan kinerja yang memadai. Ini menunjukkan bahwa model ini mampu memberikan hasil yang layak, terutama dalam mengantisipasi skenario dengan risiko yang rendah.

3. **Prediksi dengan tingkat kesalahan di bawah 10% (sangat baik):** Sebanyak 33,6% dari hasil prediksi termasuk dalam kategori ini. Ini menunjukkan kemampuan model untuk memberikan hasil yang sangat dekat dengan kenyataan, dan potensinya untuk memberikan prediksi yang sangat akurat.

# CONCLUSION & RECOMENDATION

# Conclusion

1. Model terbaik yang didapatkan pada proses modeling untuk memprediksi harga sewa apartment adalah model **XGBoost**
2. Setelah proses Hyperparameter Tuning dan Predict Test Set didapatkan bahwa model **XGBoost** mempunyai performa yang lebih baik pada matriks R-Squared, Adjusted R-Squared, MRSE, MAE dan MAPE dibandingkan pada model **Ridge** dan model **KNN**
3. Feature yang paling berpengaruh terhadap harga apartment di Daegu adalah tipe apartment terraced, Year_Built (tahun pembuatan), dan  Apt_Facilities (Fasilitas Apartment).
4. Jika melihat nilai MAPE yang dihasilkan oleh model setelah dilakukan tuning hyperparameter, yaitu sebesar 19.069%, maka estimasi rata-rata akan menyimpang kurang lebih sebesar 19.069% dari harga sebenarnya.
5. Model berikut yaitu model XGboost pada dataset data_daegu_apartment dapat memprediksi harga dengan range mulai dari 90001 won dan seterusnya sampai dengan diatas 500000. Dimana pada rentang harga ini nilai MAPE bernilai dibawah 25 %.
6. Akan tetap ada kemungkinan harga akan meleset dari perkiraan karena adanya tingkat bias yang masih cukup tinggi antara harga aktual dan prediksi. Untuk mengatasi hal ini, kita dapat memperluas jumlah fitur yang akan lebih mendalam dalam menjelaskan faktor-faktor yang mempengaruhi harga.

# Recommendation

1. Mengambil pertimbangan untuk menambah feature yang memiliki korelasi yang lebih signifikan terhadap harga apartemen, seperti jumlah personel keamanan yang ditempatkan di apartment, informasi inklusivitas furniture, lantai berapa tempat apartemen berada, tahun penjualan unit apartemen, jumlah ruangan yang terdapat di dalam unit (termasuk jumlah kamar tidur, kamar mandi, dan dapur) dan variabel-variabel lain yang memiliki dampak langsung terhadap harga.
2. Melakukan ekspansi dataset dengan mengumpulkan data yang lebih mutakhir dan informatif terkait harga apartemen di Daegu, Korea Selatan. diharapkan model regresi dapat memahami lebih baik pola-pola dalam data, yang pada akhirnya dapat menghasilkan peningkatan akurasi prediksi.
3. Mengaplikasikan model machine learning yang lebih canggih guna membangun prediksi harga apartemen. Meskipun model yang lebih kompleks dapat memberikan hasil yang lebih akurat, perlu diperhatikan bahwa kompleksitasnya juga dapat menghambat pemahaman terhadap model yang dihasilkan.
4. Memperhitungkan variabel eksternal yang secara keseluruhan memiliki potensi untuk memengaruhi harga jual apartemen di Daegu, Korea Selatan, termasuk namun tidak terbatas pada faktor-faktor seperti ketersediaan kredit dalam sewa apartment, tingkat suku bunga, Indeks Harga Konsumen (IHK), performa Produk Domestik Bruto (PDB) setiap tahunnya, dan faktor lain yang relevan.

# Contact details

If you have any questions, feel free to contact me!

**MOBILE PHONE**

+62 877 39 300 900

**EMAIL ADDRESS**

alfianghaffar01@gmail.com