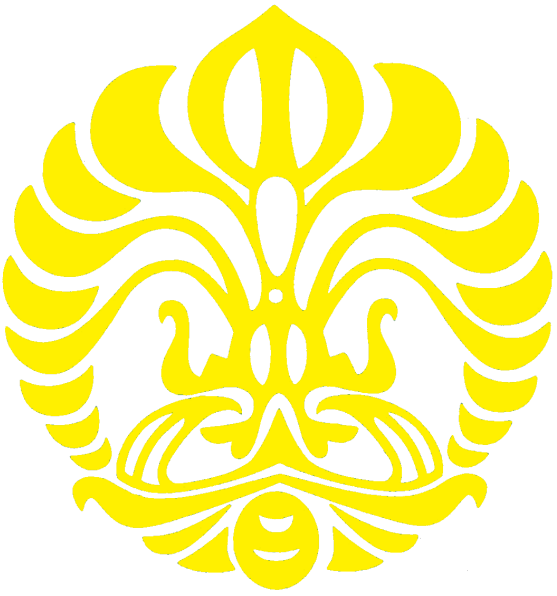
Data Preparation & Result Visualization

Data Mining and Business Intelligenc



Disusun Oleh:

Amira Husna Nur Adilah

Fakultas Ilmu Komputer

Universitas Indonesia

Depok

2022

Daftar Isi

1. Pendahuluan

Sebelum menggunakan data untuk dianalisis dan diproses, terdapat tahapan yang harus dilakukan untuk memastikan bahwa data tersebut dapat digunakan dengan baik. Tahapan tersebut dikenal dengan *data preparation*. *Data preparation* bertujuan untuk mempersiapkan *raw data* menjadi data yang siap untuk diolah. Terdapat beberapa tahapan yang umum dilakukan pada *data preparation*, diantaranya yaitu *collecting*, *cleaning*, dan *labeling* (AWS, 2022).

Data preparation penting dilakukan karena data yang digunakan untuk menganalisis akan memengaruhi hasil kesimpulan. Data yang tidak dipersiapkan terlebih dahulu dapat menimbulkan bias dan kesalahan interpretasi pada kesimpulan dan keputusan yang dihasilkan. *Data preparation* yang dilakukan dengan baik akan meningkatkan keakuratan dari data tersebut.

Pada laporan ini akan dijelaskan mengenai tahapan pada *data preparation* menggunakan bahasa pemrograman Python dengan menggunakan Google Colab sebagai notebook serta akan ditampilkan dampak dari *data preparation* terhadap *raw data* menggunakan Tableau.

2. Deskripsi data

Dataset yang digunakan yaitu dataset_harga_rumah bertipe .csv. Dataset tersebut memiliki 29.451 baris dan 12 kolom dengan deskripsi setiap kolomnya adalah sebagai berikut:

- diposting_oleh: merupakan kolom yang berisi *value* terkait dengan pemilik properti rumah. Terdapat tiga jenis pemilik, yaitu *owner*, *dealer*, dan *builder*.
- sedang_pembangunan: merupakan kolom yang berisi *value* berupa status pembangunan terkait properti (rumah) dengan 0 yang menandakan bahwa properti tidak sedang dalam masa pembangunan dan 1 jika kebalikannya.
- disetujui_pemerintah: merupakan kolom yang berisi *value* berupa status persetujuan pemerintah terkait properti (rumah) dengan 0 yang menandakan bahwa properti belum disetujui dan 1 jika kebalikannya.
- total_ruangan: merupakan kolom yang berisi *value* berupa banyaknya ruangan pada properti (rumah).

- tipe_a_atau_b: merupakan kolom yang berisi *value* berupa tipe rumah. Terdapat dua tipe, yaitu tipe a dan tipe b.
- kaki_persegi: merupakan kolom yang berisi *value* berupa luas rumah dalam satuan *square feet*.
- siap_pindah: merupakan kolom yang berisi *value* berupa status kesiapan huni dari suatu rumah dengan 0 yang menandakan bahwa properti belum siap dihuni dan 1 jika kebalikannya.
- dijual_kembali: merupakan kolom yang berisi *value* untuk menandakan bahwa suatu rumah pernah terjual dengan 0 yang menandakan bahwa properti belum pernah terjual dan 1 jika kebalikannya.
- alamat: merupakan kolom yang berisi *value* berupa alamat dari suatu rumah.
- longitude: merupakan kolom yang berisi *value* berupa longitude dari suatu rumah.
- latitude: merupakan kolom yang berisi *value* berupa latitude dari suatu rumah.
- harga: merupakan kolom yang berisi *value* berupa harga dari suatu rumah dalam satuan rupee.

3. Proses Data Preparation

Berikut penjelasan dan screenshot dari setiap langkah pada *data preparation* menggunakan dataset dataset_harga_rumah.csv:

3.1. Import Libraries

Pertama, libraries perlu di-*import* untuk memudahkan dan membuat proses lebih sederhana. *Libraries* yang diimport pada *data preparation* kali ini terdiri dari *Stats* untuk digunakan dalam memudahkan perhitungan statistika, *Pandas* yang digunakan untuk mengolah data dalam bentuk tabel berupa *Data Frame*, *Numpy* yang digunakan untuk mengolah data dengan operasi vektor dan matriks, *Matplotlib* digunakan untuk menampilkan data secara visual, *Seaborn* yang fungsinya sama dengan Matplotlib, sklearn untuk digunakan dalam encoding, serta geopandas, geopy dan nominatim yang digunakan untuk me-*replace* null values menggunakan informasi lokasi yang ada.

```
# import libraries needed
from scipy import stats
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
import geopandas as gpd
import geopy
from geopy.geocoders import Nominatim
```

3.2. Membaca File Dataset

Setelah libraries diimport, langkah selanjutnya yaitu meng-*import* dataset. Data disimpan di dalam Google Drive sehingga perlu menggunakan *path* tempat dataset disimpan dan menggunakan modul drive dari libraries google.colab untuk menghubungkan Google Colab dengan drive. Dataset akan disimpan dalam variabel df dan dibaca *menggunakan* read_csv karena file dataset yang digunakan bertipe .csv.

```
from google.colab import drive
drive.mount('/content/drive', force_remount = True)
```

```
df = pd.read_csv('drive/MyDrive/Coolyeah/PDIB/Tugas 4/dataset_harga_rumah.csv', low_memory = False)
```

3.3. Menampilkan Data

Setelah data berhasil di-*import* dan di-*assign* ke variabel df, selanjutnya data dapat ditampilkan. *Karena* data tersebut memiliki baris yang cukup banyak, maka hanya ditampilkan lima baris awal menggunakan df.head() dan lima baris akhir menggunakan df.tail().

```
# get first view about data
df.head()
```

	diposting_oleh	sedang_pembangunan	disetujui_pemerintah	total_ruangan	tipe_a_atau_b	kaki_persegi	siap_pindah	dijual_kembali	alamat	longitude
0	Owner	0.0	0	2.0	tipe_a	1300.236407	1	1	Ksfc Layout,Bangalore	12.969910
1	Dealer	0.0	0	2.0	tipe_a	1275.0	1	1	Vishweshwara Nagar,Mysore	12.274538
2	Owner	0.0	0	2.0	tipe_a	933.1597222	1	1	Jigani,Bangalore	12.778033
3	Owner	0.0	1	2.0	tipe_a	929.9211427	1	1	Sector-1 Vaishali,Ghaziabad	28.642300
4	Dealer	1.0	0	2.0	tipe_a	999.009247	0	1	New Town,Kolkata	22.592200

```
df.tail()
```

	diposting_oleh	sedang_pembangunan	disetujui_pemerintah	total_ruangan	tipe_a_atau_b	kaki_persegi	siap_pindah	dijual_kembali	alamat	long
29446	Owner	0.0	0	3.0	tipe_a	2500.0	1	1	Shamshabad Road,Agra	27.1
29447	Owner	0.0	0	2.0	tipe_a	769.2307692	1	1	E3-108, Lake View Recidency,,Vapi	39.9
29448	Dealer	0.0	0	2.0	tipe_a	1022.641509	1	1	Ajmer Road,Jaipur	26.9
29449	Owner	0.0	0	2.0	tipe_a	927.0790093	1	1	Sholinganallur,Chennai	12.9
29450	Dealer	0.0	1	2.0	tipe_a	896.7741935	1	1	Jagatpura,Jaipur	26.8

3.4. Menampilkan Informasi dari Dataset

Dataset dapat dieksplorasi terlebih dahulu agar dapat mengetahui masalah yang dapat diperbaiki dari dataset. Pertama, dimensi data perlu diketahui. DImensi tersebut didapat dari .shape seperti berikut:

```
df.shape
```
```
(29451, 12)
```

Dari df.shape diketahui bahwa baris dari dataset ada 29451 dan memiliki 12 kolom. Selanjutnya yaitu melihat jumlah data dan tipe data per kolom menggunakan df.info().

```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29451 entries, 0 to 29450
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   diposting_oleh       29451 non-null  object 
 1   sedang_pembangunan   27476 non-null  float64
 2   disetujui_pemerintah 29451 non-null  int64  
 3   total_ruangan        28071 non-null  float64
 4   tipe_a_atau_b        29451 non-null  object 
 5   kaki_persegi         27765 non-null  object 
 6   siap_pindah          29451 non-null  object 
 7   dijual_kembali       29451 non-null  object 
 8   alamat               27436 non-null  object 
 9   longitude            29451 non-null  float64
 10  latitude             26008 non-null  float64
 11  harga                29451 non-null  float64
dtypes: float64(5), int64(1), object(6)
memory usage: 2.7+ MB
```

Selain itu, dapat diketahui juga statistika deskriptif dari setiap kolom menggunakan df.describe()

df.describe untuk data numerik:

```
df.describe()
```

	sedang_pembangunan	disetujui_pemerintah	total_ruangan	longitude	latitude	harga
count	27476.000000	29451.000000	28071.000000	29451.000000	26008.000000	29451.000000
mean	0.179684	0.317918	2.392220	21.300255	76.820157	142.898746
std	0.383931	0.465675	0.882229	6.205306	10.697823	656.880713
min	0.000000	0.000000	1.000000	-37.713008	-121.761248	0.250000
25%	0.000000	0.000000	2.000000	18.452663	73.798100	38.000000
50%	0.000000	0.000000	2.000000	20.750000	77.326330	62.000000
75%	0.000000	1.000000	3.000000	26.900926	77.943705	100.000000
max	1.000000	1.000000	20.000000	59.912884	152.962676	30000.000000

df.describe untuk data kategorik:

```
[ ] df.describe(include='object')
```

	diposting_oleh	tipe_a_atau_b	kaki_persegi	siap_pindah	dijual_kembali	alamat
count	29451	29451	27765	29451	29451	27436
unique	3	2	18115	9	5	6648
top	Dealer	tipe_a	1000.0	1	1	Zirakpur,Chandigarh
freq	18291	29427	446	21363	23610	473

Kategori atau nilai unik dari setiap kolom juga dapat dicek seperti berikut:

```
# Mengecek nilai unik daris etiap kolom
for col in df.columns:
  print(col+" unique")
  print(df[col].unique())
  print()
```

```
diposting_oleh unique
['Owner' 'Dealer' 'Builder']

sedang_pembangunan unique
[ 0.  1. nan]

disetujui_pemerintah unique
[0 1]

total_ruangan unique
[ 2.  3.  1.  4.  5.  6. nan 12.  8. 20. 10.  7.  9. 13. 17. 15. 11.]
```

```
tipe_a_atau_b unique
['tipe_a' 'tipe_b']

kaki_persegi unique
['1300.236407' '1275.0' '933.1597222' ... '1022.641509' '927.0790093'
 '896.7741935']

siap_pindah unique
['1' '0' 'testes' 'a' 'awa' '1-1-1-' 'benar' 'salah' 'asik']

dijual_kembali unique
['1' '0' 'Aku paling benar' 'SALAH' 'Semangat Dek']

alamat unique
['Ksfc Layout,Bangalore' 'Vishweshwara Nagar,Mysore' 'Jigani,Bangalore'
 ... 'west mambalam,Chennai' 'Gandhi Nagar,Gulbarga'
 'E3-108, Lake View Recidency,,Vapi']

longitude unique
[12.96991  12.274538 12.778033 ... 18.9737   17.357159 39.945409]

latitude unique
[ 77.59796   76.644605  77.632191 ...  73.3321    76.841908 -86.150721]

harga unique
[  55.    51.    43.  ... 1170.  8660.    18.3]
```

3.5. Melakukan Validasi Terhadap Data

Dari informasi dataset, ditemukan banyak kejanggalan. Berikut kejanggalan yang dapat diidentifikasi:

- Longitude memiliki rentang -180º hingga 180º, sedangkan latitude memiliki rentang -90º hingga 90º. Dapat dilihat pada dataset, nilai minimum longitude lebih besar dari -90º dan nilai maksimumnya lebih kecil dari 90º, sedangkan

latitude memiliki nilai minimum dan maksimum di rentang -180° hingga 180°. Dapat diasumsikan bahwa kolom longitude dan latitude terbalik.

- Kolom siap_pindah seharusnya hanya memiliki dua kategori, bukan sembilan.
- Kolom dijual_kembali seharusnya hanya memiliki dua kategori, bukan lima.
- Kolom kaki_persegi seharusnya bertipe float, bukan object. Namun, ketika dicoba untuk dikonversi, terdapat error yang menunjukkan bahwa terdapat data yang janggal pada kolom kaki_persegi seperti ini:

```
df["kaki_persegi"] = df["kaki_persegi"].astype(float)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-12a7a78a4775> in <module>
----> 1 df["kaki_persegi"] = df["kaki_persegi"].astype(float)

                                  6 frames
/usr/local/lib/python3.7/dist-packages/pandas/core/dtypes/cast.py in astype_nansafe(arr, dtype, copy, skipna)
   1199     if copy or is_object_dtype(arr.dtype) or is_object_dtype(dtype):
   1200         # Explicit copy, or required since NumPy can't view from / to object.
-> 1201         return arr.astype(dtype, copy=True)
   1202
   1203     return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: 'salah'

SEARCH STACK OVERFLOW
```

Oleh karena itu, kolom kaki_persegi dicek secara terpisah dan menghasilkan *output* seperti berikut:

```
kaki_persegi_str = df[df['kaki_persegi'].astype(str).str.contains('[a-zA-Z]')]["kaki_persegi"]
print(kaki_persegi_str.unique())
```

```
[nan 'salah' 'awa' 'benar' 'asik' 'testes' 'a']
```

```
kaki_persegi_1 = df[df['kaki_persegi'].astype(str).str.contains('1-1-')]["kaki_persegi"]
print(kaki_persegi_1.unique())
```

```
['1-1-1-']
```

Dapat dilihat, kolom kaki_persegi memiliki nilai yang tidak seharusnya.

Berdasarkan temuan kejanggalan pada dataset, maka perlu dilakukan validasi data. Berikut langkah yang dilakukan untuk melakukan validasi terhadap data:

- Menukar kolom latitude dengan longitude dengan asumsi bahwa nama kolom tersebut terbalik:

```
# Menukar nama kolom longitude dan latitude
df['temp'] = df['longitude']
df['longitude'] = df['latitude']
df['latitude'] = df['temp']
df.describe()
```

	sedang_pembangunan	disetujui_pemerintah	total_ruangan	longitude	latitude	harga	temp
count	27476.000000	29451.000000	28071.000000	26008.000000	29451.000000	29451.000000	29451.000000
mean	0.179684	0.317918	2.392220	76.820157	21.300255	142.898746	21.300255
std	0.383931	0.465675	0.882229	10.697823	6.205306	656.880713	6.205306
min	0.000000	0.000000	1.000000	-121.761248	-37.713008	0.250000	-37.713008
25%	0.000000	0.000000	2.000000	73.798100	18.452663	38.000000	18.452663
50%	0.000000	0.000000	2.000000	77.326330	20.750000	62.000000	20.750000
75%	0.000000	1.000000	3.000000	77.943705	26.900926	100.000000	26.900926
max	1.000000	1.000000	20.000000	152.962676	59.912884	30000.000000	59.912884

- Me-*replace* data yang tidak sesuai dengan modus untuk data kategorik dan median untuk data continuous. Median digunakan karena kolom kaki_persegi memiliki cukup banyak outlier, dengan jumlah outlier sebanyak 1491 data.

```
# Check outlier data for kaki_persegi
kpm = df[df["kaki_persegi"].notnull()]['kaki_persegi']
kpm = pd.DataFrame(df[~df['kaki_persegi'].astype(str).str.contains('[a-zA-Z]')]["kaki_persegi"])
kpm = pd.DataFrame(kpm[~kpm['kaki_persegi'].astype(str).str.contains('1-1-')]["kaki_persegi"])
kpm = pd.DataFrame(kpm[kpm.notnull()]["kaki_persegi"]).astype(float)

Q1 = kpm.quantile(0.25)
Q3 = kpm.quantile(0.75)
IQR = Q3 - Q1
jumlah_outlier = kpm[((kpm < (Q1 - 1.5 * IQR)) |(kpm > (Q3 + 1.5 * IQR))).any(axis=1)].count()
print(jumlah_outlier)
```

```
kaki_persegi    1491
dtype: int64
```

```
# clean this data
# Transform data that
df.loc[~df['siap_pindah'].isin(['0','1']), 'siap_pindah'] = df['siap_pindah'].mode()[0]
df.loc[~df['dijual_kembali'].isin(['0','1']), 'dijual_kembali'] = df['dijual_kembali'].mode()[0]
kpm_median = kpm.median()
df.replace({'kaki_persegi': {"testes": kpm_median, 'a': kpm_median, 'awa':kpm_median, '1-1-1-':kpm_median, 'benar':kpm_median, 'salah':kpm_median, 'asik':kpm
```

Berikut hasil dari validasi data yang dilakukan:

```
df.describe(include="object")
```

	diposting_oleh	tipe_a_atau_b	kaki_persegi	siap_pindah	dijual_kembali	alamat
count	29451	29451	27765.000000	29451	29451	27436
unique	3	2	18109.000000	2	2	6648
top	Dealer	tipe_a	1176.470588	1	1	Zirakpur,Chandigarh
freq	18291	29427	1001.000000	24741	27649	473

Dari gambar di atas, dapat dilihat bahwa jumlah kategori untuk siap_pindah dan dijual_kembali sudah sesuai.

3.6. Null Values

Pertama-pertama akan dilakukan pengecekkan untuk *null values*:

```
df.isnull().sum(axis = 0)
```

```
diposting_oleh              0
sedang_pembangunan       1975
disetujui_pemerintah        0
total_ruangan            1380
tipe_a_atau_b               0
kaki_persegi             1686
siap_pindah                 0
dijual_kembali              0
alamat                   2015
longitude                3443
latitude                    0
harga                       0
temp                        0
dtype: int64
```

Terdapat beberapa kolom yang memiliki null value, yaitu sedang_pembangunan, total_ruangan, kaki_persegi, alamat, dan longitude. Untuk data numerik seperti kaki_persegi dan total_ruangan akan diidentifikasi terlebih dahulu apakah memiliki outlier atau tidak. Jika memiliki outlier, maka *null values* akan di-*replace* menggunakan median, sedangkan jika tidak memiliki *outlier*, maka akan di-*replace* menggunakan mean. Lalu untuk kolom sedang_pembangunan, *null values* akan di-*replace* dengan mode karena merupakan data kategorik.

```
# Menghitung Outlier
for col in ['total_ruangan', 'kaki_persegi']:
  a = pd.DataFrame(df[df[col].notnull()][col].astype(float))
  Q1 = a[col].quantile(0.25)
  Q3 = a[col].quantile(0.75)
  IQR = Q3 - Q1
  jumlah_outlier = a[((a[col] < (Q1 - 1.5 * IQR)) |(a[col] > (Q3 + 1.5 * IQR)))][col].count()
  print("jumlah outlier kolom "+col+": "+str(jumlah_outlier))
```

```
jumlah outlier kolom total_ruangan: 271
jumlah outlier kolom kaki_persegi: 1626
```

Karena kolom total_ruangan dan kaki_persegi memiliki *outlier*, maka *null values* akan di-*replace* dengan median dan kolom sedang_pembangunan dengan modus.

```
#menghitung mean untuk nilai pada kolom kaki_persegi dan total_ruangan yang tidak bernilai null
median_kaki_persegi = df["kaki_persegi"].astype(float).median()

median_total_ruangan = round(df["total_ruangan"][df['total_ruangan'].notnull()].astype(int).median())

# Me-replace null value dengan modus untuk data kategorik dan median atau mean untuk data continue
df['sedang_pembangunan'].fillna(df['sedang_pembangunan'].mode()[0], inplace = True)
df['total_ruangan'].fillna(median_total_ruangan, inplace = True)
df['kaki_persegi'].fillna(median_kaki_persegi, inplace = True)
```

Untuk kolom alamat dan longitude, *null values* dapat di-*replace* dengan menggunakan *library* Geocoders modul Nominatim. Baris pada dataset yang memiliki Latitude dan Longitude tidak bernilai null dapat dimanfaatkan untuk menentukan nilai pada kolom alamat yang bernilai null seperti berikut:

```
alamat_null = pd.DataFrame(df[(df['alamat'].isnull()) & (~df['longitude'].isnull()) & (~df['latitude'].isnull())])
```

```
list_null = alamat_null.index.tolist()
for i in list_null:
      locator = Nominatim(user_agent='myGeocoder')
      coordinates = str(df['latitude'][i])+','+str(df['longitude'][i])
      location = locator.reverse(coordinates)
      df['alamat'][i] = location.address
df.isnull().sum(axis = 0)
```

dengan begitu, jumlah *null values* untuk kolom alamat dapat berkurang dari 2015 menjadi 229:

```
diposting_oleh            0
sedang_pembangunan        0
disetujui_pemerintah      0
total_ruangan             0
tipe_a_atau_b             0
kaki_persegi              0
siap_pindah               0
dijual_kembali            0
alamat                  229
longitude              3443
latitude                  0
harga                     0
temp                      0
dtype: int64
```

Sama seperti alamat, baris pada kolom longitude yang memiliki nilai null juga dapat diidentifikasi menggunakan alamat yang sudah ada. Selama alamat tidak bernilai null dan terdapat pada library, maka dapat diubah menjadi longitude dan latitude.

```
longitude_null = pd.DataFrame(df[(~df['alamat'].isnull()) & (df['longitude'].isnull()) & (~df['latitude'].isnull())])
```

```
locator = Nominatim(user_agent='myGeocoder')
for i in longitude_null.index:
   if (df['alamat'][i] != df['alamat'][1638]) and (df['alamat'][i] != df['alamat'][2991]) and (df['alamat'][i] != df['alamat'][7454]) and (
      location = locator.geocode(df['alamat'][i])
      if location != None:
        df['longitude'][i] = location.longitude
df.isnull().sum(axis = 0)
```

Dapat dilihat, jumlah *null value* untuk kolom longitude sudah berkurang dari 3443 menjadi 1232

```
diposting_oleh            0
sedang_pembangunan        0
disetujui_pemerintah      0
total_ruangan             0
tipe_a_atau_b             0
kaki_persegi              0
siap_pindah               0
dijual_kembali            0
alamat                  229
longitude              1232
latitude                  0
harga                     0
temp                      0
dtype: int64
```

Selanjutnya yaitu melakukan drop untuk sisa null value:

```
df.dropna(inplace=True)
df.isnull().sum(axis = 0)
```

```
diposting_oleh          0
sedang_pembangunan      0
disetujui_pemerintah    0
total_ruangan           0
tipe_a_atau_b           0
kaki_persegi            0
siap_pindah             0
dijual_kembali          0
alamat                  0
longitude               0
latitude                0
harga                   0
kategori_harga          0
kategori_luas           0
dtype: int64
```

Pada tahap ini, sudah tidak ada null value di dataset.

3.7. Menyesuaikan tipe data

Dari eksplorasi terhadap data, diketahui terdapat kolom yang tipe datanya tidak sesuai. Berikut daftar kolom dengan tipe data tidak sesuai:

- Kolom siap_pindah seharusnya memiliki tipe data integer dan bukan object karena hanya terdiri dari angka biner yang merupakan bilangan bulat, yaitu dan 1.
- Kolom dijual_kembali seharusnya memiliki tipe data integer dan bukan object karena hanya terdiri dari angka biner yang merupakan bilangan bulat, yaitu dan 1.
- Kolom sedang_pembangunan seharusnya bertipe data integer, bukan float karena hanya terdiri dari angka 0 dan 1, dimana keduanya merupakan bilangan bulat.
- Kolom total_ruangan seharusnya bertipe data integer, bukan float karena jumlah ruangan seharusnya bertipe diskrit.
- Kolom kaki_persegi seharusnya bertipe float, bukan object.

Oleh karena itu, dilakukan konversi tipe data untuk setiap kolom yang telah disebutkan seperti berikut:

```
#Mengubah tipe data untuk setiap kolom yang tidak sesuai
df["sedang_pembangunan"] = df["sedang_pembangunan"].astype(int)
df["total_ruangan"] = df["total_ruangan"].astype(int)
df["siap_pindah"] = df["siap_pindah"].astype(int)
df["dijual_kembali"] = df["dijual_kembali"].astype(int)
df["kaki_persegi"] = df["kaki_persegi"].astype(float)
```

Berikut tipe data untuk setiap kolom setelah dilakukan konversi:

```
print(df.dtypes)
```

```
diposting_oleh          object
sedang_pembangunan       int64
disetujui_pemerintah     int64
total_ruangan            int64
tipe_a_atau_b           object
kaki_persegi           float64
siap_pindah              int64
dijual_kembali           int64
alamat                  object
longitude              float64
latitude               float64
harga                  float64
temp                   float64
dtype: object
```

3.8. Encoding Data

Agar data lebih mudah diproses, perlu dilakukan encoding terhadap data. Pada eksplorasi data, didapatkan dua kolom yang seharusnya dapat dilakukan encoding, yaitu kolom diposting_oleh dan kolom tipe_a_atau_b. Encoding ini akan dilakukan menggunakan modul LabelEncoder seperti berikut:

```
# Melakukan encoding untuk kolom tipe_a_atau_b dan kolom diposting_oleh
label_encoder = preprocessing.LabelEncoder()

df['tipe_a_atau_b'] = label_encoder.fit_transform(df['tipe_a_atau_b'])

df['diposting_oleh'] = label_encoder.fit_transform(df['diposting_oleh'])
```

Berikut tampilan lima baris pertama dataframe setelah dilakukan encoding:

```
df.head()
```

	diposting_oleh	sedang_pembangunan	disetujui_pemerintah	total_ruangan	tipe_a_atau_b	kaki_persegi	siap_pindah	dijual_kembali	alamat	longitude
0	2	0	0	2	0	1300.236407	1	1	Ksfc Layout,Bangalore	77.597960
1	1	0	0	2	0	1275.000000	1	1	Vishweshwara Nagar,Mysore	76.644605
2	2	0	0	2	0	933.159722	1	1	Jigani,Bangalore	77.632191
3	2	0	1	2	0	929.921143	1	1	Sector-1 Vaishali,Ghaziabad	77.344500
4	1	1	0	2	0	999.009247	0	1	New Town,Kolkata	88.484911

Keterangan Encoding:

- diposting_oleh: Owner (2), Dealer (1), Builder (0)

- tipe_a_atau_b: tipe_a (0), tipe_b (1)

3.9. Menangani Duplikasi Data

Setelah dilakukan validasi dan menangani *missing values*, selanjutnya yaitu mengecek duplikasi pada data seperti berikut:

```
df.duplicated().sum()
```

```
233
```

Duplikasi data perlu dihilangkan.

```
df.drop_duplicates(inplace=True)
```

Setelah dihilangkan, maka hasilnya menjadi seperti berikut:

```
df.duplicated().sum()
```

```
0
```

3.10. Menangani Outlier

Tahap terakhir yaitu menangani outlier. Namun, kolom yang outliernya harus ditangani hanya kolom total_ruangan, harga, kaki_persegi longitude, dan latitude karena merupakan data bertipe numerik. Berikut proses pengecekkan outlier:

```
# Check outlier data for numeric values
for i in ['kaki_persegi', 'harga', 'total_ruangan','longitude', 'latitude']:
  Q1 = df[i].quantile(0.25)
  Q3 = df[i].quantile(0.75)
  IQR = Q3 - Q1
  print("IQR "+i+": "+str(IQR))
```

```
IQR kaki_persegi: 558.4943878500001
IQR harga: 62.0
IQR total_ruangan: 1.0
IQR longitude: 3.9722309999999936
IQR latitude: 8.463072999999998
```

Data yang kurang dari Q1 dan lebi dari Q3 untuk setiap kolom di atas akan dihapus seperti berikut:

```
# Clear outlier data from dataset
for i in ['kaki_persegi', 'harga', 'total_ruangan', 'longitude', 'latitude']:
  Q1 = df[i].quantile(0.25)
  Q3 = df[i].quantile(0.75)
  IQR = Q3 - Q1
  df = df[~((df[i] < (Q1 - 1.5 * IQR)) |(df[i] > (Q3 + 1.5 * IQR)))]
```

Akibat dari melakukan filter untuk outlier, row data berkurang. Berikut dimensi data setelah penghapusan outlier:

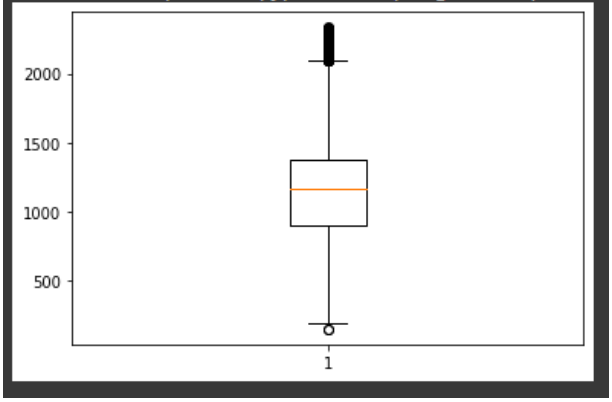
```
df.shape
```

```
(21558, 13)
```

Berikut visualisasi data setelah outlier ditangani:

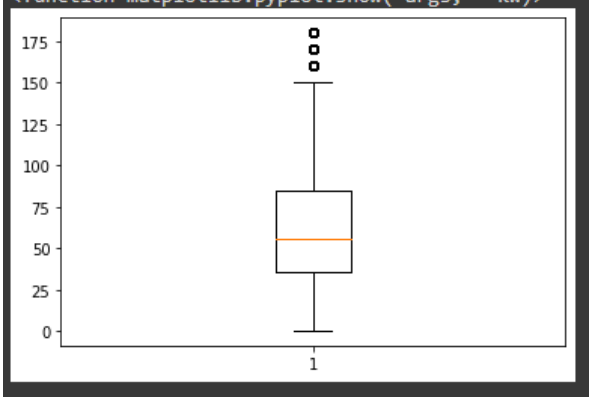
```
fig = plt.figure()
plt.boxplot(df['kaki_persegi'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



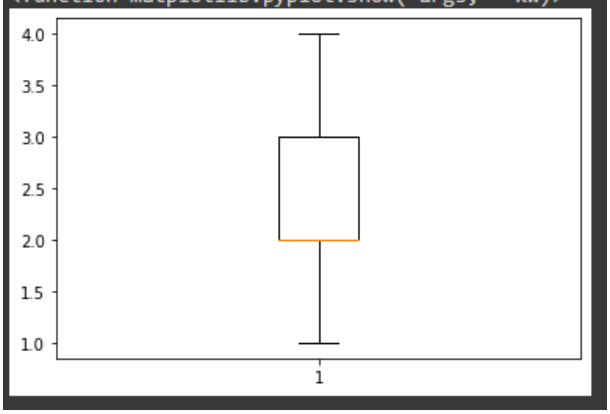
```
fig = plt.figure()
plt.boxplot(df['harga'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



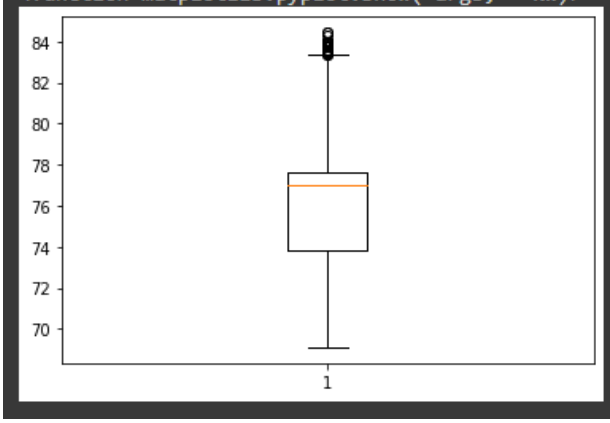

```
fig = plt.figure()
plt.boxplot(df['total_ruangan'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



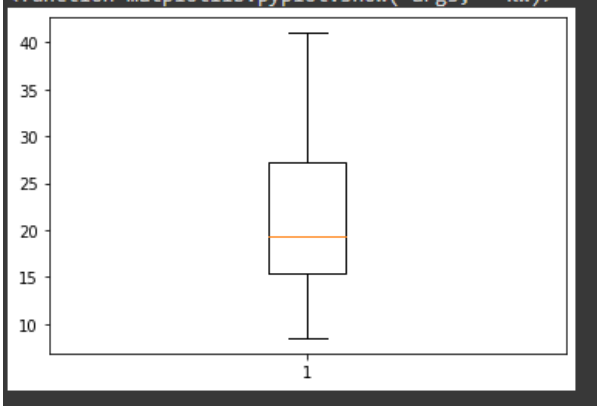

```
fig = plt.figure()
plt.boxplot(df['longitude'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



```
fig = plt.figure()
plt.boxplot(df['latitude'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



3.11. Kategorisasi Data

Data yang bersifat numerik continues dapat dikategorikan. Pada kasus ini, yang akan dikategorikan adalah kolom harga dan kaki_persegi. Kategorisasi tersebut akan disimpan pada kolom kategori_harga untuk harga dan kategori_luas untuk kaki_persegi. Berikut kategorisasi pada kedua kolom:

```
#Kategorisasi Harga
def categorizing_data_harga(cell):
    if cell > 150:
        return ">150"
    elif cell >= 100:
        return "100-150"
    elif cell >= 50:
        return "50-100"
    else:
        return "<50"

df['kategori_harga'] = df.apply(lambda cell: categorizing_data_harga(cell.harga), axis=1)
```

```
#Kategorisasi Luas
def categorizing_data_luas(cell):
    if cell > 2000:
        return ">2000"
    elif cell >= 1500:
        return "1500-2000"
    elif cell >= 1000:
        return "1000-1500"
    elif cell >= 500:
        return "500-1000"
    else:
        return "<500"

df['kategori_luas'] = df.apply(lambda cell: categorizing_data_luas(cell.kaki_persegi), axis=1)
```

Hasil dari kategorisasi:

kategori_harga	kategori_luas
50-100	1000-1500
50-100	1000-1500
<50	500-1000
50-100	500-1000
50-100	500-1000

3.12. Finalisasi dan Mengunduh Dataset

Terdapat kolom temp yang digunakan untuk menukar kolom dan belum dihapus. Oleh karena itu, kolom temp akan dihapus seperti berikut:

```
df.drop(columns="temp", inplace=True)
```

Berikut tampilan data yang sudah dilakukan *preparation*:

```
# get view of data after cleansing
df.head()
```

	diposting_oleh	sedang_pembangunan	disetujui_pemerintah	total_ruangan	tipe_a_atau_b	kaki_persegi
0	2	0	0	2	0	1300.236407
1	1	0	0	2	0	1275.000000
2	2	0	0	2	0	933.159722
3	2	0	1	2	0	929.921143
4	1	1	0	2	0	999.009247

siap_pindah	dijual_kembali	alamat	longitude	latitude	harga	kategori_harga	kategori_luas
1	1	Ksfc Layout,Bangalore	77.597960	12.969910	55.0	50-100	1000-1500
1	1	Vishweshwara Nagar,Mysore	76.644605	12.274538	51.0	50-100	1000-1500
1	1	Jigani,Bangalore	77.632191	12.778033	43.0	<50	500-1000
1	1	Sector-1 Vaishali,Ghaziabad	77.344500	28.642300	62.5	50-100	500-1000
0	1	New Town,Kolkata	88.484911	22.592200	60.5	50-100	500-1000

Pengunduhan:

```
from google.colab import files
df.to_csv('Tugas4_2006531825_AmiraHusnaNurAdilah.csv', index = None, header = True)
files.download('Tugas4_2006531825_AmiraHusnaNurAdilah.csv')
```

4. Hasil dan Temuan

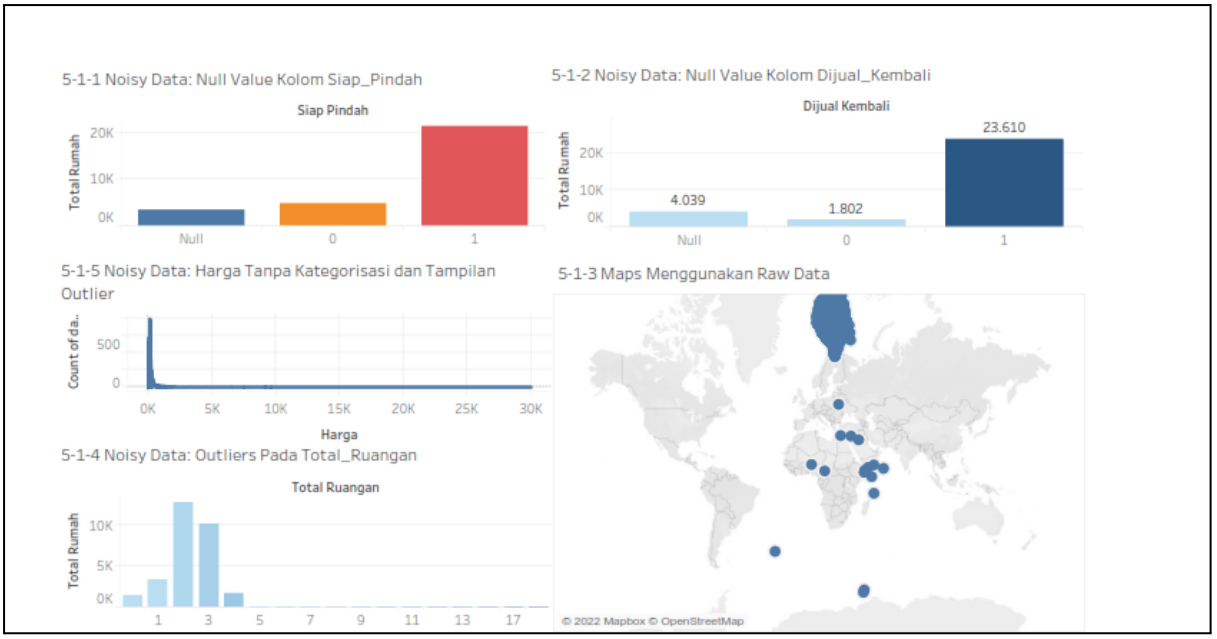
Berdasarkan proses *data preparation* yang telah dilakukan, diketahui bahwa pada awalnya dataset memiliki banyak noisy data seperti input yang tidak sesuai, *null values*, *outliers*, nama kolom yang terbalik, dan kesalahan tipe data. Untuk input yang tidak sesuai akan diganti dengan modus untuk kategorik dan median untuk numerik. Lalu untuk *null values* akan diganti dengan modus untuk kolom sedang_pembangunan, median untuk kaki_persegi dan total_ruangan, serta mengubah *null values* menggunakan Geopandas untuk kolom longitude dan alamat. Selain itu, kolom numerik seperti harga, kaki_persegi, dan total_ruangan akan difilter untuk menghasilkan data yang lebih bersih. Kesalahan input dan kolom yang terbalik juga diperbaiki agar tidak terjadi bias pada data. Tipe data juga perlu diubah agar dapat sesuai dengan deskripsi dataset. Hal di atas dilakukan untuk memastikan bahwa data yang digunakan untuk memudahkan pengolahan data dan analisis serta menghasilkan keluaran yang lebih akurat.

5. Visual Menggunakan Tableau

5.1. Menampilkan Noisy Data

Pada bagian ini akan ditampilkan *dashboard* berisi kumpulan visualisasi data menggunakan *raw data* untuk menunjukan *noisy data* dari *raw data*. Sheet yang

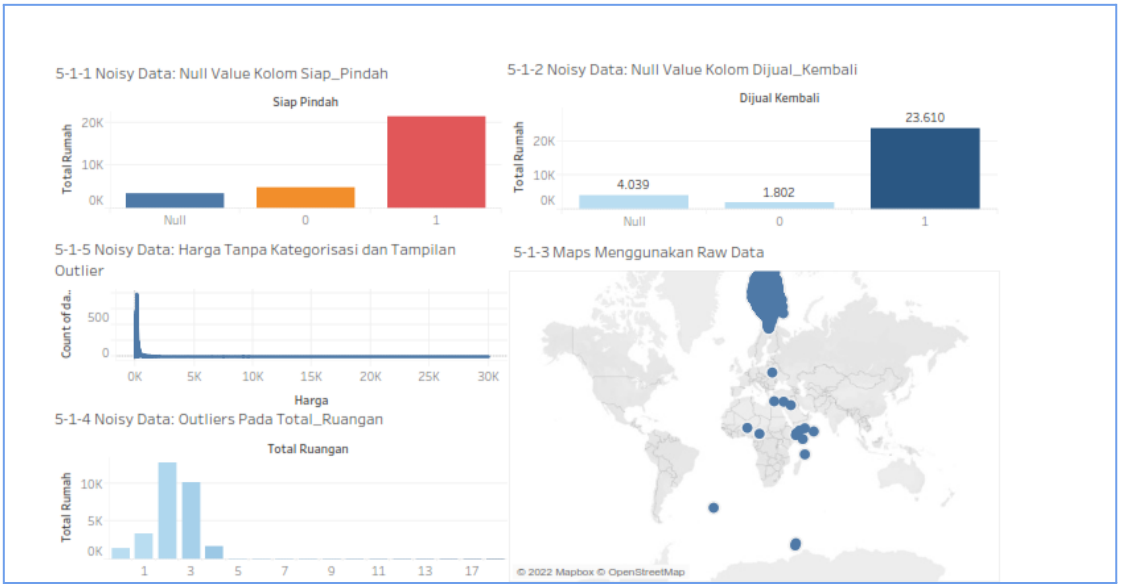
digunakan bernama 5-1-1, 5-1-2, 5-1-3, dan 5-1-4 dengan nama sheet *dashboard* yaitu 5-1. Berikut tampilan dashboard menggunakan raw data:



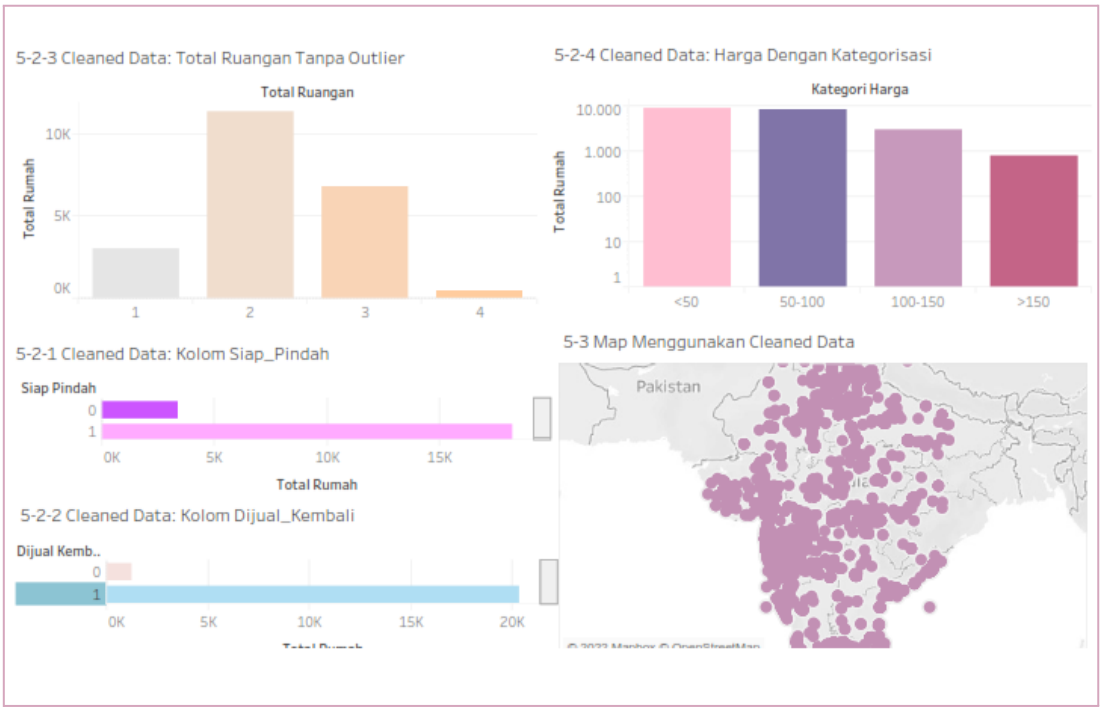
Dari visualisasi 5-1-1 dan 5-1-2 menunjukkan bahwa dapat sebuah kolom tabel memiliki null values. Untuk visualisasi 5-1-4 dan 5-1-5 menunjukkan bahwa data masih memiliki banyak outlier dan terlihat tidak bagus untuk dilihat. Selain itu, pada visualisasi 5-1-3, yaitu berupa maps, terlihat koordinat longitude dan latitude yang janggal dan aneh.

5.2. Membandingkan Raw Data Dengan Cleaned Data

Dashboard berisi visualisasi menggunakan raw data:



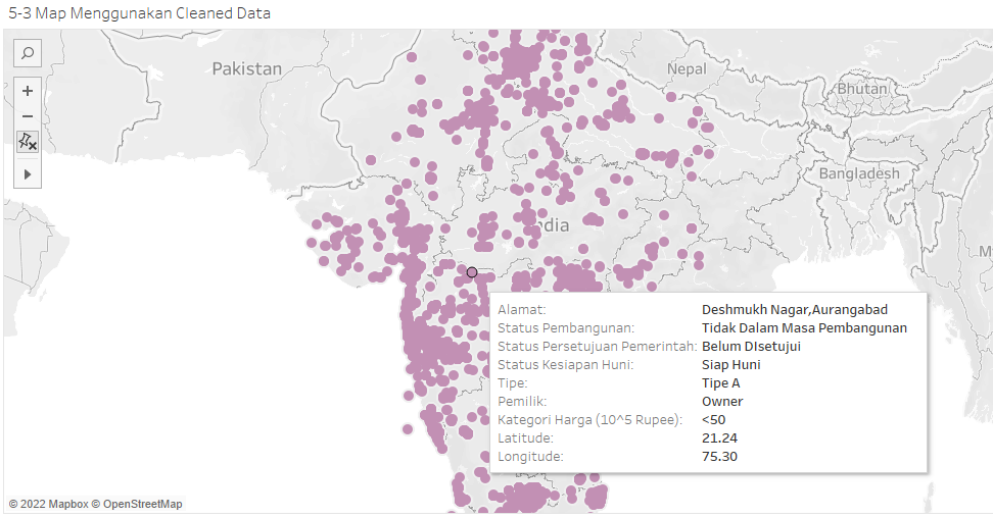
Dashboard berisi menggunakan cleaned data:



Dashboard untuk cleaned data merupakan perpaduan dari visualisasi 5-2-1, 5-2-2, 5-2-3, 5-2-4, dan 5-3. Dari kedua dashboard, dapat dibandingkan bahwa map dengan data longitude dan latitude yang sudah dibersihkan akan terpusat di India, sedangkan pada raw data cenderung ada outlier dan tidak terpusat dengan benar. Dapat dilihat juga data harga yang sudah dikategorisasi lebih teratur dan mudah dimengerti dibandingkan dengan yang belum dilakukan kategorisasi. Kolom siap_pindah dan dijual_kembali juga sudah tidak memiliki null value yang dapat mengganggu visualisasi dan analisis.

5.3. Maps

Berikut maps penjualan rumah di India menggunakan Tableau dan *cleaned data*:



Dari maps dapat diketahui alamat rumah, status pembangunan, status persetujuan dari pemerintah, status kesiapan huni, tipe, pemilik, kategori harga, latitude, dan longitude.

6. Kesimpulan

Sebagian besar *raw data* memiliki *noisy data* yang dapat mengganggu pengolahan data dan analisis. Oleh karena itu, sebelum digunakan, *raw data* harus melewati tahap *data preparation* agar dapat dipersiapkan terlebih dahulu.

Pada laporan ini, penulis menemukan banyak *noisy data* dari dataset dataset_harga_rumah.csv seperti *null values*, kesalahan input, kesalahan nama kolom, dan outliers sehingga penulis melakukan *data preparation* dengan proses seperti eksplorasi data, validasi data, *null values handling*, *duplicate handling*, penyesuaian tipe data, encoding data, kategorisasi, hingga *outliers handling*.

Hasil yang didapat dari data preparation tersebut diantaranya yaitu berkurangnya outlier, hilangnya null values, tidak adanya duplikasi data, serta visualisasi yang lebih baik dari raw data.

Dari hasil yang didapatkan, dapat disimpulkan bahwa *data preparation* berperan penting dalam proses analisis data.

7. Referensi

What is Data Preparation - Data Preparation Explained - AWS. (2022). Amazon Web Services, Inc. https://aws.amazon.com/what-is/data-preparation/

Tamboli, N. (2022, July 25). *All You Need To Know About Different Types Of Missing Data Values And How To Handle It*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/

Shakur, A. (2019). Reverse Geocoding in Python. Towards Data Science. https://towardsdatascience.com/reverse-geocoding-in-python-a915acf29eb6

Shakur, A. (2019). Geocode with Python. Towards Data Science. https://towardsdatascience.com/geocode-with-python-161ec1e62b89