TUGAS

SUB-CPMK KE-2

MATA KULIAH DATA MINING (B)

“MELAKUKAN PRE-PROCESSING DATA PADA DATASET DATA PERUMAHAN KOTA MELBOURNE AUSTRALIA”

DISUSUN OLEH:

Reza Putri Angga (22083010006)

DOSEN PENGAMPU:

Trimono, S.Si., M.Si. (21119950908269)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN” JAWA TIMUR

2024

STUDI KASUS DAN PEMBAHASAN

Berdasarkan data perumahan di kota Melbourne, Australia yang dapat di akses dari *link* berikut : https://tinyurl.com/2vwsmpx9. Diketahui bahwa data tersebut belum siap untuk dianalisis karena masih terdapat beberapa "kotoran". Oleh karena itu diperlukan proses *pre-processing* data. *Pre-processing* data merupakan proses persiapan dan penyesuaian model data sebelum dianalisis atau dimasukkan ke dalam suatu model algoritma. Pada penugasan ini, dilakukan beberapa prosedur *pre-processing* data, di antaranya yakni persiapan data, pembesihan data, transformasi data, dan validasi data yang akan dijelaskan lebih lanjut sebagai berikut.

A. PERSIAPAN DATA

Sebelum masuk kedalam proses *pre-processing* data, sebaiknya perlu dilakukan proses persiapan data untuk memastikan isi dari dataset yang akan diproses, dengan beberapa langkah-langkah, di antaranya yakni melakukan *load* dataset awal.

PERSIAPAN DATA

Melakukan Load Dataset Awal

```
#proses load dataset awal
import pandas as pd

data = pd.read_csv("data perumahan kota melbourne.csv")
data
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Distance	Bedroom2	Bathroom	Car	Landsize	BuildingArea	YearBuilt	CouncilAre
0	Abbotsford	85 Turner St	2	h	1480000	S	Biggin	2.5	2	1	1.0	202	NaN	NaN	Yar
1	Abbotsford	25 Bloomburg St	2	h	1035000	S	Biggin	2.5	2	1	0.0	156	79.0	1900.0	Yar
2	Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	2.5	3	2	0.0	134	150.0	1900.0	Yar
3	Abbotsford	40 Federation La	3	h	850000	PI	Biggin	2.5	3	2	1.0	94	NaN	NaN	Yar
4	Abbotsford	55a Park St	4	h	1600000	VB	Nelson	2.5	3	1	2.0	120	142.0	2014.0	Yar
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13575	Wheelers Hill	12 Strada Cr	4	h	1245000	S	Barry	16.7	4	2	2.0	652	NaN	1981.0	Na
13576	Williamstown	77 Merrett Dr	3	h	1031000	SP	Williams	6.8	3	2	2.0	333	133.0	1995.0	Na
13577	Williamstown	83 Power St	3	h	1170000	S	Raine	6.8	3	2	4.0	436	NaN	1997.0	Na
13578	Williamstown	96 Verdon St	4	h	2500000	PI	Sweeney	6.8	4	1	5.0	866	157.0	1920.0	Na
13579	Yarraville	6 Agnes St	4	h	1285000	SP	Village	6.3	4	1	1.0	362	112.0	1920.0	Na

13580 rows × 19 columns

tampilan proses load dataset awal

Dilakukan proses dataset awal perumahan kota Melbourne, Australia yang disimpan dalam dataframe dengan nama variabel "data". Dimana nantinya, pada saat pemrosesan nama variabel ini yang akan digunakan untuk pengambilan nilai dalam dataset tersebut, diperoleh informasi bahwa terdapat 13580 baris dan 19 kolom.

tampilan proses pengubahan (rename) nama kolom

Dikarenakan beberapa nama kolom pada dataset tersebut masih belum memiliki format yang rapi, sehingga dilakukan proses *rename* atau pengubahan nama kolom agar memiliki format nama yang lebih rapi dan memudahkan pemrosesannya dengan kode *script* tersebut memiliki *argument key:value*, sebagai contoh nama kolom *Suburb* akan diubah menjadi *Suburban Area.*

```
#pengecekan tipe data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Suburban Area   13580 non-null  object
 1   Address         13580 non-null  object
 2   Rooms           13580 non-null  int64
 3   Type            13580 non-null  object
 4   Price           13580 non-null  int64
 5   Method          13580 non-null  object
 6   Seller Group    13580 non-null  object
 7   Distance        13580 non-null  float64
 8   Bedroom         13580 non-null  int64
 9   Bathroom        13580 non-null  int64
 10  Car             13518 non-null  float64
 11  Landsize        13580 non-null  int64
 12  Building Area   7130 non-null   float64
 13  Year Built      8205 non-null   float64
 14  Council Area    12211 non-null  object
 15  Latitude        13580 non-null  float64
 16  Longitude       13580 non-null  float64
 17  Region Name     13580 non-null  object
 18  Property Count  13580 non-null  int64
dtypes: float64(6), int64(6), object(7)
memory usage: 2.0+ MB
```

tampilan proses pengecekan tipe data

Kemudian, tahap persiapan data terakhir dilakukan proses pengecekan tipe data untuk memastikan bahwa tipe data di setiap kolom memiliki tipe yang sesuai dengan data aslinya. Agar dapat mempermudah proses *pre-processing* data.

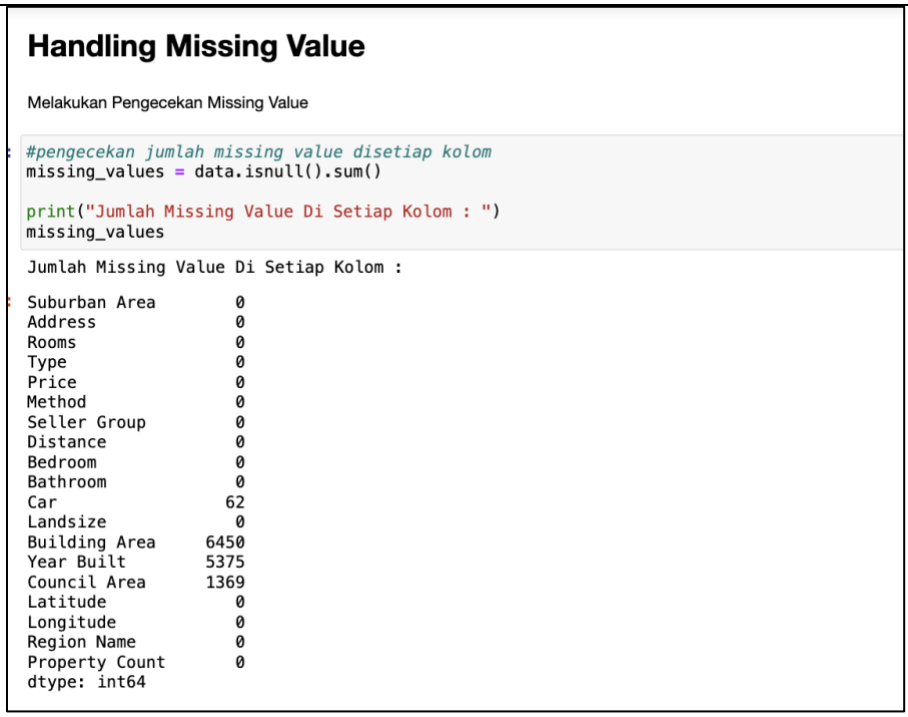
B. PEMBERSIHAN DATA (DATA CLEANING)

Pembersihan data dilakukan untuk mengidentifikasi, menangani, dan menghapus beberapa data yang tidak relevan dari dataset. Bertujuan untuk memastikan bahwa data

akan digunakan untuk analisis memiliki kualitas tinggi dengan integritas data yang terjaga. Dilakukan beberapa tahapan, di antaranya yakni menanggani *missing value*, menanggani *data duplicate*, meanggani *data typo*, dan menanggani *data outlier* yang akan dijelaskan lebih lanjut sebagai berikut.

1. Handling Missing Value (Penangganan Missing Value)

Langkah awal dalam proses pembersihan data adalah identifikasi dan menanggani *missing value* (nilai yang hilang) baik dalam kolom numerik ataupun kategorik. Hal ini bertujuan untuk memastikan adanya integritas dan kualitas data yang akan dianalisis lebih lanjut.



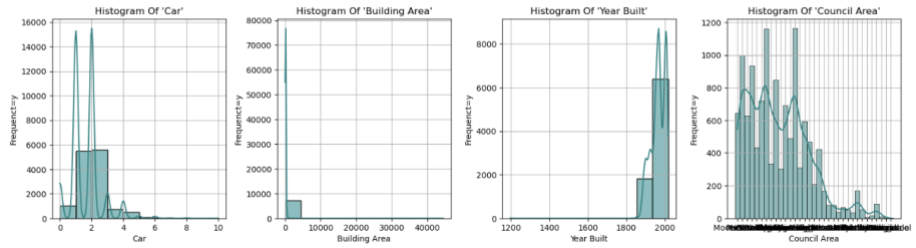
tampilan proses pengecekan missing value

Dengan menggunakan *function isnull()* dan menjumalahkan semua *missing value* yang ditemukan. Diperoleh informasi, bahwa terdapat 4 kolom yang memiliki *missing value* dimana, nilai pada kolom *Car, Building Area*, dan *Year Built* memiliki tipe data numerik. Oleh karena itu, untuk menanggani atau melakukan imputasi (pengisian) nilai pada data yang hilang diperlukan proses pengecekan distribusi terlebih dahulu.

Agar dapat diketahui apakah data tersebut berdistribusi normal atau tidak sehingga dapat diketahui lebih cocok di imputasi menggunakan *mean* (rata-rata) atau *median* (nilai tengah).

Melakukan Penangganan Missing Value

```
#melakukan visualisasi untuk pengecekan apakah kolom car, building area, year huilt, council area
#memiliki distribusi normal atau tidak
#untuk memutuskan apakah lebih cocok ditanggani menggunakan nilai rata-rata atau median
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#pemilihan kolom yang memiliki missing value untuk divisualisasikan
columns_to_visualize = ["Car", "Building Area", "Year Built", "Council Area"]

#visualisasi histogram untuk setiap kolom yang dipilih
columns_to_visualize = ["Car", "Building Area", "Year Built", "Council Area"]

#membuat subplot
plt.figure(figsize=(16, 8))

for i, column in enumerate(columns_to_visualize, start=1):
    plt.subplot(2, 4, i)
    sns.histplot(data[column].dropna(), bins=10, color="teal", kde=True)
    plt.title(f"Histogram Of '{column}'")
    plt.xlabel(column)
    plt.ylabel("Frequenct=y")
    plt.grid()

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



tampilan proses pengecekan distribusi

Berdasarkan visualisasi yang dihasilkan, diketahui bahwa kolom-kolom numerik tersebut tidak memiliki distribusi normal (sebenarnya kolom “Council Area” tidak perlu divisualisasika, karena bertipe kategorik). Sehingga, nilai median lebih cocok digunakan untuk menaggani atau mengisi nilai “*Nan*” dari setiap kolom numerik karena, memiliki nilai distribusi yang cenderung miring.

```
#berdasarkan hasil visualisasi tersebut, diperlihatkan bahwa nilai dalam masing-masing kolom
#yang memiliki missing value, tidak berdistribusi normal
#oleh karena itu, dikarenakan nilai dalam kolom tersebut numerik dan tidak berdistribusi normal
#nilai yang cocok untuk imputasi (menanggani) missing value pada masing-masing kolom adalah "median"
import numpy as np

#nama kolom yang memiliki missing value
column_missing_value = ["Car", "Building Area", "Year Built", "Council Area"]

#mengganti nilai missing value dari setiap kolom numerik dengan menghitung nilai "median" dari data yang ada
for col in column_missing_value:
    if data[col].dtype != "object":  #hanya kolom numerik
        median = data[col].median()
        data[col].fillna(median, inplace=True)

print("Hasil Penangganan (Imputasi) Missing Value :")
data
```

tampilan proses pengisian nilai “Nan” menggunakan median pada tipe data numerik

```python
#jika biasanya, data kategorikal yang memiliki nilai "Nan" diganti dengan nilai "modus"
#pada kali ini digunakan penerapan algoritma klasifikasi "Random Forest"
#untuk memprediksi nilai yang hilang "Nan" pada "Council Area" menggunakan kolom yang berkorelasi
#meliputi, "Suburban Area", "Latitude", "Longitude", "Distance", "Region Name" sebagai fitur
#untuk memprediksi nilai yang hilang di "Council Area"
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

#pemilihan kolom atau fitur yang akan digunakan untuk perhitungan
kolom_calculate = ["Suburban Area", "Distance", "Latitude", "Longitude", "Region Name"]

#pemisahan data menjadi dua bagian, satu "Council Area" yang tidak memiliki "Nan" dan memiliki "Nan"
data_train = data.dropna(subset=["Council Area"]).copy()  #data training
data_predict = data[data["Council Area"].isnull()].copy()  #data testing

#pemisahan kolom atau fitur dan target untuk data yang lengkap
x_train = data_train[kolom_calculate]
y_train = data_train["Council Area"]

#penggunaan ColumnTransformer untuk mengubah data kategorikal menjadi numerik
categorical_features = ["Suburban Area", "Region Name"]
categorical_transformer = Pipeline(steps=[("onehot", OneHotEncoder(handle_unknown="ignore"))])
preprocessor = ColumnTransformer(transformers=[("cat", categorical_transformer, categorical_features)])

#pembuatan pipeline untuk preprocessing dan model
clf = Pipeline(steps=[("preprocessor", preprocessor), ("classifier", RandomForestClassifier())])

#pelatihan pembentukan model
clf.fit(x_train, y_train)

#prediksi untuk data yang memiliki nilai "NaN" pada kolom "Council Area"
x_predict = data_predict[kolom_calculate]
predicted_council_area = clf.predict(x_predict)

#penyimpanan hasil prediksi
data_predict.loc[:, "Council Area"] = predicted_council_area
data = pd.concat([data_train, data_predict], axis=0)
data
```

tampilan proses pengisian nilai "Nan" menggunakan Random Forest pada tipe data kategorik

ls	Type	Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	Latitude	Longitude	Region Name	Property Count
2	h	1480000	S	Biggin	2.5	2	1	1.0	202	126.0	1970.0	Yarra	-37.79960	144.99840	Northern Metropolitan	4019
2	h	1035000	S	Biggin	2.5	2	1	0.0	156	79.0	1900.0	Yarra	-37.80790	144.99340	Northern Metropolitan	4019
3	h	1465000	SP	Biggin	2.5	3	2	0.0	134	150.0	1900.0	Yarra	-37.80930	144.99440	Northern Metropolitan	4019
3	h	850000	PI	Biggin	2.5	3	2	1.0	94	126.0	1970.0	Yarra	-37.79690	144.99690	Northern Metropolitan	4019
4	h	1600000	VB	Nelson	2.5	3	1	2.0	120	142.0	2014.0	Yarra	-37.80720	144.99410	Northern Metropolitan	4019
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	h	1245000	S	Barry	16.7	4	2	2.0	652	126.0	1981.0	Monash	-37.90562	145.16761	South-Eastern Metropolitan	7392
3	h	1031000	SP	Williams	6.8	3	2	2.0	333	133.0	1995.0	Hobsons Bay	-37.85927	144.87904	Western Metropolitan	6380
3	h	1170000	S	Raine	6.8	3	2	4.0	436	126.0	1997.0	Hobsons Bay	-37.85274	144.88738	Western Metropolitan	6380
4	h	2500000	PI	Sweeney	6.8	4	1	5.0	866	157.0	1920.0	Hobsons Bay	-37.85908	144.89299	Western Metropolitan	6380
4	h	1285000	SP	Village	6.3	4	1	1.0	362	112.0	1920.0	Maribyrnong	-37.81188	144.88449	Western Metropolitan	6543

tampilan proses pengisian nilai missing value

Kemudian, untuk menananggani nilai yang hilang pada kolom kategorik *Council Area* dipergunakan penerapan algoritma klasifiksi Random Forest. Penerapan algoritma ini digunakan karena adanya informasi yang cukup tersedia untuk memprediksi diwilayah administratif dimana rumah tersebut berada. Seperti, pada kolom *Latitude* dan *Longitude* utamanya yang berisi informasi geografis. Kemudian ditambahkan kolom lain seperti *Suburban Area, Distance,* dan *Region Name*.

Dengan penerapan langkah algoritma Random Forest yang digunakan adalah pemisahan data *training* dan *testing* dan melakukan pengubahan kategorikal menjadi numerik menggunakan *OneHotEncoder* melalui *ColumnTransformer*. Menggunakan

data nilai *Council Area* yang tidak hilang ditambah dengan fitur atau kolom yang telah di inisasi sebelumnya. Diperoleh hasil imputasi atau penangganan nilai yang hilang *"Nan"* pada kolom *Council Area*.

Perlu diketahui, bahwa sebenarnya ketika menanggani nilai yang hilang pada kolom kategorik dapat digunakan nilai modus. Namun pada data ini, penggunaan nilai modus dianggap tidak relevan dikarenakan adanya informasi mengenai lokasi geografis. Sehingga, ketika digunakan nilai modus maka, nilai yang hilang akan di isi dengan ***Moreland*** tanpa memperhitungkan dan melihat letak geografisnya. Hal tersebut akan membuat data menjadi tidak konsisten.

Oleh karena itu, penerapan algoritma Random Forest dengan perhitungan fitur yang ada dianggap lebih cocok untuk menanggani nilai yang hilang.

```
#pengecekan jumlah missing value disetiap kolom setelah di imputasi (dilakukan penangganan)
missing_values_imputed = data.isnull().sum()

print("Jumlah Missing Value Di Setiap Kolom Setalah Di Tangani (Imputasi) : ")
missing_values_imputed
```

```
Jumlah Missing Value Di Setiap Kolom Setalah Di Tangani (Imputasi) :

Suburban Area     0
Address           0
Rooms             0
Type              0
Price             0
Method            0
Seller Group      0
Distance          0
Bedroom           0
Bathroom          0
Car               0
Landsize          0
Building Area     0
Year Built        0
Council Area      0
Latitude          0
Longitude         0
Region Name       0
Property Count    0
dtype: int64
```

tampilan proses pengecekan missing value setelah di tangani/di imputasi

Setelah melakukan proses pengisian nilai yang hilang dilakukan kembali proses pengecekan *missing value* untuk memastikan bahwa semua nilai yang hilang sudah berhasil di tangani *(imputasi*). Diperoleh informasi bahwa *Car*, *Building Area*, *Year Built*, dan *Council Area* yang sebelumnya memiliki nilai yang hilang, kini telah terisi lengkap.

2. Handling Data Duplicate (Penangganan Data Duplikat)

Diperlukan proses penangganan data duplikat untuk memastikan adanya integritas data dan mencegah penyimpangan dalam analisis data lebih lanjut.

Handling Data Duplicate

Melakukan Pengecekan Data Duplikat

```
#pengecekan data duplikat untuk menjaga konsistensi data
duplicate_rows = data[data.duplicated(keep=False)]

if not duplicate_rows.empty:
    print("Terdapat Data Duplikat Dalam Dataset.")
else:
    print("Tidak Terdapat Data Duplikat Dalam Dataset.")
```

```
Terdapat Data Duplikat Dalam Dataset.
```

```
#penampilan jumlah baris dan data duplikat
print("Jumlah Data Duplikat : ", len(duplicate_rows))
print("Data Duplikat : ")
duplicate_rows
```

```
Jumlah Data Duplikat :  36
Data Duplikat :
```

	Suburban Area	Address	Rooms	Type	Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	
1306	Brunswick	8/2 Pottery Ct	3	t	641000	SP	RW	5.2	3	2	1.0	0	112.0	1900.0	Moreland	-
1344	Brunswick	8/2 Pottery Ct	3	t	641000	SP	RW	5.2	3	2	1.0	0	112.0	1900.0	Moreland	-
1574	Camberwell	3/220 Warrigal Rd	2	u	435000	SP	LITTLE	7.8	2	1	1.0	896	77.0	1960.0	Boroondara	-
1584	Camberwell	3/220 Warrigal Rd	2	u	435000	SP	LITTLE	7.8	2	1	1.0	896	77.0	1960.0	Boroondara	-

tampilan proses penampilan data duplikat

Dilakukan proses pengecekan data duplikat disetiap baris. Kemudian ditampilkan bahwa terdapat sebanyak 36 baris yang memiliki data duplikat. Untuk mengatasi bias yang tidak diinginkan dalam proses asnalisis, dapat digunakan proses penghapusan data duplikat dengan tetap mempertahankan baris pertamanya,

Melakukan Penangganan Data Duplikat

```
#dilakukan penangganan untuk data duplikat dengan menghapus data duplikat
#namun, masih tetap mempertahankan satu baris diawal, bertujuan untuk menjaga konsistensi data
data.drop_duplicates(keep="first", inplace=True)

print("Data Setelah Penghapusan Data Duplikat : ")
data
```

```
Data Setelah Penghapusan Data Duplikat :
```

	Suburban Area	Address	Rooms	Type	Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	Lati
0	Abbotsford	85 Turner St	2	h	1480000	S	Biggin	2.5	2	1	1.0	202	126.0	1970.0	Yarra	-37.7
1	Abbotsford	25 Bloomburg St	2	h	1035000	S	Biggin	2.5	2	1	0.0	156	79.0	1900.0	Yarra	-37.8
2	Abbotsford	5 Charles St	3	h	1465000	SP	Biggin	2.5	3	2	0.0	134	150.0	1900.0	Yarra	-37.8
3	Abbotsford	40 Federation La	3	h	850000	PI	Biggin	2.5	3	2	1.0	94	126.0	1970.0	Yarra	-37.7
4	Abbotsford	55a Park St	4	h	1600000	VB	Nelson	2.5	3	1	2.0	120	142.0	2014.0	Yarra	-37.8

tampilan proses penghapusan data duplikat

Sehingga, diperoleh informasi dataset yang awalnya terdapat 13580 baris kemudian dilakukan pengecekan data duplikat dan terdapat 36 baris data duplikat. Dengan tetap mempertahankan baris pertama, sehingga 36 : 2 = 18, maka terdapat 18 baris yang akan

dihapus. Sehingga, saat ini terdapat data sebesar 13580 – 18 = 13562 baris dan 19 kolom.

3. Handling Data Typo (Penangganan Data Salah Ketik)

Dikarenakan terdapat beberapa kolom yang memiliki tipe data ketegorik, diperlukan pengecekan, identifikasi, dan memperbaiki kesalahahan pengetikan dalam dataset.

Handling Data Typo

```
#mengecek nilai unik di setiap kolom dengan tipe data objek
for column in data.columns:
    if data[column].dtype == 'object':
        unique_values = data[column].unique()
        print(f"Nilai Unik Dari Kolom Kategorik '{column}':")
        print(unique_values)
        print()
```

```
Nilai Unik Dari Kolom Kategorik 'Suburban Area':
['Abbotsford' 'Airport West' 'Albert Park' 'Alphington' 'Altona'
 'Altona North' 'Armadale' 'Ascot Vale' 'Ashburton' 'Ashwood'
 'Avondale Heights' 'Balaclava' 'Balwyn' 'Balwyn North' 'Bentleigh'
 'Bentleigh East' 'Box Hill' 'Braybrook' 'Brighton' 'Brighton East'
 'Brunswick' 'Brunswick West' 'Bulleen' 'Burwood' 'Camberwell'
 'Canterbury' 'Carlton North' 'Carnegie' 'Caulfield' 'Caulfield North'
 'Caulfield South' 'Chadstone' 'Clifton Hill' 'Coburg' 'Coburg North'
 'Collingwood' 'Doncaster' 'Eaglemont' 'Elsternwick' 'Elwood' 'Essendon'
 'Essendon North' 'Fairfield' 'Fitzroy' 'Fitzroy North' 'Flemington'
 'Footscray' 'Glen Iris' 'Glenroy' 'Gowanbrae' 'Hadfield' 'Hampton'
 'Hampton East' 'Hawthorn' 'Heidelberg Heights' 'Heidelberg West'
 'Hughesdale' 'Ivanhoe' 'Kealba' 'Keilor East' 'Kensington' 'Kew'
 'Kew East' 'Kooyong' 'Maidstone' 'Malvern' 'Malvern East' 'Maribyrnong'
 'Melbourne' 'Middle Park' 'Mont Albert' 'Moonee Ponds' 'Moorabbin'
 'Newport' 'Niddrie' 'North Melbourne' 'Northcote' 'Oak Park'
 'Oakleigh South' 'Parkville' 'Pascoe Vale' 'Port Melbourne' 'Prahran'
 'Preston' 'Reservoir' 'Richmond' 'Rosanna' 'Seddon' 'South Melbourne'
 'South Yarra' 'Southbank' 'Spotswood' 'St Kilda' 'Strathmore' 'Sunshine'
```

tampilan proses pengecekan nilai unik disetiap kolom

Dapat dipergunakan cara dengan mengecek nilai unik disetiap kolom yang memiliki tipe kategorik dan membacanya secara satu persatu. Diketahui, bahwa tidak terdapat kesalahan pengetikan (*typo*) disetiap kolom kategorik. Ataupun, agar lebih pasti dapat digunakan algoritma *Levenshtein* untuk mengecek ada atau tidaknya kesalahan pengetikan.

```
#dilakukan pengecekan kesalahan pengetikan "typo"
#menggunakan library "Levenshtein" untuk implementasi algoritma
import Levenshtein

#pengecekan nilai untuk di setiap kolom kategorik
Valid_suburban_areas = data['Suburban Area'].unique()
Valid_addresss = data['Address'].unique()
Valid_types = data['Type'].unique()
Valid_methods = data['Method'].unique()
Valid_seller_groups = data['Seller Group'].unique()
Valid_council_areas = data['Council Area'].unique()
Valid_region_names = data['Region Name'].unique()

#define function pengecekan pengetikan suatu kolom
def detect_typo_in_column(value, valid_values):
    min_distance = float('inf')
    closest_value = None

    for valid_value in valid_values:
        distance = Levenshtein.distance(str(value), str(valid_value))
        if distance < min_distance:
            min_distance = distance
            closest_value = valid_value

    return closest_value, min_distance

#kolom untuk menunjukkan kemungkinan kesalahan ketik
columns_to_check = ['Suburban Area', 'Address', 'Type', 'Method', 'Seller Group', 'Council Area', 'Region Name']
for column in columns_to_check:
    valid_column_name = f"Valid_{column.lower().replace(' ', '_')}s"
    data[f"Closest_{column}"], data[f"Levenshtein_Distance_{column}"] = zip(*data[column].apply(lambda x, valid_valu
                                                                                                detect_typo_in_colu

#menampilkan hasil untuk baris pertama dalam dataset
for column in columns_to_check:
    print(f'Kolom {column} :', data.loc[0, column])
    print(f'Closest {column} :', data.loc[0, f'Closest_{column}'])
    print(f'Levenshtein Distance {column} : ', data.loc[0, f'Levenshtein_Distance_{column}'])
    print()
```

```
Kolom Suburban Area : Abbotsford
Closest Suburban Area : Abbotsford
Levenshtein Distance Suburban Area :  0

Kolom Address : 85 Turner St
Closest Address : 85 Turner St
Levenshtein Distance Address :  0

Kolom Type : h
Closest Type : h
Levenshtein Distance Type :  0

Kolom Method : S
Closest Method : S
Levenshtein Distance Method :  0

Kolom Seller Group : Biggin
Closest Seller Group : Biggin
Levenshtein Distance Seller Group :  0

Kolom Council Area : Yarra
Closest Council Area : Yarra
Levenshtein Distance Council Area :  0

Kolom Region Name : Northern Metropolitan
Closest Region Name : Northern Metropolitan
Levenshtein Distance Region Name :  0
```

tampilan proses pengecekan typo menggunakan library levenshtein

Dengan menggunakan sebuah fungsi untuk mendeteksi kesalahan geneticin dan dilakukan itersi untuk mencari nilai terdekat dari nilai yang diberikan. Dapat diperoleh informasi bahwa tidak terdapat adanya kesalahan pengetikan disetiap kolom dengan tipe data kategorik.

4. Handling Data Outlier (Penangganan Outlier Pada Data Numerik)

Dilakukan proses pengecekan, identifikasi, dan penangganan *outlier* pada setiap kolom numerik untuk memastikan integritas dan keakuratan data. *Outlier* merupakan nilai atau titik data yang memiliki perbedaa secara signifikan dari sebagian besar data lainnya dalam sebuah dataset.

Handling Data Outlier - Pada Data Numerik

Identifikasi Data Outlier

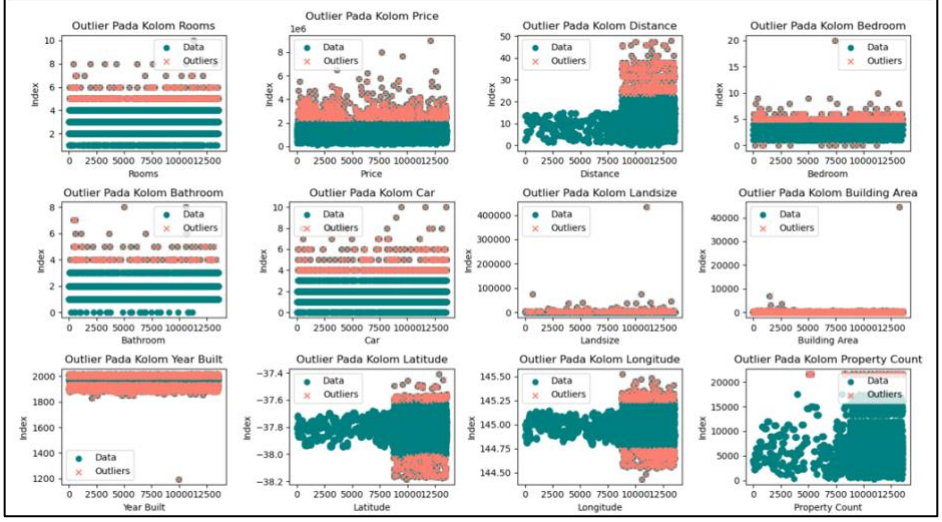
```
#pengecekan outlier pada kolom numerik menggunakan iqr dan thresold sebesar 1,5
kolom_numerik = data.select_dtypes(include=['int64', 'float64'])

#iqr untuk setiap kolom numerik
Q1 = kolom_numerik.quantile(0.25)
Q3 = kolom_numerik.quantile(0.75)
IQR = Q3 - Q1

#nilai threshold
threshold = 1.5

#batas atas dan batas bawah dengan menggunakan threshold
lower_bound = Q1 - (threshold * IQR)
upper_bound = Q3 + (threshold * IQR)

#kondisi nilai dikatakan sebagai outlier
outliers = ((kolom_numerik < lower_bound) | (kolom_numerik > upper_bound))

#nilai outlier di setiap kolom
for column in kolom_numerik.columns:
    outlier_values = kolom_numerik[column][outliers[column]]
    if not outlier_values.empty:
        print("Outlier Pada Kolom", column, ":")
        print(outlier_values)
```

```
Outlier Pada Kolom Rooms :
47       6
55       5
93       5
124      5
142      5
        ..
13468    5
13487    5
13503    6
13560    5
13567    5
Name: Rooms, Length: 682, dtype: int64
Outlier Pada Kolom Price :
80       2850000
92       2615000
97       2575000
102      3010000
103      2800000
```

tampilan proses pengecekan outlier

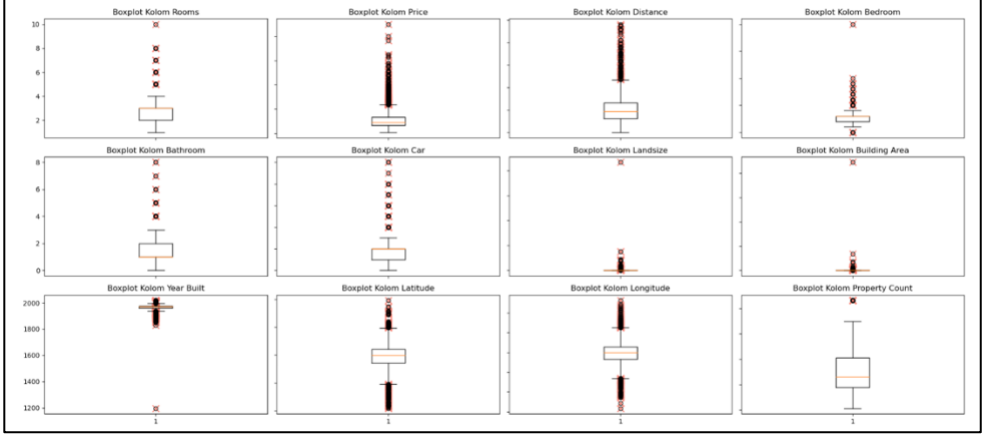
Dilakukan proses pengecekan *outlier* disetiap kolom numerik menggunakan metode *interquartile range* (IQR) dengan *threshold* sebesar 1,5. Dan ditampikan, nilai-nilai *outlier* disetiap kolom dataset. Kemudian, setelah dilakukan pengecekan sebenarnya nilai-nilai tersebut bukanlah *outlier.* Sehingga, dilakukan penangganan pada nilai *outlier* tersebut.

```
#visualisasinya
import matplotlib.pyplot as plt

#menggunakan scatter plot
fig = plt.figure(figsize=(14, 8))
n_columns = len(kolom_numerik.columns)
n_rows = (n_columns - 1) // 4 + 1

for i, column in enumerate(kolom_numerik.columns, 1):
    outlier_values = kolom_numerik[column][outliers[column]]
    if not outlier_values.empty:
        plt.subplot(n_rows, 4, i)
        plt.scatter(range(len(kolom_numerik[column])), kolom_numerik[column], color="teal", label="Data", marker="o"
        plt.scatter(outlier_values.index, outlier_values, color="salmon", label="Outliers", marker="x")
        plt.title("Outlier Pada Kolom {}".format(column))
        plt.xlabel(column)
        plt.ylabel("Index")
        plt.legend()

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



tampilan proses visualisasi outlier menggunakan scatter plot

```python
import matplotlib.pyplot as plt

#visualisasi boxplot untuk setiap kolom numerik
n_columns = len(kolom_numerik.columns)
n_rows = (n_columns - 1) // 4 + 1

fig, axs = plt.subplots(n_rows, 4, figsize=(20, n_rows * 3))

for i, kolom in enumerate(kolom_numerik.columns, 0):
    row = i // 4
    col = i % 4
    axs[row, col].boxplot(kolom_numerik[kolom])
    axs[row, col].set_title('Boxplot Kolom ' + kolom)

    #menambahkan penanda outlier pada boxplot
    outliers = ((kolom_numerik[kolom] < lower_bound[kolom]) | (kolom_numerik[kolom] > upper_bound[kolom]))
    for index, row_data in kolom_numerik[outliers].iterrows():
        axs[row, col].scatter(1, row_data[kolom], color='salmon', marker='x', s=100)

for ax in axs.flat:
    ax.label_outer()

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



tampilan proses visualisasi outlier menggunakan box plot

Dapat dilakukan proses visualisasi baik menggunakan *scatter plot* ataupun *box plot* untuk mengetahui dimana letak *outlier*. Dengan diperlihatkan bahwa warna *salmon* merupakan tanda yang mewakili adanya *outlier*.

Penangganan Data Outlier

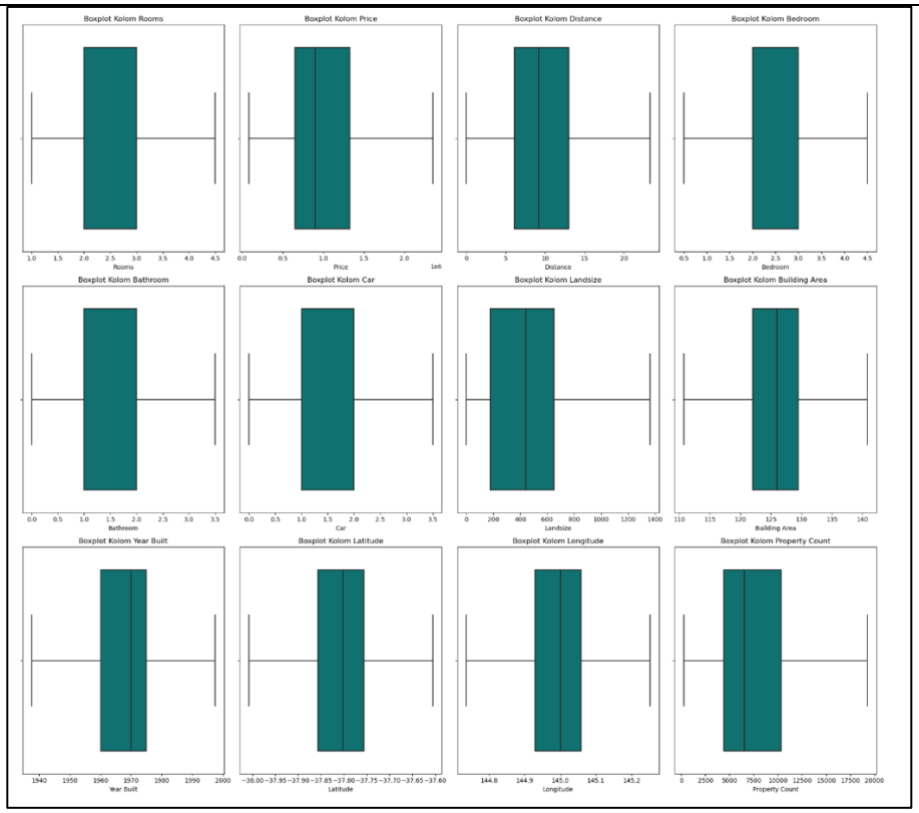
```python
#mengganti outlier dikolom numerik menggunakan lower upper
numeric_cols = data.select_dtypes(include=['float64', 'int64'])

#define function untuk menangani outlier
def handle_outliers(column):

    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    column = column.mask(column < lower_bound, lower_bound)
    column = column.mask(column > upper_bound, upper_bound)
    return column

#pemrosesan setiap kolom numerik
for col in numeric_cols.columns:
    numeric_cols[col] = handle_outliers(numeric_cols[col])
data[numeric_cols.columns] = numeric_cols

data
```

```python
#pengeekan nilai outlier apakah masih ada atau tidak
for col in numeric_cols.columns:
    Q1, Q3 = data[col].quantile([0.25, 0.75])
    IQR = Q3 - Q1
    lower_bound, upper_bound = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
    if any((data[col] < lower_bound) | (data[col] > upper_bound)):
        print(f"Masih Terdapat Outlier Pada Kolom '{col}'")
    else:
        print(f"Tidak Terdapat Outlier Pada Kolom '{col}'")
else:
    print("Tidak Terdapat Outlier Dalam Setiap Data Setelah Transformasi.")
```

```
Tidak Terdapat Outlier Pada Kolom 'Rooms'
Tidak Terdapat Outlier Pada Kolom 'Price'
Tidak Terdapat Outlier Pada Kolom 'Distance'
Tidak Terdapat Outlier Pada Kolom 'Bedroom'
Tidak Terdapat Outlier Pada Kolom 'Bathroom'
Tidak Terdapat Outlier Pada Kolom 'Car'
Tidak Terdapat Outlier Pada Kolom 'Landsize'
Tidak Terdapat Outlier Pada Kolom 'Building Area'
Tidak Terdapat Outlier Pada Kolom 'Year Built'
Tidak Terdapat Outlier Pada Kolom 'Latitude'
Tidak Terdapat Outlier Pada Kolom 'Longitude'
Tidak Terdapat Outlier Pada Kolom 'Property Count'
```

tampilan proses penangganan outlier menggunakan upper lower dan pengecekan

Kemudian, seperti yang telah disampaikan diawal bahwa sebenarnya nilai tersebut bukanlah *outlier* karena masi direntang tertentu. Sehingga, diperlukan proses penangganan *outlier* dengan menggunakan metode IQR. Dan diperoleh informasi, bahwa tidak terdapat lagi kolom yang memiliki *outlier*. Dengan visualisasi sebagai berikut.



tampilan proses visualisasi setelah outlier ditangani

C. TRANSFORMASI DATA

Setelah melakukan proses pembersihan data (*data cleaning*), langkah selanjutnya adalah melakukan proses transformasi data. Bertujuan untuk menyesuaikan skala dari setiap kolom numerik dalam dataset sehingga memiliki rentang nilai yang seragam di rentang antara 0 hingga 1.

TRANSFORMASI DATA

Pada Kolom Numerik

```
#transformasi data pada kolom numerik
#lakukan proses standarisasi dengan menggunakan min max scaling
from sklearn.preprocessing import MinMaxScaler

#hanya untuk kolom numerik
numeric_columns = data.select_dtypes(include=['int', 'float']).columns

#normalisasi menggunakan min-max scaling hanya pada kolom-kolom numerik
scaler = MinMaxScaler()
data[numeric_columns] = scaler.fit_transform(data[numeric_columns])

print("DataFrame Setelah Normalisasi : ")
data
```

```
DataFrame Setelah Normalisasi :
```

Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	Latitude	Longitude	Region Name	Property Count
0.615894	S	Biggin	0.107239	0.375	0.285714	0.285714	0.148311	0.507890	0.541667	Yarra	0.517502	0.508600	Northern Metropolitan	0.198332
0.419426	S	Biggin	0.107239	0.375	0.285714	0.000000	0.114537	0.000000	0.000000	Yarra	0.496846	0.498891	Northern Metropolitan	0.198332
0.609272	SP	Biggin	0.107239	0.625	0.571429	0.000000	0.098385	1.000000	0.000000	Yarra	0.493361	0.500833	Northern Metropolitan	0.198332
0.337748	PI	Biggin	0.107239	0.625	0.571429	0.285714	0.069016	0.507890	0.541667	Yarra	0.524221	0.505687	Northern Metropolitan	0.198332
0.668874	VB	Nelson	0.107239	0.625	0.285714	0.571429	0.088106	1.000000	1.000000	Yarra	0.498588	0.500250	Northern Metropolitan	0.198332
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.512141	S	Barry	0.716354	0.875	0.571429	0.571429	0.478708	0.507890	0.725000	Monash	0.253652	0.837182	South-Eastern Metropolitan	0.375779
0.417660	SP	Williams	0.291689	0.625	0.571429	0.571429	0.244493	0.740449	0.958333	Hobsons Bay	0.369002	0.276820	Western Metropolitan	0.322540
0.479029	S	Raine	0.291689	0.625	0.571429	1.000000	0.320117	0.507890	0.991667	Hobsons Bay	0.385253	0.293015	Western Metropolitan	0.322540
1.000000	PI	Sweeney	0.291689	0.875	0.285714	1.000000	0.635830	1.000000	0.000000	Hobsons Bay	0.369475	0.303908	Western Metropolitan	0.322540
0.529801	SP	Village	0.270241	0.875	0.285714	0.285714	0.265786	0.042774	0.000000	Maribyrnong	0.486941	0.287403	Western Metropolitan	0.331115

tampilan proses transformasi data dengan normalisasi menggunana min max scalling

Untuk memastikan, bahwa setiap kolom numerik dalam dataset memiliki skala yang seragam atau standarisasi setelah dilakukan proses penangganan *outlier*. Dilakukan proses transformasi data menggunakan metode *min max scaling*. Dengan memilih setiap kolom numerik dalam dataset kemudian mengubahnya ke dalam rentang antara 0 hingga 1.

```
#menyimpan data yang telah di cleaning dan normalisasi ke dalam dataset baru
import pandas as pd

#nama file menyimpan data
data_cleaned = data
nama_file_csv = 'data_perumahan_kota_melbourne_pre-processing.csv'

#simpan data ke dalam file CSV
data_cleaned.to_csv(nama_file_csv, index=False)

print(f'Data Hasil Cleaning Dan Normalisasi Telah Di Simpan Dalam File CSV : {nama_file_csv}')
```

```
Data Hasil Cleaning Dan Normalisasi Telah Di Simpan Dalam File CSV : data_perumahan_kota_melbourne_pre-processing.c
sv
```

tampilan proses penyimpanan output kedalam file .csv

`Kemudian, hasil dari proses *pre-processing* data akan disimpan dalam *file .csv* yang berama "data_perumahan_kota_melbourne_pre-processing.csv" yang dapat diakses dalam *link* berikut : https://tinyurl.com/perumahan-melbourne-preproces.

D. VALIDASI DATA

Kemudian untuk memastikan bahwa data yang tersimpan dalam "data_perumahan_kota_melbourne_pre-processing.csv" sudah bersih dan siap dilakukan analisis lebih lanjut. Dilakukan proses validasi data dengan beberapa tahapan pengecekan *missing value*, visualisasi *outlier*, pengecekan tipe data, dan statistika deskriptif.

VALIDASI DATA

```
#validasi data menggunakan pengecekan missing value, visualisasi outlier, tipe data, dan statistik deskriptif
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#dataset setelah preprocessing
data_set = pd.read_csv("data_perumahan_kota_melbourne_pre-processing.csv")

#periksa nilai yang hilang
missing_values = data_set.isnull().sum()
print("Jumlah Nilai Yang Hilang Per-Kolom:")
print(missing_values)

#periksa outliers
numeric_cols = data_set.select_dtypes(include=np.number).columns.tolist()
for col in numeric_cols:
    sns.boxplot(x=data_set[col])
    plt.title(col)
    plt.show()

#tipe data
print("Tipe Data Setiap Kolom : ")
print(data_set.dtypes)


#eksplorasi statistik deskriptif
print("Statistik deskriptif:")
print(data_set.describe())
```

tampilan proses validasi data

Diperoleh hasil bahwa data telah bersih dan siap dianalisis lebih lanjut. Karena, belum diketahui mengenai tujuan pengolahaan analisis data ini, **maka tahapan untuk *pre-processing* berhenti sampai tahapan ini.** Dengan *file* dataset yang telah dilakukan *pre-processing* tersimpan dalam *file* bernama tersebut yang dapat di akses dari *link* berikut : https://tinyurl.com/perumahan-melbourne-preproces .

Jika, ingin dilakukan proses penentuan tujuan dataset di analisis, dapat dilakukan prosedur tambahan untuk melakukan *pre-processing* data pada dataset dengan tujuan untuk menganalisis faktor-faktor yang memengaruhi harga perumahan di kota Melbourne.

ANALISIS LEBIH LANJUT - UNTUK MENGANALISIS FAKTOR-FAKTOR YANG MEMPENGARUHI HARGA PERUMAHAN DI KOTA MELBOURNE

PEMILIHAN FITUR

```
#dilakukan penghapusan kolom "Addres" karena, latitude, longitude, surban area, dll
#sudah bisa memprediksi harga perumahan
data = data.drop("Address", axis=1)
data
```

	Suburban Area	Rooms	Type	Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	Latitu
0	Abbotsford	0.285714	h	0.615894	S	Biggin	0.107239	0.375	0.285714	0.285714	0.148311	0.507890	0.541667	Yarra	0.5175
1	Abbotsford	0.285714	h	0.419426	S	Biggin	0.107239	0.375	0.285714	0.000000	0.114537	0.000000	0.000000	Yarra	0.4968
2	Abbotsford	0.571429	h	0.609272	SP	Biggin	0.107239	0.625	0.571429	0.000000	0.098385	1.000000	0.000000	Yarra	0.4933
3	Abbotsford	0.571429	h	0.337748	PI	Biggin	0.107239	0.625	0.571429	0.285714	0.069016	0.507890	0.541667	Yarra	0.5242
4	Abbotsford	0.857143	h	0.668874	VB	Nelson	0.107239	0.625	0.285714	0.571429	0.088106	1.000000	1.000000	Yarra	0.4985
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13575	Wheelers Hill	0.857143	h	0.512141	S	Barry	0.716354	0.875	0.571429	0.571429	0.478708	0.507890	0.725000	Monash	0.2536
13576	Williamstown	0.571429	h	0.417660	SP	Williams	0.291689	0.625	0.571429	0.571429	0.244493	0.740449	0.958333	Hobsons Bay	0.3690
13577	Williamstown	0.571429	h	0.479029	S	Raine	0.291689	0.625	0.571429	1.000000	0.320117	0.507890	0.991667	Hobsons Bay	0.3852
13578	Williamstown	0.857143	h	1.000000	PI	Sweeney	0.291689	0.875	0.285714	1.000000	0.635830	1.000000	0.000000	Hobsons Bay	0.3694
13579	Yarraville	0.857143	h	0.529801	SP	Village	0.270241	0.875	0.285714	0.285714	0.265786	0.042774	0.000000	Maribyrnong	0.4869

13562 rows × 18 columns

tampilan proses penghapusan kolom alamat karena sudah ada kolom-kolom lain yang bisa mewakilinya

```
#proses pemilihan fitur untuk melakukan encode agar bisa diperhitungkan numerik untuk proses prediksi
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

#inisialisasi nama kolom yang akan diencode
columns_to_encode = ["Suburban Area", "Type", "Method", "Seller Group", "Council Area", "Region Name"]

#label encoding pada kolom yang ditentukan
for column in columns_to_encode:
    data[column] = label_encoder.fit_transform(data[column])
data
```

	Suburban Area	Rooms	Type	Price	Method	Seller Group	Distance	Bedroom	Bathroom	Car	Landsize	Building Area	Year Built	Council Area	Latitude	Longi
0	0	0.285714	0	0.615894	1	23	0.107239	0.375	0.285714	0.285714	0.148311	0.507890	0.541667	31	0.517502	0.50
1	0	0.285714	0	0.419426	1	23	0.107239	0.375	0.285714	0.000000	0.114537	0.000000	0.000000	31	0.496846	0.49
2	0	0.571429	0	0.609272	3	23	0.107239	0.625	0.571429	0.000000	0.098385	1.000000	0.000000	31	0.493361	0.50
3	0	0.571429	0	0.337748	0	23	0.107239	0.625	0.571429	0.285714	0.069016	0.507890	0.541667	31	0.524221	0.50
4	0	0.857143	0	0.668874	4	155	0.107239	0.625	0.285714	0.571429	0.088106	1.000000	1.000000	31	0.498588	0.50
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13575	302	0.857143	0	0.512141	1	16	0.716354	0.875	0.571429	0.571429	0.478708	0.507890	0.725000	20	0.253652	0.83
13576	305	0.571429	0	0.417660	3	251	0.291689	0.625	0.571429	0.571429	0.244493	0.740449	0.958333	10	0.369002	0.27
13577	305	0.571429	0	0.479029	1	194	0.291689	0.625	0.571429	1.000000	0.320117	0.507890	0.991667	10	0.385253	0.29
13578	305	0.857143	0	1.000000	0	222	0.291689	0.875	0.285714	1.000000	0.635830	1.000000	0.000000	10	0.369475	0.30
13579	313	0.857143	0	0.529801	3	239	0.270241	0.875	0.285714	0.285714	0.265786	0.042774	0.000000	16	0.486941	0.28

13562 rows × 18 columns

tampilan proses encode kolom kategorik

Untuk melakukan analisis faktor-faktor yang memengaruhi harga perumahan di kota Melbourne dapat dilakukan beberapa tahapan, yakni menghapus kolom *Adress* karena terdapat kolom-kolom lain yang sudah cukup mewakili untuk melakukan analisis faktor-

faktor yang memengaruhi harga perumahan di kota Melbourne. Kemudian, proses tersebut dilanjutkan dengan melakukan proses label *encoding* pada setiap kolom yang memiliki tipe data kategorik dengan tujuan agar nilai tersebut dapat diubah menjadi nilai numerik yang bisa diperhitungkan.

Dengan menerapkan metode seperti regresi linier sederhana, analisis regresi linier berganda, dan analisis korelasi. Dapat dilakukan analisis faktor-faktor yang memengaruhi harga perumahan di kota Melbourne, Australia.

```
#menyimpan data yang telah di cleaning, normalisasi, dan encoding ke dalam dataset baru
import pandas as pd

#nama file menyimpan data
data_processing = data
nama_file_csv = 'data_perumahan_kota_melbourne_full_pre-processing-analysis.csv'

#simpan data ke dalam file CSV
data_processing.to_csv(nama_file_csv, index=False)

print(f'Data Hasil Pre-Processing Telah Di Simpan Dalam File CSV : {nama_file_csv}')
```

```
Data Hasil Pre-Processing Telah Di Simpan Dalam File CSV : data_perumahan_kota_melbourne_full_pre-processing-analys
is.csv
```

tampilan proses penyimpanan output kedalam file .csv

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

#dataset setelah preprocessing
data_set = pd.read_csv("data_perumahan_kota_melbourne_full_pre-processing-analysis.csv")

#periksa nilai yang hilang
missing_values = data_set.isnull().sum()
print("Jumlah Nilai Yang Hilang Per-Kolom:")
print(missing_values)

#periksa outliers
numeric_cols = data_set.select_dtypes(include=np.number).columns.tolist()
for col in numeric_cols:
    sns.boxplot(x=data_set[col])
    plt.title(col)
    plt.show()

#tipe data
print("Tipe Data Setiap Kolom : ")
print(data_set.dtypes)


#eksplorasi statistik deskriptif
print("Statistik deskriptif:")
print(data_set.describe())


#uji korelasi
correlation_matrix = data_set.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Korelasi antar variabel")
plt.show()
```

tampilan proses validasi data

Kemudian *file* dari proses tersebut akan disimpan dalam *file .csv* dengan nama "data_perumahan_kota_melbourne_full_pre-processing-analiysis.csv" yang dapat diakses di *link* berikut : https://tinyurl.com/perumahan-preprocess-analysis . Dan dilakuakn validasi data untuk mengetahui bahwa hasil data yang telah diproses telah bersih dan siap dianalisis lebih lanjut. Untuk keseluruhan *output* dari proses ini dapat diakses di *file* kode *script* pemrograman dengan *link* : https://tinyurl.com/source-kode-pemrograman .

Lampiran :

Link dataset setelah dilakukan *pre-processing* : https://tinyurl.com/perumahan-melbourne-preproces

Link dataset setelah dilakukan *pre-processing* dengan analisis lebih lanjut : https://tinyurl.com/perumahan-preprocess-analysis

Link kode *script* pemrograman dan *output* lengkapnya : https://tinyurl.com/source-kode-pemrograman