Predicting The Price of A House Using Machine Learning

Data Profiling:

Before creating a machine learning model, I need to understand the dataset first. So I'll do data profiling to get a better data understanding.

Importing Data

```
#Importing Data
import pandas as pd
pd.set_option('display.max_columns',None)
df = pd.read_csv('Data Rumah.csv')
```

The first step is to initialize the library, import the dataset into Python using Pandas, and assign it as df. The data downloaded from Kaggle will be saved as 'Data Rumah.csv'.

Showing The Length of The Data

```
#Showing The Length of The Data
print("\nThe Length of The Data: ", len(df))
```

Showing The Information of The Data

```
#Showing The Information of The Data
print("\nThe Information of The Data: ")
print(df.info())
```

```
The Information of The Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
```

The fourth step is to get information from the data using the .info()

Showing The Statistical Calculations

```
#Showing The Statistical Calculations
print("\nThe Statistical Calculations: ")
print(df.describe().T)
```

The fifth step is to display a statistical analysis of the data using .describe().

Showing The Unique Data

```
#Showing The Unique Data
print("\nThe Unique Data: ")
print(df.nunique())
```

```
The Unique Data:
date              70
price           1741
bedrooms          10
bathrooms         26
sqft_living      566
sqft_lot        3113
floors             6
waterfront         2
view               5
condition          5
sqft_above       511
sqft_basement    207
yr_built         115
yr_renovated      60
street          4525
city              44
statezip          77
country            1
dtype: int64
```

Changing The Column’s Name And The Column’s Value

#Changing The Column's Name And The Column's Value

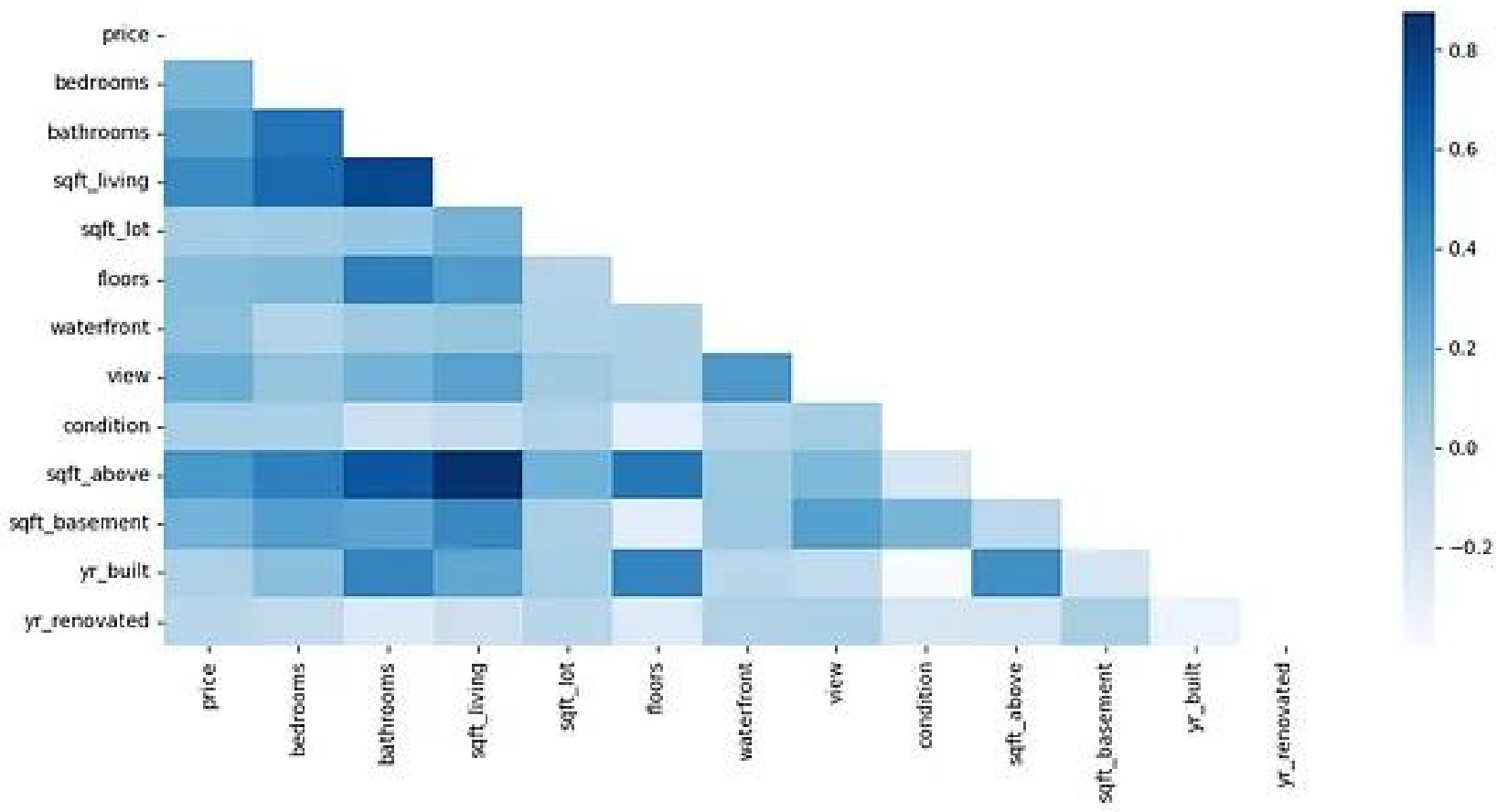
```
kolom_string = list(df.dtypes[df.dtypes == 'object'].index)
for col in kolom_string:
    df[col] = df[col].str.lower().str.replace(' ', '_')

df.columns = df.columns.str.capitalize().str.replace('_', ' ')

df.rename(columns ={'Bedrooms': 'Bedroom(s)',
                    'Bathrooms': 'Bathroom(s)',
                    'Sqft living': 'Square Meter Living',
                    'Sqft lot': 'Square Meter Lot',
                    'Floors': 'Floor(s)',
                    'Sqft above': 'Square Meter Above',
                    'Sqft basement': 'Basement',
                    'Yr built': 'Year Built',
                    'Yr renovated': 'Renovated',
                    'Statezip': 'State ZIP'}, inplace=True)
```

Looking For A Correlation

```
#Looking For A Correlation
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(17, 15))
corr_mask = np.triu(df.corr())
h_map = sns.heatmap(df.corr(), mask=corr_mask,
cmap='Blues')
plt.yticks(rotation=360)
plt.show()
```



Separating Features and Labels

```
#Separating Features and
Labels
X = df.drop('Price', axis=1)
y = df['Price'].astype(int)
```

Evaluating The Machine Learning Model

The next stage after building a machine learning model is evaluating that model. In this evaluation process, I will look at the Mean Squared Error, Mean Absolute Error, and Root Mean Squared Error values. I will take the smallest RMSE value to decide which one is better.

Linear Regression

The result of using the Linear Regression is 13978053502 MSE, 81551 MAE, and 118229 RMSE.

Decision Tree Regressor

The result of using the Decision Tree Regressor is 30269819377 MSE, 117398 MAE, and 173982 RMSE.

Random Forest Regressor

The result of using the Random Forest Regressor is 15404909596 MSE, 84112 MAE, and 124117 RMSE.

Lasso

The result of using the Lasso is 13967964699 MSE, 81518 MAE, and 118186 RMSE.

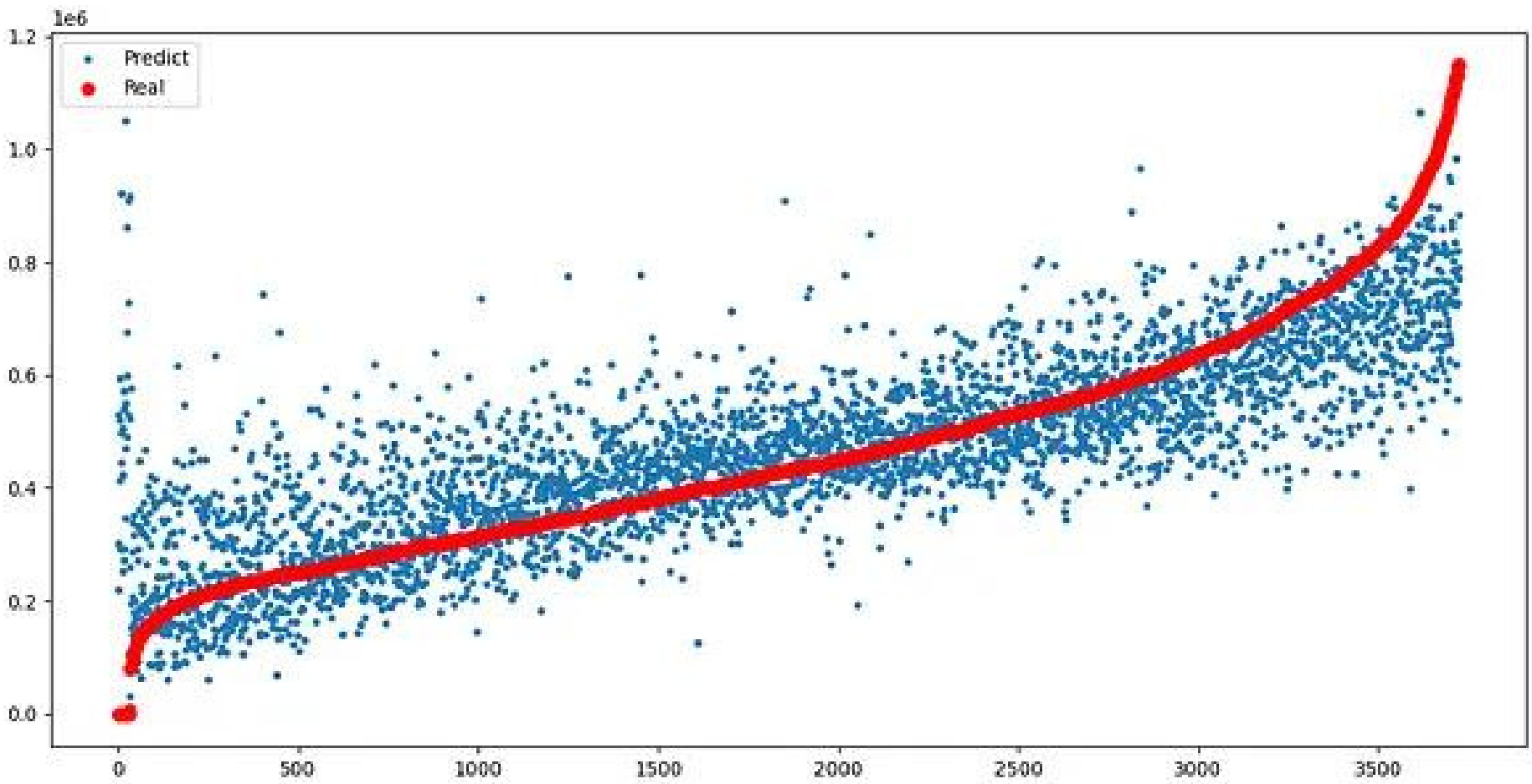
Ridge

The result of using the Ridge is 13213872570 MSE, 80624 MAE, and 114952 RMSE.

From this evaluation, the Machine Learning model using Ridge has the smallest RMSE.

Visualize The Machine Learning Model

```
#Visualize The Machine Learning Model
fig = plt.figure(figsize=(17, 10))
df = df.sort_values(by=['Price'])
X = df.drop('Price', axis=1)
y = df['Price']
plt.scatter(range(X.shape[0]),
model_ridge.predict(X), marker='.', label='Predict')
plt.scatter(range(X.shape[0]), y, color='red',
label='Real')
plt.legend(loc='best', prop={'size': 10})
plt.show()
```



Validating The Machine Learning Model

```
#Validating The Machine Learning Model
for i in range(5):
    real = y_val.iloc[i]
    pred =
int(model_ridge.predict(X_val.iloc[i].to_frame().T)
[0])
    print(f'Real Value       ----->>>>> {real} $\n'
          f'Predicted Value ----->>>>> {pred} $')
```

Actual Price	Predicted Price
646000	598828
255000	446798
530000	521669
385000	383312
549900	647619

Removing Duplicated Data

```
#Removing Duplicated Data
print("\nRemoving Duplicated Data")
df.drop_duplicates(inplace=True)
print('\nThe Shape of The Data After Removing The Duplicated Data: ', df.shape)
```