

MODEL PREDIKSI HARGA RUMAH

Gilang Agung Saputra

Artificial Intelligence merupakan bidang ilmu yang mempelajari dan membuat mesin yang dapat berpikir layaknya manusia, *Artificial Intelligence* dapat diterapkan dalam berbagai hal seperti pengelompokan benda, rekomendasi lagu, film, memprediksi terjadinya kebakaran, memprediksi harga saham dan masih banyak lagi.

Prediksi harga rumah merupakan hal yang dapat dilakukan oleh *Artificial Intelligence*, dengan menggunakan data spesifikasi dan harga rumah serta menggunakan algoritma yang ada pada *Artificial Intelligence* kita dapat membuat mesin yang mampu memprediksi harga rumah.

Percobaan kali ini kita akan menggunakan data harga rumah yang diperoleh dari “https://github.com/adimasmudi/house_prediction”, membandingkan beberapa algoritma kemudian menggunakan algoritma dengan nilai akurasi tertinggi. Berikut adalah langkah-langkah untuk membuat mesin prediksi harga rumah:

1. *Import* data yang telah diunduh

Import data yang ingin digunakan dengan *library* pandas

```
import pandas as pd
```

File yang digunakan memiliki format “csv”

```
import file yang berada di komputer (local)
```

```
dataset = pd.read_csv('kc_house_data.csv')
```

2. *Data Understanding*

Tahap ini merupakan proses untuk mengetahui definisi dari nama variabel yang ada pada data, memahami data dan variabel yang perlu digunakan, memeriksa data yang kita gunakan apakah memiliki kualitas yang baik dan sudah sesuai dengan mesin yang ingin kita buat. *Import library* yang kita butuhkan untuk melakukan proses *Data Understanding* dan proses-proses selanjutnya.

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

Variabel yang ingin akan digunakan adalah 'floors', 'bedrooms', 'sqft_living', 'grade', 'yr_built', dan 'price'.

```
# Memuat data dari komputer (local)
dataFrame = pd.read_csv('kc_house_data.csv', usecols=['floors','bedrooms', 'bathrooms', 'sqft_living', 'grade','yr_built','price'])
dataFrame
```

✓ 0.1s

	price	bedrooms	bathrooms	sqft_living	floors	grade	yr_built
0	221900.0	3	1.00	1180	1.0	7	1955
1	538000.0	3	2.25	2570	2.0	7	1951
2	180000.0	2	1.00	770	1.0	6	1933
3	604000.0	4	3.00	1960	1.0	7	1965
4	510000.0	3	2.00	1680	1.0	8	1987
...
21608	360000.0	3	2.50	1530	3.0	8	2009
21609	400000.0	4	2.50	2310	2.0	8	2014
21610	402101.0	2	0.75	1020	2.0	7	2009
21611	400000.0	3	2.50	1600	2.0	8	2004
21612	325000.0	2	0.75	1020	2.0	7	2008

Penjelasan dari variabel:

- bedrooms : jumlah kamar tidur
- bathrooms : jumlah kamar mandi
- sqft_living : luas ruangan dalam satuan sqft (square feet)
- floors : jumlah lantai
- grade : klasifikasi dari kelas rumah
- yr_built : tahun bangunan
- price : harga rumah (\$)

Melihat tipe data dari masing-masing variabel

```
#melihat nama kolom, jumlah data, dan tipe data
dataFrame.info()
```

✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   price       21613 non-null  float64
1   bedrooms    21613 non-null  int64
2   bathrooms   21613 non-null  float64
3   sqft_living 21613 non-null  int64
4   floors      21613 non-null  float64
5   grade       21613 non-null  int64
6   yr_built    21613 non-null  int64
```

Ada tipe data yang kurang sesuai pada variabel 'bathrooms' dan 'floors', kita harus merubah menjadi *integer* agar menjadi bilangan bulat

3. Data Preparation

Data Preparation adalah proses pengumpulan, penggabungan, penataan, dan pengorganisasian data sehingga dapat digunakan dalam analitik, dan visualisasi data.

Melihat apakah ada data yang kosong pada masing-masing variabel

```
#melihat data yang kosong
dataFrame.isnull().sum()
✓ 0.9s
```

price	0
bedrooms	0
bathrooms	0
sqft_living	0
floors	0
grade	0
yr_built	0

Dari hasil di atas tidak ada data yang kosong, sehingga tidak perlu manipulasi data kosong

Mengubah tipe data *float* dari variabel 'bathrooms' dan 'floors' menjadi *integer*

```
#mengubah tipe data kolom 'bathrooms' menjadi integer(int)
dataFrame['bathrooms'] = dataFrame['bathrooms'].astype(int)

#mengubah tipe data kolom 'floors' menjadi integer(int)
dataFrame['floors'] = dataFrame['floors'].astype(int)
dataFrame.dtypes
```

price	float64
bedrooms	int64
bathrooms	int32
sqft_living	int64
floors	int32
grade	int64
yr_built	int64

Variabel 'bathrooms' dan 'floors' berhasil diubah menjadi *integer*

Menghapus variabel 'bedrooms' dengan nilai 33

```
#menghapus data dengan 'bedrooms' dengan nilai 33
dataFrame = dataFrame[dataFrame['bedrooms'] != 33]

#cek apakah kolom 'bedrooms' dengan nilai 33 sudah dihilangkan
dataFrame[dataFrame['bedrooms'] == 33]

0.6s

price  bedrooms  bathrooms  sqft_living  floors  grade  yr_built
```

Variabel 'bedrooms' dengan nilai 33 berhasil dihapus

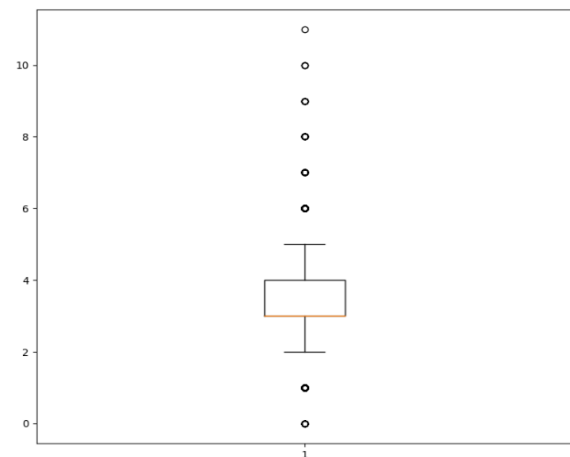
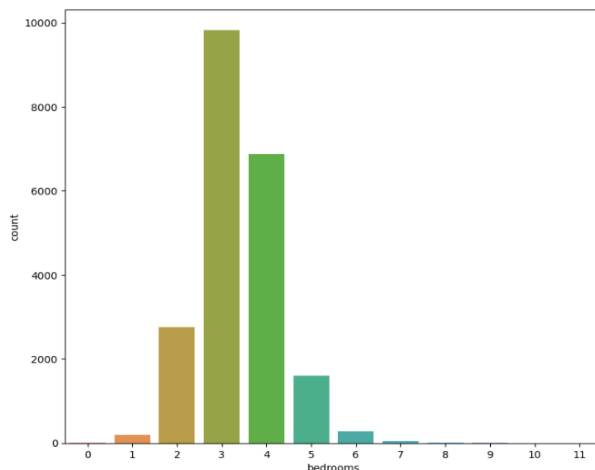
Proses selanjutnya adalah *Exploratory Data Analysis* (EDA), EDA merupakan proses untuk melihat pola-pola tertentu dalam data, EDA dapat dibagi menjadi *Univariate Analysis* yang menganalisa masing-masing variabel tanpa melihat hubungan dengan variabel lain dan *Bivariate Analysis* yang melihat hubungan satu variabel dengan variabel lain, dalam kasus ini *Bivariate Analysis* yang kita lakukan adalah melihat korelasi masing-masing variabel terhadap harga.

Univariate Analysis variabel 'bedrooms'

```
# univariate analysis variabel 'bedrooms'
f = plt.figure(figsize=(20,8))

# add_subplot(baris, kolom, posisi)
#plot kiri
f.add_subplot(1,2,1)
sns.countplot(dataFrame['bedrooms'])

#plot kanan
f.add_subplot(1,2,2)
plt.boxplot(dataFrame['bedrooms'])
plt.show()
```



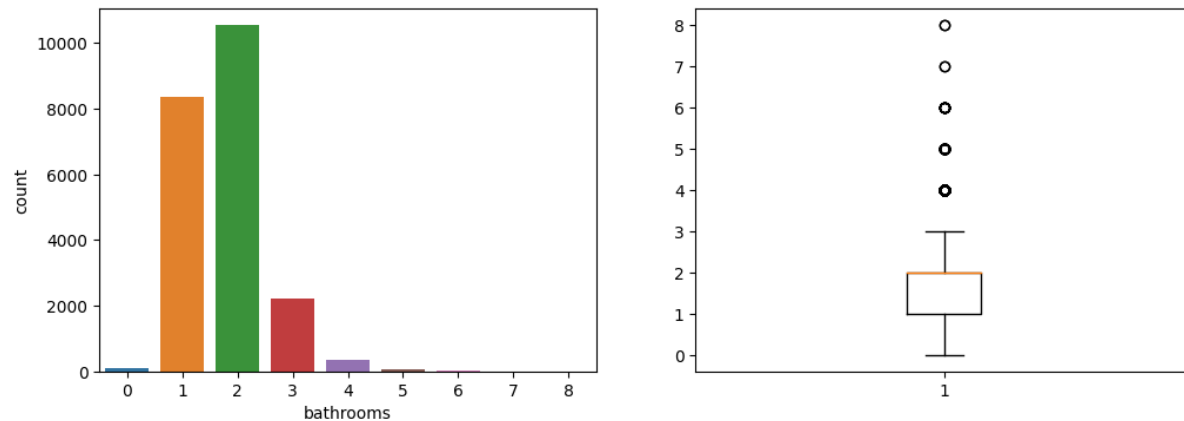
Grafik menunjukkan variabel 'bedrooms' memiliki nilai terbanyak di angka 3 dan 4, variabel 'bedrooms' memiliki *outliers*.

Univariate Analysis variabel 'bathrooms'

```
# univariate analysis variabel 'bathrooms'
f = plt.figure(figsize=(12,4))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
sns.countplot(dataFrame['bathrooms'])

f.add_subplot(1,2,2)
plt.boxplot(dataFrame['bathrooms'])
plt.show()
```



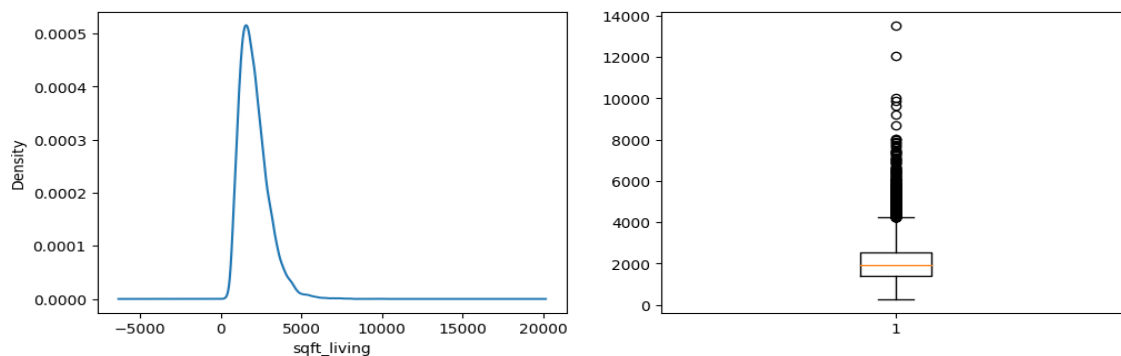
Nilai variabel 'bathrooms' terbanyak adalah 2 dan 1, dan memiliki *outlier* di atas garis maksimum.

Univariate Analysis variabel 'sqft_living'

```
#univariate analysis variabel 'sqft_living'
f = plt.figure(figsize=(12,4))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
dataFrame['sqft_living'].plot(kind='kde')
plt.xlabel('sqft_living')

f.add_subplot(1,2,2)
plt.boxplot(dataFrame['sqft_living'])
plt.show()
```



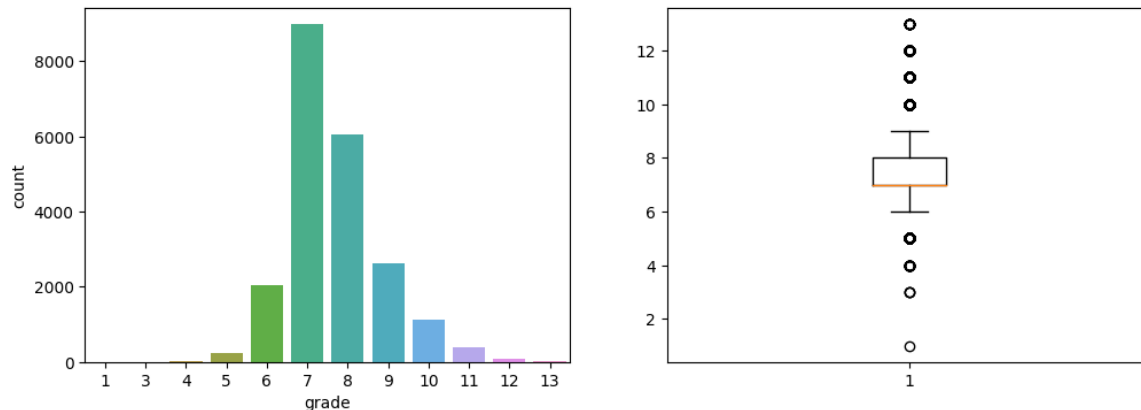
Kepadatan variabel 'sqft_living' ada di angka 2000, dan memiliki banyak outliers di atas garis maksimum.

Univariate Analysis variabel 'grade'

```
# univariate analysis variabel 'grade'
f = plt.figure(figsize=(12,4))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
sns.countplot(df['grade'])

f.add_subplot(1,2,2)
plt.boxplot(df['grade'])
plt.show()
```



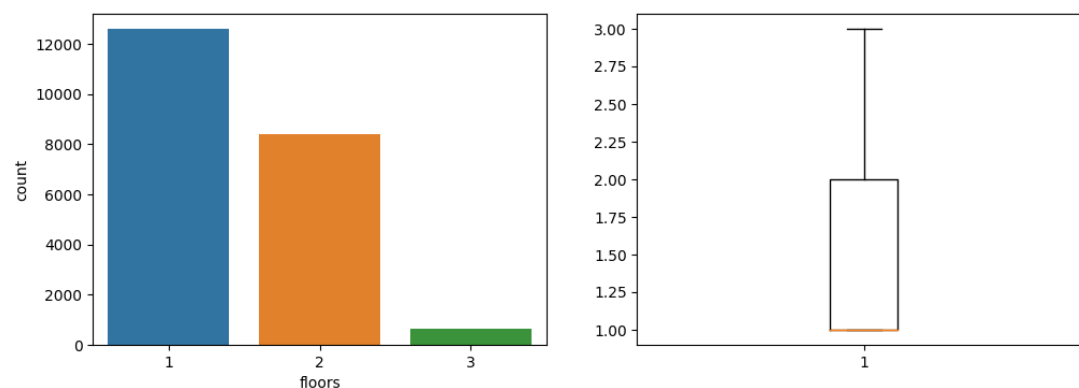
Data ini memiliki kekurangan dimana tidak ada 'grade' yang bernilai 2, kebanyakan rumah memiliki 'grade' di angka 7 dan 8, data memiliki *outlier* di bawah 6 dan di atas 9.

Univariate Analysis variabel 'floors'

```
# univariate analysis variabel 'floors'
f = plt.figure(figsize=(12,4))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
sns.countplot(dataFrame['floors'])

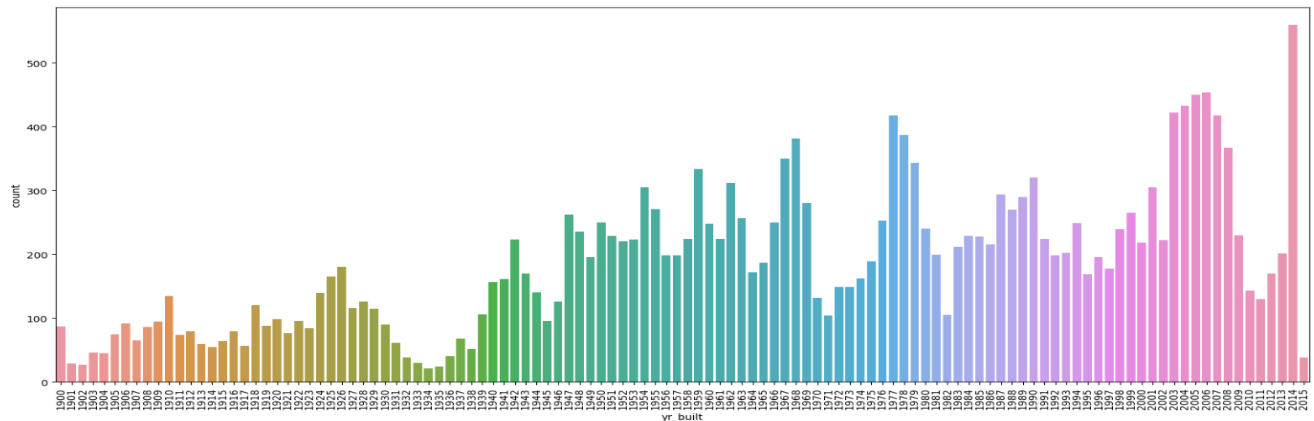
f.add_subplot(1,2,2)
plt.boxplot(dataFrame['floors'])
plt.show()
```



Data menunjukkan kebanyakan rumah hanya memiliki 1 lantai

Univariate Analysis variabel 'yr_built'

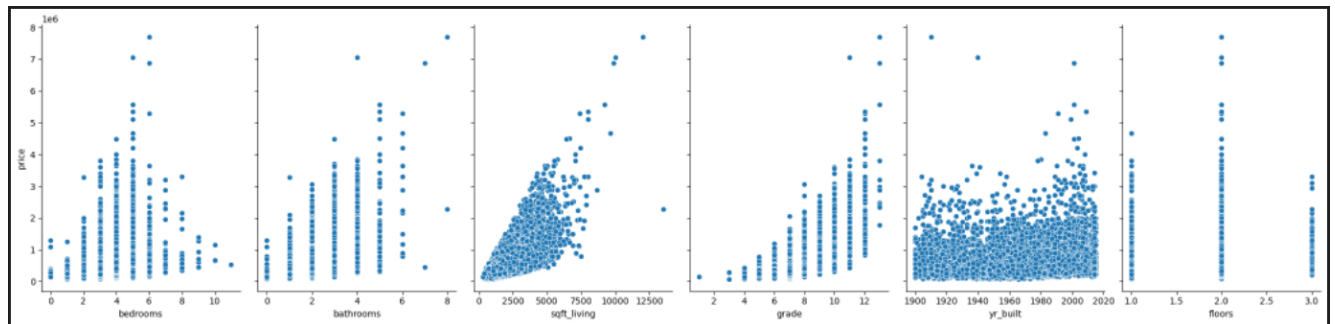
```
# univariate analysis variabel 'yr_built'
# melihat distribusi dari variabel 'yr_built'
f = plt.figure(figsize=(20,8))
sns.countplot(dataFrame['yr_built'])
plt.xticks(rotation=90)
```



Rumah yang dijual dengan 'yr_built' 1901 hingga 1938 memiliki jumlah yang lebih sedikit, mulai tahun 1939 hingga 2014 memiliki jumlah yang lebih banyak

Bivariate Analysis pada masing-masing variabel terhadap harga rumah

```
plt.figure(figsize=(10, 8))
sns.pairplot(data=dataFrame,
             x_vars=['bedrooms', 'bathrooms', 'sqft_living', 'grade', 'yr_built', 'floors'],
             y_vars=['price'],
             size=5,
             aspect=0.7)
```



Dari plot yang dihasilkan didapatkan informasi yaitu:

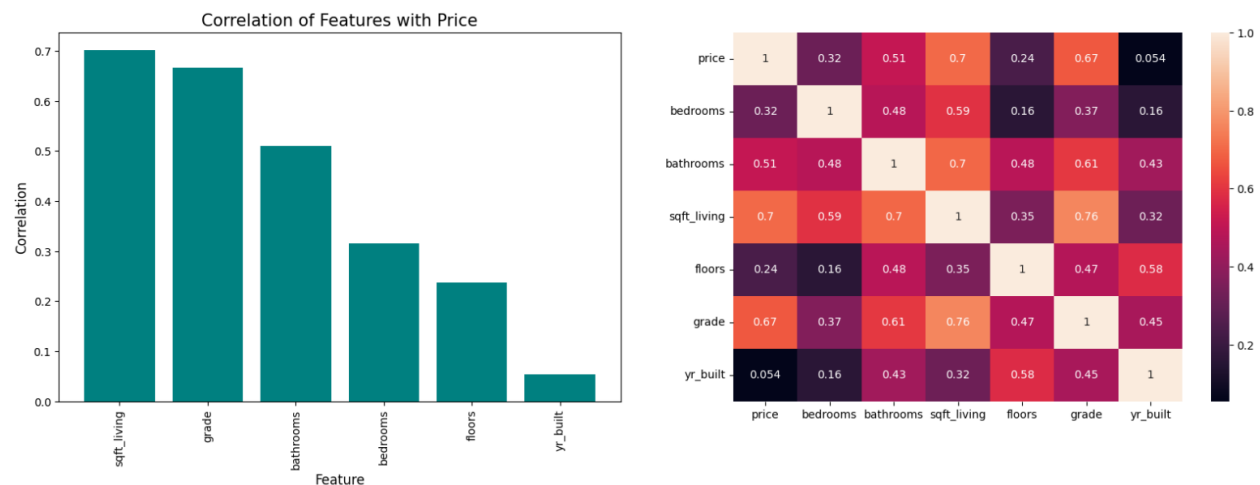
- Rumah dengan jumlah 'bedrooms' 5 dan 6 memiliki harga lebih bervariasi
- Semakin banyak jumlah 'bathroom' semakin bervariasi harganya dan cenderung lebih mahal
- 'sqft_living' hingga 6000 memiliki kepadatan yang tinggi, semakin besar nilai 'sqft_living' maka semakin bervariasi harganya, namun cenderung semakin mahal
- 'yr_built' persebaran yang merata, rumah dengan 'yr_built' tua bisa memiliki harga yang murah maupun mahal
- 'floors' dengan angka 2 memiliki harga yang lebih bervariasi dan cenderung lebih mahal dibanding 1 dan 3

Melihat variabel yang paling berpengaruh terhadap harga

```
# Melihat variabel yang paling berpengaruh terhadap harga
cor = dataframe.corr()['price'].sort_values(ascending=False).drop(['price'])
f=plt.figure(figsize=(20,6))

f.add_subplot(1,2,1)
plt.bar(x=list(cor.index), height=list(cor.values), color='teal')
plt.xticks(rotation=90)
plt.xlabel('Feature', fontsize=12)
plt.ylabel('Correlation', fontsize=12)
plt.title('Correlation of Features with Price', fontsize=15)

f.add_subplot(1,2,2)
sns.heatmap(dataframe.corr(), annot=True)
plt.show()
```



Dari variabel yang digunakan, 'sqft_living' memiliki korelasi tertinggi terhadap harga, sedangkan 'yr_built' memiliki pengaruh paling rendah terhadap harga.

3. Modelling

Pada proses *Modelling* kita akan memisahkan antara variabel independen (yang mempengaruhi) dan variabel dependen/target (yang dipengaruhi), membagi *data train* dan *data test*, membandingkan beberapa algoritma regresi, dan memilih algoritma dengan nilai akurasi tertinggi.

Membuat variabel 'X' yang berisi variabel independen dan variabel 'y' yang berisi variabel dependen/target.

```
# X(Independent variables) and y(target variables)
X = dataframe[['bedrooms', 'bathrooms', 'sqft_living', 'grade', 'floors', 'yr_built']]
y = dataframe['price']
```

Import fungsi `train_test_split` dari `sklearn`.

```
from sklearn.model_selection import train_test_split
```


Membagi seluruh data menjadi data *train* sebanyak 70% dan data *test* sebanyak 30%, pembagian data tidak ada ketentuan khusus namun kebanyakan menggunakan pembagian 70% 30% atau 80% 20%, `random_state` berguna untuk menentukan nilai acak dari data, jika `random_state` ditentukan maka keacakan data akan tetap sama dan hasil training akan tetap sama ketika kode dijalankan.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)
```

Percobaan dengan algoritma *Linear Regression*

```
#menggunakan model linear regression
from sklearn.linear_model import LinearRegression

linReg = LinearRegression()
linReg.fit(X_train, y_train)
```

```
print('akurasi dengan data train:', linReg.score(X_train, y_train)*100)
✓ 0.1s
akurasi dengan data train: 61.74012796448525
```

Hasil *training* mendapatkan nilai sebesar 61.74%

Percobaan dengan algoritma *Gradient Boosting Regressor*

```
#menggunakan model gradient boosting regressor
from sklearn.ensemble import GradientBoostingRegressor

GBR = GradientBoostingRegressor()
GBR.fit(X_train, y_train)
```

```
print('akurasi dengan data train:', GBR.score(X_train, y_train)*100)
✓ 0.1s
akurasi dengan data train: 73.07654423210988
```

Hasil *training* mendapatkan nilai sebesar 73.07%

Percobaan dengan algoritma *Random Forest Regressor*

```
#random forest regressor
from sklearn.ensemble import RandomForestRegressor

RFG = RandomForestRegressor()
RFG.fit(X_train, y_train)
```

```
print('akurasi dengan data train:', RFG.score(X_train, y_train)*100)
✓ 0.4s
akurasi dengan data train: 94.54190211926897
```

Hasil *training* mendapat nilai sebesar 94.54%

Random Forest Regressor menghasilkan nilai terbesar, kita akan coba untuk memberi parameter `n_estimator` dan `random_state`, kemudian melihat apakah dapat menghasilkan nilai lebih tinggi

```
#random forest regressor
from sklearn.ensemble import RandomForestRegressor

RFG = RandomForestRegressor(n_estimators=300, random_state=50)
RFG.fit(X_train, y_train)

print('akurasi dengan data train:', RFG.score(X_train, y_train)*100)
✓ 1.2s

akurasi dengan data train: 94.64860767576566
```

Penggunaan parameter `n_estimator` dapat meningkatkan nilai akurasi, besar `n_estimator` dapat diubah untuk melihat apakah nilai akurasi dapat lebih besar atau bahkan menurunkan nilai akurasi.

Cek hasil *Mean Absolute Percent Error (MAPE)* dan *R squared*

```
#cek MAPE untuk dan r2 random forest regressor
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score

print("mean absolute percentage error:", mean_absolute_percentage_error(y_test, RFG.predict(X_test))*100)
print("r2 score:", r2_score(y_test, RFG.predict(X_test))*100)
✓ 0.9s

mean absolute percentage error: 28.20851748714248
r2 score: 64.71446412917705
```

MAPE merupakan acuan untuk mengetahui seberapa besar kesalahan nilai prediksi dibanding nilai aslinya, nilai MAPE dapat dibagi menjadi 4 kategori yaitu: $<10\%$ = sangat akurat, $10\text{-}20\%$ = baik, $20\text{-}50\%$ = wajar (). Nilai MAPE yang kita dapatkan sebesar 28.20% yang masuk dalam kategori wajar, sebagai referensi dapat dilihat di pada website berikut: <https://www.aindhae.com/2019/12/cara-menghitung-mean-absolute.html>

R squared (r^2) merupakan angka yang berkisar antara 0 sampai 1 (0 sampai 100 jika dalam persen) yang mengindikasikan besarnya kombinasi variabel independen secara bersama – sama mempengaruhi nilai variabel dependen/target semakin mendekati angka satu (100 dalam persen) maka model yang digunakan dalam regresi semakin baik. nilai r^2 dikategorikan kuat jika lebih dari 0,67, moderat jika $0,33 < r^2 < 0,67$, dan lemah jika $0,19 < r^2 < 0,33$. Nilai r^2 yang kita dapatkan sebesar 64.71% dan termasuk dalam moderat, sebagai referensi dapat dilihat pada website berikut:

<https://accounting.binus.ac.id/2021/08/12/memahami-koefisien-determinasi-dalam-regresi-linear/>

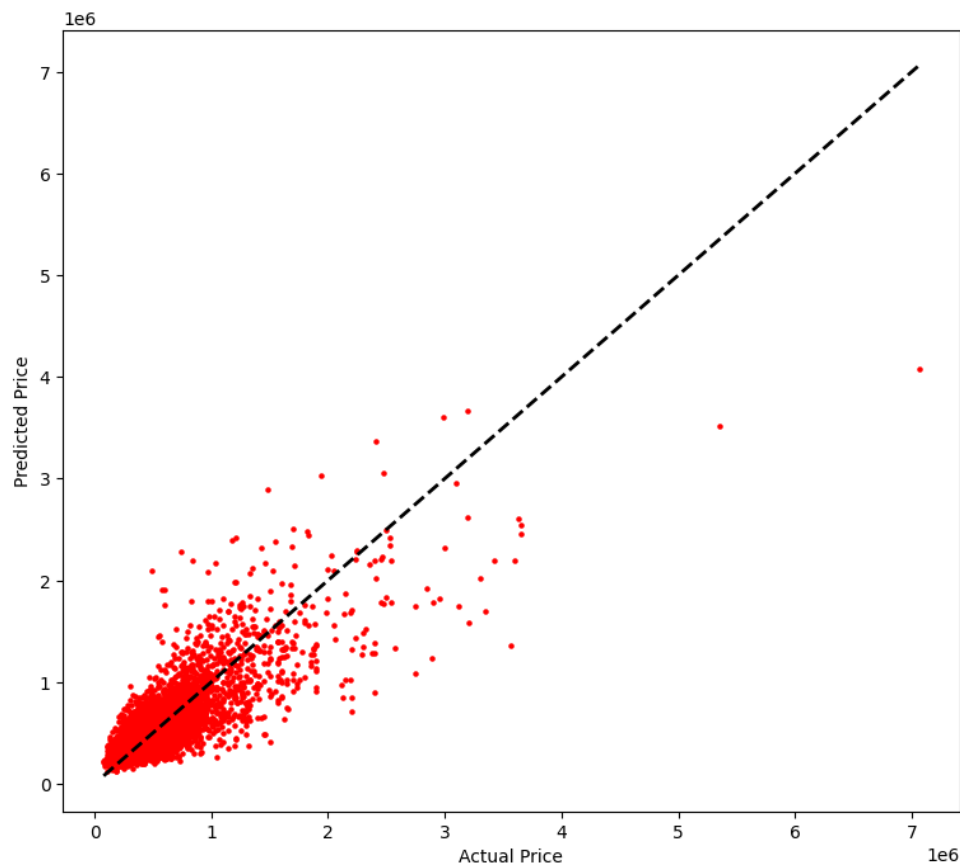
Melihat rata-rata selisih antara harga asli dan harga prediksi dari *data test*

```
#melihat rata-rata persentase selisih antara price dan PredictedPrice
y_df['Difference'] = (y_df['PredictedPrice']-y_df['price'])
y_df['Difference'] = y_df['Difference']/y_df['price']*100
selisih = y_df['Difference'].mean()
print('rata-rata persentase selisih antara harga asli dan harga prediksi: ', selisih, '%')
✓ 0.1s
rata-rata persentase selisih antara harga asli dan harga prediksi: 9.566747327763267 %
```

Melihat persebaran dari harga asli terhadap harga prediksi

```
#melihat plot antara harga prediksi dengan harga real
f = plt.figure(figsize=(20,8))

f.add_subplot(1,2,1)
plt.scatter(y_test, RFG.predict(X_test), color='red', s=5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='black', lw=2)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.show()
```



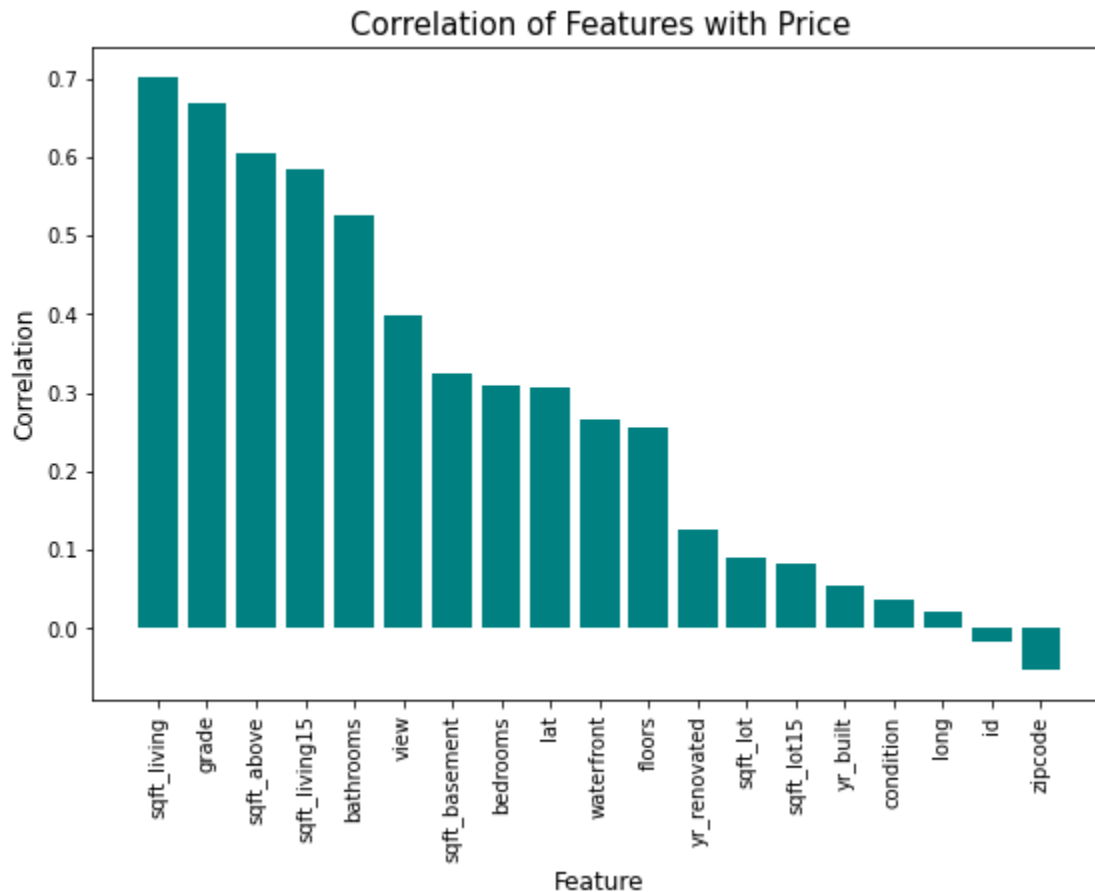
Mencoba Menambahkan Variabel Data

Kita akan melihat apakah dengan penambahan variabel mampu mendapatkan akurasi, nilai MAPE, r^2 , dan persebaran harga asli terhadap harga prediksi menjadi lebih baik.

Melihat korelasi seluruh variabel terhadap harga

```
# Melihat variabel yang paling berpengaruh terhadap harga
cor = dataset.corr()['price'].sort_values(ascending=False).drop(['price'])
f=plt.figure(figsize=(20,6))

f.add_subplot(1,2,1)
plt.bar(x=list(cor.index), height=list(cor.values), color='teal')
plt.xticks(rotation=90)
plt.xlabel('Feature', fontsize=12)
plt.ylabel('Correlation', fontsize=12)
plt.title('Correlation of Features with Price', fontsize=15)
plt.show()
✓ 0.3s
```



Jika kita lihat dari *bar chart* di atas, terdapat variabel dengan korelasi tinggi namun tidak digunakan di model sebelumnya, pada percobaan kedua kita akan menambahkan variabel 'sqft_above' dan 'sqft_living15'.

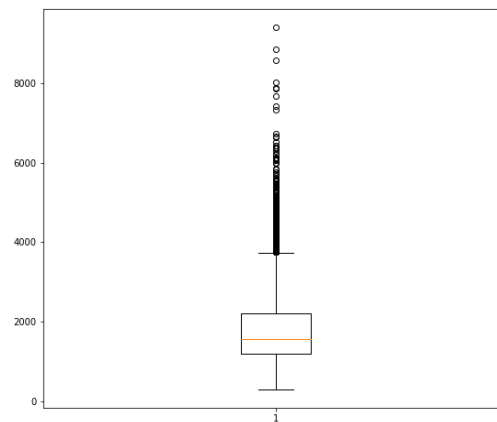
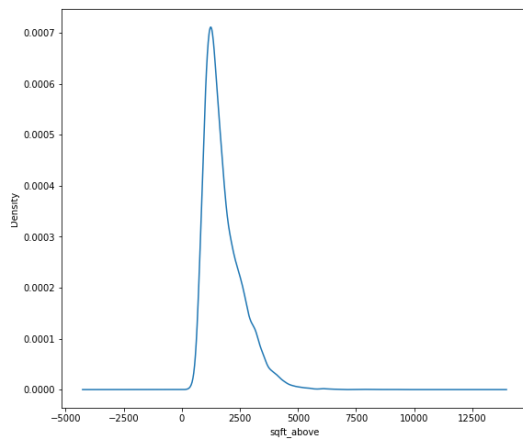
- 'sqft_above' : luas ruangan yang berada di permukaan tanah
- 'sqft_living15' : rata-rata luas tanah dari 15 tetangga terdekat

Univariate Analysis variabel 'sqft_above'

```
#univariate analysis variabel 'sqft_living'
f = plt.figure(figsize=(20,8))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
dataFrame['sqft_above'].plot(kind='kde')
plt.xlabel('sqft_above')

f.add_subplot(1,2,2)
plt.boxplot(dataFrame['sqft_above'])
plt.show()
```



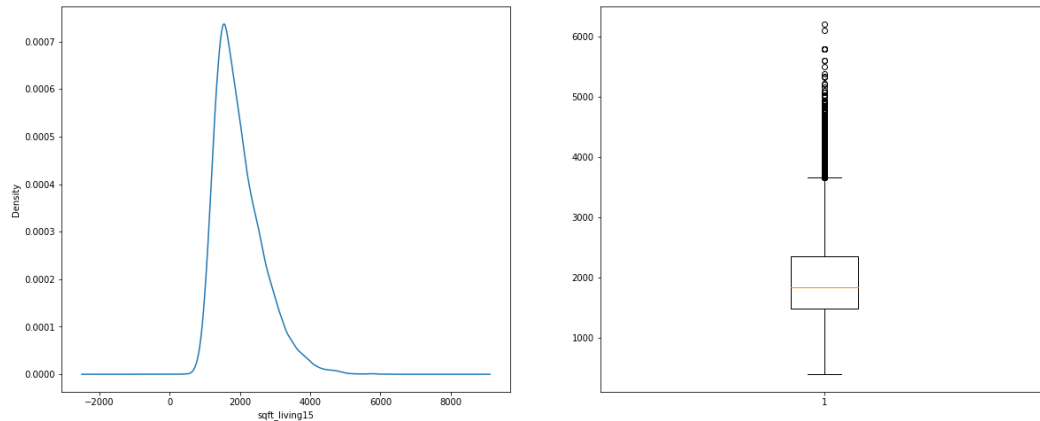
Kepadatan variabel 'sqft_above' berada di angka 1500, terdapat *outlier* di atas 4000.

Univariate Analysis variabel 'sqft_living15'

```
#univariate analysis variabel 'sqft_living'
f = plt.figure(figsize=(20,8))

# add_subplot(baris, kolom, posisi)
f.add_subplot(1,2,1)
dataFrame['sqft_living15'].plot(kind='kde')
plt.xlabel('sqft_living15')

f.add_subplot(1,2,2)
plt.boxplot(dataFrame['sqft_living15'])
plt.show()
```



Kepadatan variabel 'sqft_above' berada di angka 1500, terdapat *outlier* di atas 3600.\

Training model dengan algoritma *Random Forest Regression*

```
#random forest regressor
from sklearn.ensemble import RandomForestRegressor

RFG = RandomForestRegressor(n_estimators=300, random_state=50)
RFG.fit(X_train, y_train)

print('akurasi dengan data train:', RFG.score(X_train, y_train)*100)

akurasi dengan data train: 95.57677709489144
```

Dengan pembagian dan parameter yang sama, penambahan variabel 'sqft_above' dan 'sqft_living15' meningkatkan nilai akurasi hingga 95.57%

Cek hasil MAPE dan *R Square*

```
#cek MAPE dan r2 untuk random forest regressor
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score

print("mean absolute percentage error:", mean_absolute_percentage_error(y_test, RFG.predict(X_test))*100)
print("r2 score:", r2_score(y_test, RFG.predict(X_test))*100)

mean absolute percentage error: 25.48307354803152
r2 score: 70.2785467115288
```

MAPE mendapatkan nilai sebesar 25.48% yang termasuk dalam kategori wajar, *R Square* mendapatkan nilai sebesar 70.27% yang termasuk dalam kategori kuat

Melihat rata-rata selisih antara harga asli dan harga prediksi dari *data test*

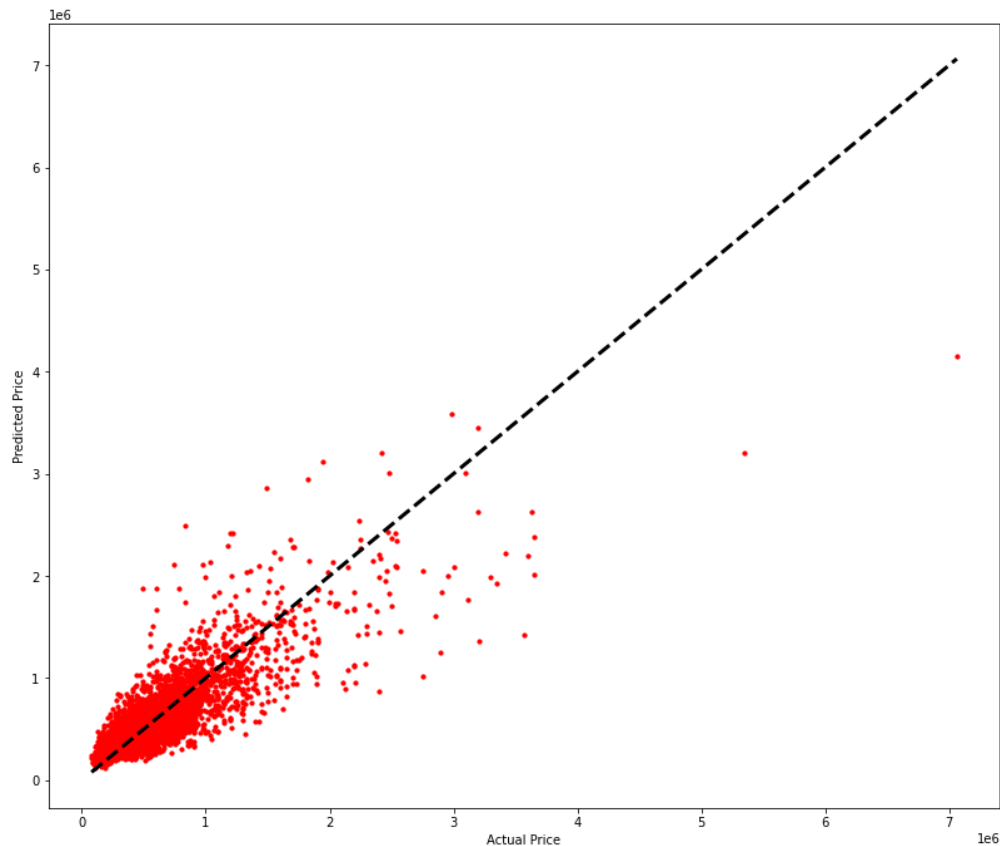
```
#melihat rata-rata persentase selisih antara price dan PredictedPrice
y_df['Difference'] = (y_df['PredictedPrice']-y_df['price'])
y_df['Difference'] = y_df['Difference']/y_df['price']*100
selisih = y_df['Difference'].mean()
print('rata-rata persentase selisih antara harga asli dan harga prediksi: ', selisih, '%')

rata-rata persentase selisih antara harga asli dan harga prediksi: 8.964860987347253 %
```

Melihat plot antara harga prediksi dengan harga asli

```
#melihat plot antara harga prediksi dengan harga real
f = plt.figure(figsize=(30,11.5))

f.add_subplot(1,2,1)
plt.scatter(y_test, RFG.predict(X_test), color='red', s=10)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='black', lw=3)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.show()
```



Tabel Perbandingan

Nilai akurasi	
Tanpa penambahan variabel	Penambahan 2 variabel
94.64%	95.57%

Penambahan variabel 'sqft_above' dan 'sqft_living15' meningkatkan nilai akurasi dari *Random Forest Regression* sebesar 0.93%.

Nilai MAPE	
Tanpa penambahan variabel	Penambahan 2 variabel
28.20%	25.48%

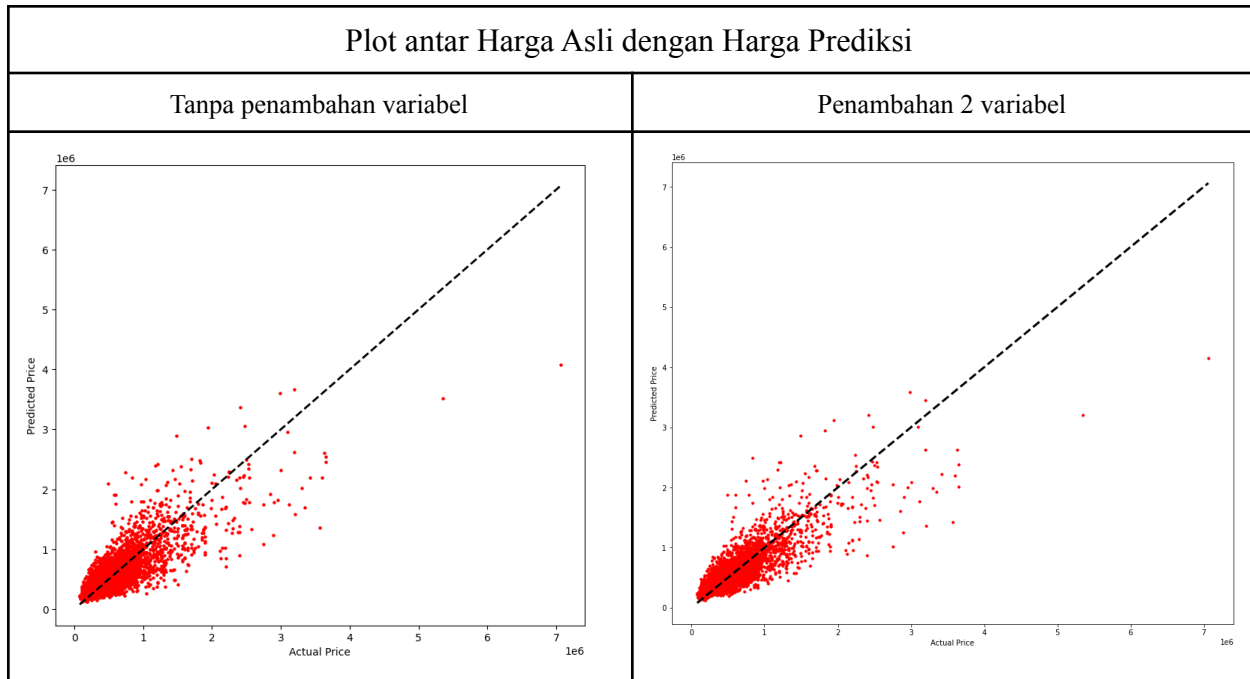
Penambahan variabel 'sqft_above' dan 'sqft_living15' dapat menurunkan nilai MAPE hingga 2.72%, semakin menurun nilai MAPE menunjukkan semakin kecil kesalahan dalam prediksi.

Nilai <i>R Square</i>	
Tanpa penambahan variabel	Penambahan 2 variabel
64.71%	70.27%

Penambahan variabel 'sqft_above' dan 'sqft_living15' dapat meningkatkan nilai *R Square* hingga 5.56%, dari kategori moderat menjadi kategori kuat.

Rata-rata Kesalahan Prediksi	
Tanpa penambahan variabel	Penambahan 2 variabel
9.56%	8.96%

Penambahan variabel 'sqft_above' dan 'sqft_living15' dapat menurunkan rata-rata nilai kesalahan prediksi hingga 0.6%



Penambahan variabel 'sqft_above' dan 'sqft_living15' membuat plot semakin rapat dengan garis acuan, menunjukkan bahwa harga prediksi dengan harga asli memiliki selisih yang lebih kecil.

4. Menyimpan Model ke Dalam *File* Pickle dan Kompres dengan Joblib

Pickle merupakan *library* python yang dapat menyimpan model *training* mencari satu file terpisah sehingga dapat diterapkan di berbagai *platform* (contohnya adalah *website*), sedangkan Joblib dapat digunakan untuk memperkecil ukuran model.

Import pickle

```
import pickle
```

Membuat *file* untuk menyimpan model, dan memasukkan model ke *file* tersebut

```
import joblib

filename = 'model_prediksi_rumah_joblib.pkl'
joblib.dump(RFG, 'model_prediksi_rumah_joblib.pkl', compress=9)
```

Mencoba prediksi harga dengan *file* pickle yang telah dibuat

```
loaded_model = pickle.load(open('model_prediksi_rumah_new.pkl', 'rb'))
# 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
print(loaded_model.predict([[3, 1, 1660, 1, 7, 960, 1510, 1941]]))

[512195.20666667]
```

File pickle yang telah dibuat dapat berjalan dengan baik

Membuat tampilan *website* agar *user* dapat melakukan prediksi dengan mudah, *website* dibuat dengan library streamlit dari python agar lebih mudah.

```
1 import streamlit as st
2 import pickle
3 import numpy as np
4 import sklearn
5 from sklearn.preprocessing import StandardScaler
6 from PIL import Image
7 import pandas as pd
8 import time
9 import matplotlib.pyplot as plt
10
11 #load model yang sudah di train
12 pickle_in = open("model_prediksi_rumah_new.pkl", "rb")
13 random_forest_regression_model = pickle.load(pickle_in)
14
15 def Home():
16     return ('Sabar gan')
17 # 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
18 def predicting(bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built):
19     prediction = random_forest_regression_model.predict([[bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built]])
20     output = round(prediction[0], 2)
21     return output
22 # 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
23 def main():
24     html_temp = """
25     <div style="background-color:navy;padding:10px">
26     <h1 style="color:white;text-align:center;">Prediksi Harga Rumah ($) </h1>
27     </div>
28     """
29
30     st.markdown(html_temp, unsafe_allow_html=True)
31
32     st.subheader("Jumlah Kamar") #bedrooms
33     bedrooms = st.slider("", min_value=0, max_value=11, value=0, step=1)
34
35     st.subheader("Jumlah Kamar Mandi") #bathrooms
36     bathrooms = st.slider("", min_value=0, max_value=8, value=0, step=1)
37
38     st.subheader("Luas") #sqft_living
39     sqft_living = st.text_input("luas bangunan secara keseluruhan (sqft)", "")
40
41     st.subheader("Jumlah Lantai") #floors
42     floors = st.slider("", min_value=1, max_value=3, value=1, step=1)
43
44     st.subheader("Kelas / Grade") #grade
45     grade = st.slider("", min_value=1, max_value=13, value=1, step=1)
46
47     st.subheader("Luas bangunan di atas tanah") #sqft_above
48     sqft_above = st.text_input("sqft", "")
49
50     st.subheader("Rata-rata luas sekitar") #sqft_living15
51     sqft_living15 = st.text_input("rata-rata luas tanah 15 tetangga terdekat (sqft)", "")
52
53     st.subheader("Tahun Dibangun") #yr_built
54     yr_built = st.text_input("", "")
55
56     result = ""
57     if st.button("Perkiraan harga"):
58
59         my_bar = st.progress(0)
60         for percent_complete in range(100):
61             time.sleep(0.01)
62             my_bar.progress(percent_complete + 1)
63         result = predicting(bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built)
64
65     st.success('Hasil Prediksi : {} Dollars'.format(result))
66
67
68 nav = st.sidebar.radio("Halaman", ["Home", "Data Sample", "About Me"])
69 if nav == "Home":
70     if __name__ == "__main__":
71         main()
72
73 if nav == "Data Sample":
74     st.title("Data Sample")
75     data = pd.read_csv("dataFrame_test_train_new.csv", usecols=['sqft_living', 'grade', 'sqft_above', 'sqft_living15', 'bathrooms', 'bedrooms', 'floors', 'price', 'yr_built'])
76     st.table(data.sample(100))
77     plt.show()
78
79 # 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
80 if nav == "About Me":
81     st.title(" Prediksi Harga Rumah")
82     st.subheader("Gilang Agung Saputra (672019229)")
```

Berikut adalah tampilan dari *website* yang telah dibuat
Halaman pertama:

Prediksi Harga Rumah (\$)

Jumlah Kamar

0

11

Jumlah Kamar Mandi

0

8

Luas

luas bangunan secara keseluruhan (sqft)

Jumlah Lantai

1

3

Kelas / Grade

1

13

Luas bangunan di atas tanah

(sqft)

Rata-rata luas sekitar

rata-rata luas tanah 15 tetangga terdekat (sqft)

Tahun Dibangun

Perkiraan harga

Hasil Prediksi : Dollars

Made with Streamlit

Halaman kedua:

Halaman

- Home
- Data Sample
- About Me

Data Sample

	price	bedrooms	bathrooms	sqft_living	floors	grade	sqft_above	yr_built	sqft_living15
11685	1,135,000.0000	6	4	6900	2	9	4820	2002	4170
13440	720,000.0000	3	2	2120	2	8	2120	1981	2620
20345	400,200.0000	4	3	2260	2	7	2260	2013	1270
21372	325,000.0000	4	3	2800	2	7	2800	2011	2380
20263	649,000.0000	4	2	3130	2	9	3130	2014	1570
7133	445,000.0000	2	1	930	1	6	930	1918	1900
12762	355,000.0000	3	1	1370	2	7	1370	1982	1370
15027	470,000.0000	3	3	1710	2	8	1360	2003	1420
15560	1,089,000.0000	5	3	5600	2	10	3440	1988	3470
15107	235,000.0000	3	2	1110	1	7	1110	1990	1390
15919	239,000.0000	2	1	710	1	6	710	1953	1500
15474	225,900.0000	3	1	1510	1	7	1010	1963	1290
15643	455,000.0000	2	1	1260	1	7	1260	1941	1640
3746	320,000.0000	3	2	1820	1	7	1820	1972	2190
11684	510,000.0000	4	2	3180	2	7	3180	2001	2610
13328	280,000.0000	4	2	2200	2	7	2200	2008	2200
10221	270,000.0000	4	1	1850	1	7	1100	1956	2260
3223	262,000.0000	1	0	520	1	3	520	1920	1340
3411	459,500.0000	2	1	1250	1	7	850	1929	1370
99	437,500.0000	3	2	2320	2	9	2320	1992	2550
7101	199,950.0000	2	1	1010	1	6	1010	1961	1040
6328	372,000.0000	3	2	1920	2	7	1920	1990	2370

Halaman ketiga:

Halaman

- Home
- Data Sample
- About Me

Prediksi Harga Rumah

Gilang Agung Saputra (672019229)

Made with Streamlit

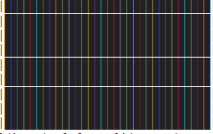
Kita coba di localhost dengan membuka CommandPrompt ke path folder kita, lalu ketik “streamlit run *nama_file.py*”

Seluruh kode dan data bisa di akses di <https://github.com/GilangAgungS/house-prediction-fix1>

Hosting Website dari Google Colaboratory dengan LocalTunnel

Unggah file ipynb dan data yang digunakan untuk *train test* (bisa upload ke Google Colab secara temporer atau ke Google Drive), install streamlit pada Colab dan buat file baru bernama “app.py”(atau bisa nama lain berformat .py) dengan kode sebagai berikut:

```
[68] !pip install streamlit -q
```



```
9.1 MB 38.9 MB/s
235 kB 69.8 MB/s
164 kB 78.8 MB/s
181 kB 72.5 MB/s
4.3 MB 59.8 MB/s
78 kB 9.4 MB/s
63 kB 1.6 MB/s
1.6 MB 85.7 MB/s
51 kB 8.2 MB/s
Building wheel for validators (setup.py) ... done
```

```
[69] %%writefile app.py
import streamlit as st

import streamlit as st
import pandas as pd
import time
import matplotlib.pyplot as plt
import joblib

#load model yang sudah di train
#pickle_in = open("model_prediksi_rumah_new.pkl", "rb")
#random_forest_regression_model = pickle.load(pickle_in)

loaded_model = joblib.load('model_prediksi_rumah_joblib.pkl')

def Home():
    return ('Sabar gan')
# 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
def predicting(bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built):
    prediction = loaded_model.predict([[bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built]])
    output = round(prediction[0], 2)
    return output
# 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
def main():
    html_temp = """
    <div style="background-color:navy;padding:10px">
    <h1 style="color:white;text-align:center;">Prediksi Harga Rumah ($) </h1>
    </div>
    """
```

```

st.markdown(html_temp, unsafe_allow_html=True)

st.subheader("Jumlah Kamar") #bedrooms
bedrooms = st.slider("", min_value=0, max_value=11, value=0, step=1)

st.subheader("Jumlah Kamar Mandi") #bathrooms
bathrooms = st.slider("", min_value=0, max_value=8, value=0, step=1)

st.subheader("Luas") #sqft_living
sqft_living = st.text_input("luas bangunan secara keseluruhan (sqft)", "")

st.subheader("Jumlah Lantai") #floors
floors = st.slider("", min_value=1, max_value=3, value=1, step=1)

st.subheader("Kelas / Grade") #grade
grade = st.slider("", min_value=1, max_value=13, value=1, step=1)

st.subheader("Luas bangunan di atas tanah")#sqft_above
sqft_above = st.text_input("(sqft)", "")

st.subheader("Rata-rata luas sekitar") #sqft_living15
sqft_living15 = st.text_input("rata-rata luas tanah 15 tetangga terdekat (sqft)", "")

st.subheader("Tahun Dibangun") #yr_built
yr_built = st.text_input("tahun rumah dibangun", "")

result = ""
if st.button("Perkiraan harga"):

    my_bar = st.progress(0)
    for percent_complete in range(100):
        time.sleep(0.01)
        my_bar.progress(percent_complete + 1)
    result = predicting(bedrooms, bathrooms, sqft_living, floors, grade, sqft_above, sqft_living15, yr_built)

st.success('Hasil Prediksi : {} Dollars'.format(result))

nav = st.sidebar.radio("Halaman", ["Home", "Data Sample", "About Me"])
if nav == "Home":
    if __name__ == "__main__":
        main()
    ..

if nav == "Data Sample":
    st.title("Data Sample")
    data = pd.read_csv("dataFrame_test_train_new.csv", usecols=['bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'])
    st.table(data.sample(100))
    plt.show()

# 'bedrooms', 'bathrooms', 'sqft_living', 'floors', 'grade', 'sqft_above', 'sqft_living15', 'yr_built'
if nav == "About Me":
    st.title(" Prediksi Harga Rumah")
    st.subheader("Gilang Agung Saputra (672019229)")

```

Atau dapat dengan mengetik “%%writefile app.py” kemudian salin kode untuk GUI yang telah dibuat, jalankan *cell* kemudian jalankan kode dibawah ini:

```
!streamlit run app.py & npx localtunnel --port 8501
```

```

2022-08-29 01:00:13.310 INFO    numexpr.utils: NumExpr defaulting to 2 threads.
npx: installed 22 in 6.109s
your url is: https://icy-moose-wink-34-106-75-248.local.lt

```

You can now view your Streamlit app in your browser.

Network URL: <http://172.28.0.2:8501>
 External URL: <http://34.106.75.248:8501>

Tunggu hingga muncul url, *Network URL* dan *External URL*, klik url tersebut

Klik tombol “Click to Continue”

icy-moose-wink-34-106-75-248.local.it

Friendly Reminder

This website is served via a [localtunnel](#). This is just a reminder to always check the website address you're giving personal, financial, or login details to is actually the real/official website.

Phishing pages often look similar to pages of known banks, social networks, email portals or other trusted institutions in order to acquire personal information such as usernames, passwords or credit card details.

Please proceed with caution.

Click to Continue

If you're the developer...

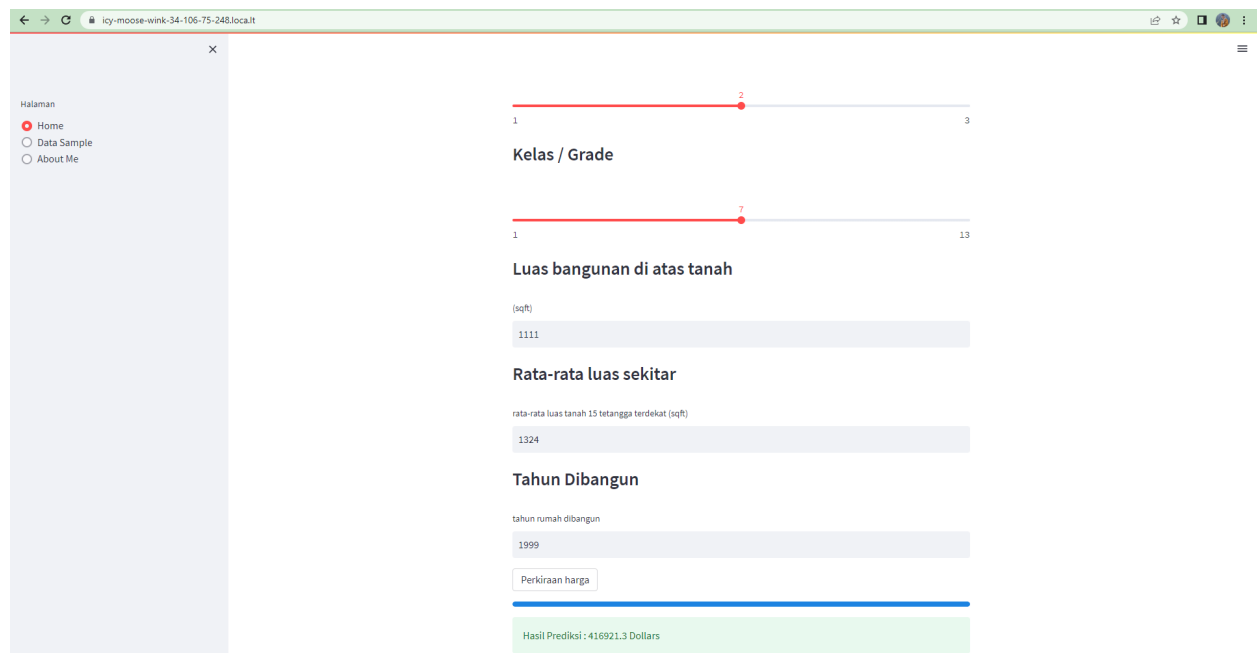
You and other visitors will only see this page from a standard web browser once per IP every 7 days.

Webhook, IPN, and other non-browser requests "should" be directly tunneled to your localhost. If your webhook/ipn provider happens to send requests using a real browser user-agent header, those requests will unfortunately also be blocked / be forced to see this tunnel reminder page. FYI, this page returns a 401 HTTP Status.

Options to bypass this page:

1. Set and send a `Bypass-Tunnel-Reminder` request header (its value can be anything).
2. or, Set and send a custom / non-standard browser `User-Agent` request header.

Berikut adalah tampilan dari *website*



Website berhasil dijalankan dan dapat diakses oleh banyak orang, namun localtunnel memiliki kekurangan, jika Google Colab berhenti bekerja maka website tidak akan bisa diakses, dan jika *cell* Colab dijalankan kembali maka url akan berubah. Tutorial menjalankan localtunnel melalui Google Colab dapat dilihat di video berikut:

https://www.youtube.com/watch?v=NEhrkeF2o_M

Link github: <https://github.com/GilangAgungS/house-prediction-fix1>

