UTS Data Sains dan Analisis
Hasil Analisa

Oleh:
Alifia Mutiara Rahma (1103200025)

Analisa Data dengan Metode Statistik (Materi 5)

File Source Code: UTSDataSains2.ipynb

1. Baris pertama diisi dengan import library. Baris kedua memanggil dataset dan membaca dataset, baru kemudian dibaris ketiga ditampilkan 5 dataset teratas. Dataset yang digunakan adalah Melbourne house prices, yang isinya terdapat harga rumah beserta keterangan identitas rumah yang ada di Melbourne.
   Output:

```
import pandas as pd
import numpy as np
```
[1] ✓ 4.0s Python

```
dataset = "MELBOURNE_HOUSE_PRICES_LESS.csv"
df = pd.read_csv(dataset)
```
[2] ✓ 0.4s Python

```
df.head()
```
[3] ✓ 0.1s Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilArea
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543	7.5	Moonee Valley City Council
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464	10.4	Moonee Valley City Council

2. Pada baris berikutnya, data ditelaah dengan syntax ‘df.dtypes()’ untuk menampilkan tipe data dari setiap kolom dalam dataframe. Serta menggunakan ‘df.describe()’ untuk menampilkan statistik dasar (jumlah data non-null (count), mean, standar deviasi (std), nilai minimum (min), kuartil pertama (25%), kuartil kedua atau nilai median (50%), kuartil ketiga (75%), dan nilai maksimum (max)) dari setiap kolom dalam dataframe yang bertipe numerik.
   Ouput:

```
df.dtypes
```
✓ 0.2s

```
Suburb          object
Address         object
Rooms            int64
Type            object
Price          float64
Method          object
SellerG         object
Date            object
Postcode         int64
Regionname      object
Propertycount    int64
Distance       float64
CouncilArea     object
dtype: object
```

```
df.describe()
```
✓ 0.2s

	Rooms	Price	Postcode	Propertycount	Distance
count	63023.000000	4.843300e+04	63023.000000	63023.000000	63023.000000
mean	3.110595	9.978982e+05	3125.673897	7617.728131	12.684829
std	0.957551	5.934989e+05	125.626877	4424.423167	7.592015
min	1.000000	8.500000e+04	3000.000000	39.000000	0.000000
25%	3.000000	6.200000e+05	3056.000000	4380.000000	7.000000
50%	3.000000	8.300000e+05	3107.000000	6795.000000	11.400000
75%	4.000000	1.220000e+06	3163.000000	10412.000000	16.700000
max	31.000000	1.120000e+07	3980.000000	21650.000000	64.100000

3. Metode Statistik ‘df.sum()’ berfungsi untuk menghitung total jumlah dari masing-masing kolom dalam dataframe. Dalam output dibawah, data bertipe object juga ikut dijumlahkan, sementara untuk data yang bertipe numerik seperti Rooms, Price, Postcode, Propertycount, Distance dijumlahkan.
Output:

```
df.sum()
✓ 25.3s

Suburb          AbbotsfordAbbotsfordAbbotsfordAberfeldieAirpor...
Address         49 Lithgow St59A Turner St119B Yarra St68 Vida...
Rooms                                                      196039
Type            hhhhhtuhhhuhhhhhhhhuhhuuhuhthhhhhuthhuhhhhhhhuhh...
Price                                             48331205530.0
Method          SSSSSSSSPPISSSSSSPSNSSSSVBSSSSSSSSSNSPSPPIPIPI...
SellerG         JellisMarshallNelsonBarryNelsonJellisBarryNels...
Date            1/04/20171/04/20171/04/20171/04/20171/04/20171...
Postcode                                                196989346
Regionname      Northern MetropolitanNorthern MetropolitanNort...
Propertycount                                           480092080
Distance                                                799436.0
CouncilArea     Yarra City CouncilYarra City CouncilYarra City...
dtype: object
```

4. Metode statistik ‘df.mean()’ digunakan untuk menghitung nilai rata-rata atau mean dari masing-masing kolom yang bernilai numerik. Dalam source code dibawah, saya menghilangkan terlebih dahulu kolom yang bertipe string agar tidak terjadi eror dalam perhitungan nilai rata-rata. Nilai rata-rata yang didapatkan berasal dari rumus jumlah total data dibagi dengan banyaknya data dalam kolom yang dihitung. Tentu saja apabila kolomnya bertipe string tidak akan bisa dihitung nilainya.
Output:

```
to_drop = ['Suburb',
           'Address',
           'Type',
           'Method',
           'SellerG',
           'Date',
           'CouncilArea',
           'Regionname']

df.drop(to_drop, inplace=True, axis=1)
df.mean()
✓ 0.0s

Rooms               3.110595
Price          997898.241488
Postcode         3125.673897
Propertycount    7617.728131
Distance           12.684829
dtype: float64
```

5. Metode statistik ‘df.median()’ digunakan untuk menunjukan nilai tengah (median) atau nilai kuartil kedua (50%) dari masing-masing kolom yang bertipe numerik. Seperti yang ada di output, beberapa kolom seperti rooms, price, postcode, propertycount, distance yang bertipe numerik ditampilkan nilai tengah dari data kolom yang sudah diurutkan dari yang paling kecil.
Output:

```
df.median()

Rooms               3.0
Price          830000.0
Postcode         3107.0
Propertycount    6795.0
Distance           11.4
dtype: float64
```

6. Metode statistik ‘df.var()’ digunakan untuk menghitung variansi dalam dataframe di setiap kolom yang bertipe numerik. Dalam perhitungan varian diambil dari selisih antara setiap titik data dengan rata-rata, lalu hasilnya dikuadratkan dan dibagi dengan jumlah total titik data. Dan output yang tertampil pun sesuai dengan rumus variansi.
   Output:

```
df.var()
```

```
Rooms            9.169045e-01
Price            3.522410e+11
Postcode         1.578211e+04
Propertycount    1.957552e+07
Distance         5.763870e+01
dtype: float64
```

7. Metode statistik ‘df.quantile(0.75)’ digunakan untuk menghitung kuartil dalam suatu dataframe, karena di inputan df.quantile(0.75) maka untuk menghitung kuartil kuartil ketiga (Q3) dalam tipe data dalam kolom yang bertipe numerik.
   Output:

```
df.quantile(0.75)
```

```
Rooms                4.0
Price          1220000.0
Postcode          3163.0
Propertycount    10412.0
Distance            16.7
Name: 0.75, dtype: float64
```

8. Metode statistik pencilan dengan Tukey’s fences yang pertama ini digunakan untuk menghitung rentang antar kuartil (IQR) dari sebuah dataframe. IQR ialah jarak antara kuartil pertama (Q1) dan kuartil ketiga (Q3) dalam dataframe. Dalam output dibawah ini yang dihitung adalah dataset yang bertipe numerik.
   Output:

```
#Mencari pencilan dengan Tukey's fences (1)
q1 = df.quantile(0.25)
q3 = df.quantile(0.75)
iqr = q3 - q1
iqr
```

```
Rooms              1.0
Price         600000.0
Postcode        107.0
Propertycount  6032.0
Distance          9.7
dtype: float64
```

9. Dalam output dan source code dibawah ini berfungsi untuk menyaring atau memfilter nilai-nilai yang termasuk dalam kategori pencilan (outliers) pada sebuah dataframe ‘df’. Filter ini dibuat dengan menggunakan rumus Tukey's fences. Lalu nilai-nilai yang tidak termasuk dalam kategori pencilan (yang berada di dalam batas bawah dan batas atas) ditandai dengan nilai **True** pada filter, sedangkan nilai-nilai yang termasuk dalam kategori pencilan ditandai dengan nilai **False**. Kemudian hasil dari operasi ~outlier_filter adalah filter yang menunjukkan nilai-nilai mana saja yang tidak termasuk dalam kategori pencilan.
   Output:

```
# filter outliers from dataframe
df_filtered = df[~outlier_filter]

# print filtered dataframe
df_filtered
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilAre
0	Abbotsford	49 Lithgow St	3.0	h	1490000.0	S	Jellis	1/04/2017	3067.0	Northern Metropolitan	4019.0	3.0	Yarra Cit Counc
1	Abbotsford	59A Turner St	3.0	h	1220000.0	S	Marshall	1/04/2017	3067.0	Northern Metropolitan	4019.0	3.0	Yarra Cit Counc
2	Abbotsford	119B Yarra St	3.0	h	1420000.0	S	Nelson	1/04/2017	3067.0	Northern Metropolitan	4019.0	3.0	Yarra Cit Counc
3	Aberfeldie	68 Vida St	3.0	h	1515000.0	S	Barry	1/04/2017	3040.0	Western Metropolitan	1543.0	7.5	Moone Valley Cit Counc
4	Airport West	92 Clydesdale Rd	2.0	h	670000.0	S	Nelson	1/04/2017	3042.0	Western Metropolitan	3464.0	10.4	Moone Valley Cit Counc
...	...	...	...	...	...	...	...	...	...	...	...	...	
63018	Roxburgh Park	3 Carr Pl	3.0	h	566000.0	S	Raine	31/03/2018	3064.0	Northern Metropolitan	5833.0	20.6	Hume Cit Counc
63019	Roxburgh Park	9 Parker Ct	3.0	h	500000.0	S	Raine	31/03/2018	3064.0	Northern Metropolitan	5833.0	20.6	Hume Cit Counc
63020	Roxburgh	5 Parkinson	3.0	h	545000.0	S	Raine	31/03/2018	3064.0	Northern	5833.0	20.6	Hume Cit

Cell 2 of 19

10. Dalam metode statistik dibawah ini tertampil **False**, hal ini dikarenakan tidak ada nilai yang memenuhi kodisi dari outlier_filter. Kondisi outlier_filter adalah hasil operasi dari (df < q1 - 1.5 * iqr_new) | (df > q3 + 1.5 * iqr_new).
    Output:

```
#Mencari pencilan dengan Tukey's fences (2)
#Handle warning
import warnings
warnings.filterwarnings('ignore')

#Outlier filter
df_align, iqr_new = df.align(iqr, axis=1, copy=False, join='outer')
outlier_filter = (df < q1 - 1.5 * iqr_new) | (df > q3 + 1.5 * iqr_new)
outlier_filter
```

	Address	CouncilArea	Date	Distance	Method	Postcode	Price	Propertycount	Regionname	Rooms	SellerG	Suburb	Type
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
63018	False	False	False	False	False	False	False	False	False	False	False	False	False
63019	False	False	False	False	False	False	False	False	False	False	False	False	False
63020	False	False	False	False	False	False	False	False	False	False	False	False	False
63021	False	False	False	False	False	False	False	False	False	False	False	False	False
63022	False	False	False	False	False	False	False	False	False	False	False	False	False

63023 rows × 13 columns

11. Metode pencilan dengan Tukey's fences yang ketiga ini berguna untuk mendeteksi nilai outlier atau nilai yang berada jauh dari rentang nilai yang normal (antara kuartil). Output menunjukan hasil filter yang diurutkan berdasarkan nilai pada kolom 'Price'.
    Ouput:

```
#Mencari pencilan dengan Tukey's fences (3)
df[outlier_filter['Price']] \
    .loc[:, ['Address', 'Price']] \
    .sort_values(by=['Price'], ascending=False)
```

	Address	Price
59166	6 Cole St	11200000.0
61142	35 Bevis St	9000000.0
23841	49 Mangarra Rd	8000000.0
33780	49 Lisson Gr	7650000.0
51592	1 Barnato Gr	7000000.0
...	...	...
57357	2 Hazel St	2125000.0
45487	2 Idinia Ct	2123500.0
23386	69 Beaver St	2123000.0
35054	39 Atheldene Dr	2123000.0
1335	14 Knightsbridge Ct	2121800.0

2272 rows × 2 columns

12. Metode statistic value count() ini menghasilkan frekuensi setiap nilai unik di dalam kolom, yang tertinggi count-nya merupakan modus pada kolom 'Address'. Alamat dengan frekuensi paling tinggi adalah 5 Charles St dan seterusnya. Nilai dengan frekuensi yang sama diurutkan berdasarkan abjad.
    Output:

```
#Value Count()
df['Address'].value_counts()
```

```
5 Charles St         7
3 Donald St          7
14 Moray St          7
57 Bay Rd            7
52 Station St        7
                    ..
1/33 Leila Rd        1
2/9 Davey St         1
1/34 MacGregor St    1
3/3 Rennison St      1
1 Diadem Wy          1
Name: Address, Length: 57754, dtype: int64
```

13. Metode groupby memungkinkan analisa dilakukan secara per kelompok nilai antara kolom 'Address' dan kolom 'Price' seperti yang tertampil dalam output.
    Output:

```
#Analisis dengan groupby
#Misal: rerata harga rumah menurut alamatnya
df.groupby('Address')['Price'].mean()
```

```
Address
1 Abbot Ct                NaN
1 Abercrombie St    2100000.0
1 Aberdeen Ct        570000.0
1 Aberfeldie Wy      680000.0
1 Abraham Dr         655000.0
                        ...
9b Marquis Rd       1435000.0
9b Powys Dr         1420000.0
9b Stewart St       1160000.0
9b Veronica St      1217500.0
9c State St          976000.0
Name: Price, Length: 57754, dtype: float64
```

14. Metode corr() menghasilkan tabel korelasi pearson antar kolom-kolom numerik (Price, Postcode, Propertycount, Distance). Nilai yang didapatkan -1 = korelasi negatif | 0 = tidak ada korelasi linier | +1 = korelasi positif.
    Output:

```
#Korelasi Pearson antara kolom-kolom numerik
df.loc[:,'Price':].corr()
```

	Price	Postcode	Propertycount	Distance
Price	1.000000	0.003112	-0.060769	-0.253668
Postcode	0.003112	1.000000	-0.002557	0.500263
Propertycount	-0.060769	-0.002557	1.000000	0.014050
Distance	-0.253668	0.500263	0.014050	1.000000

Analisa Data dengan Visualisasi, Proses Grouping, dan Anova (Materi 6)

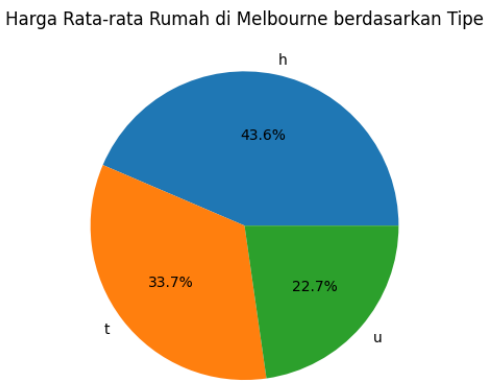
File Source Code: UTSDataSains3.ipynb

1. PieChart

   Visualisasi pertama menggunakan Pie Chart, pertama-tama dataframe di bersihkan dengan menghapus baris yang memiliki nilai kosong pada kolom ‘Price’. Kedua dataframe di grouping berdasarkan nilai pada kolom ‘Type’ dan dihitung nilai rata-rata pada kolom ‘ Price’ lalu grouping ini disimpan dalam dataframe df_grouped. Selanjutnya baru hasil grouping divisualisasikan menggunakan pie chart dengan menggunakan library matplotlib.pyplot. Dalam pie chart ini, data divisualisasikan adalah nilai rata-rata (mean) harga rumah di Melbourne berdasarkan tipe rumah (h,t,u) dalam dataframe.

   Output:

```
#Melakukan Grouping
df_grouped = df.groupby('Type')['Price'].mean()


#Visualisasi dengan Pie chart
plt.pie(df_grouped, labels=df_grouped.index, autopct='%1.1f%%')
plt.title('Harga Rata-rata Rumah di Melbourne berdasarkan Tipe')
plt.show()
```

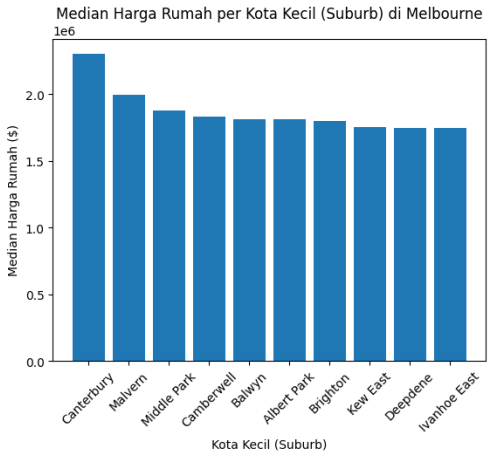


2. Bar Chart

   Visualisasi data ini menggunakan Bar Chart, pertama-tama dataframe di bersihkan dengan menghapus baris yang memiliki nilai kosong pada kolom ‘Price’. Kemudian dataframe di grouping berdasarkan kolom ‘Suburn’ dan dihitung nilai median harga rumah dalam kolom ‘Price’. Lalu data diurutkan berdasarkan nilai median kolom ‘Price’ dan divisualisasikan dengan judul nilai median harga rumah per ‘Suburn’ atau kota kecil.

   Output:

```
#Grouping berdasarkan 'Suburn' dan menghitung median 'Price'.
df_grouped = df.groupby('Suburb')['Price'].median().sort_values(ascending=False)[:10]


#Visualisasi dengan Bar Chart
plt.bar(df_grouped.index, df_grouped.values)
plt.title('Median Harga Rumah per Kota Kecil (Suburb) di Melbourne')
plt.xlabel('Kota Kecil (Suburb)')
plt.ylabel('Median Harga Rumah ($)')
plt.xticks(rotation=45)
plt.show()
```

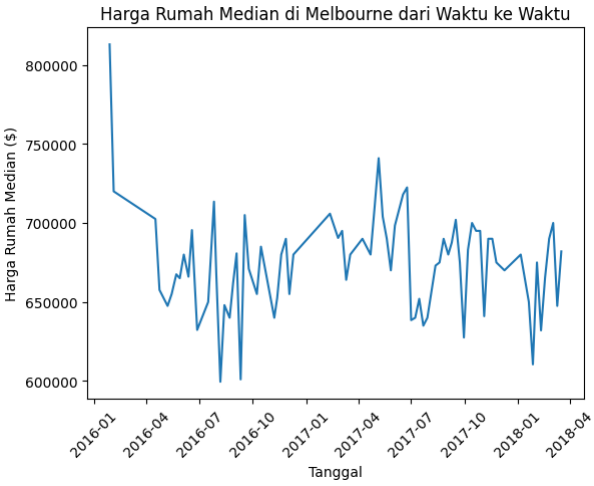


3. Line Graphs

   Dalam visualisasi line graphs ini menvisualisasikan harga rumah median di Melbourne dari waktu ke waktu. Menggunakan kolom tanggal dan kolom price. Urutan membuatannya hamper sama dengan metode visualisasi bar chart.

   Output:

```
# Mengubah tipe data string 'Date' ke tipe data object 'Datetime'
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')


#Grouping berdasarkan Date dan menghitung median 'Price'
df_grouped = df.groupby('Date')['Price'].median()


#Visualisasi dengan Line Chart
plt.plot(df_grouped.index, df_grouped.values)
plt.title('Harga Rumah Median di Melbourne dari Waktu ke Waktu')
plt.xlabel('Tanggal')
plt.ylabel('Harga Rumah Median ($)')
plt.xticks(rotation=45)
plt.show()
```

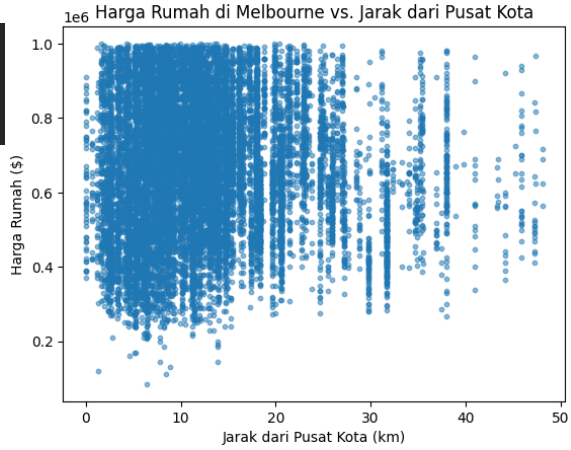


4. Scatter Plot

   Visualisasi scatter plot ini menvisualisasikan harga rumah di Melbourne vs jarak dari pusat kota. Untuk membuat visualisasinya, data yang kosong didrop dan mempertahankan nilai dalam kolom ‘Price’ yang kurang dari 3000000, sehingga outlier pada kolom ‘Price’ dihilangkan. Dalam scatter plot ditampilkan nilai ukuran titik (s) bernilai 10 dan nilai transparansi titik (alpha) senilai 0.5

   Output:

```
plt.scatter(df['Distance'], df['Price'], s=10, alpha=0.5)
plt.title('Harga Rumah di Melbourne vs. Jarak dari Pusat Kota')
plt.xlabel('Jarak dari Pusat Kota (km)')
plt.ylabel('Harga Rumah ($)')
plt.show()
```

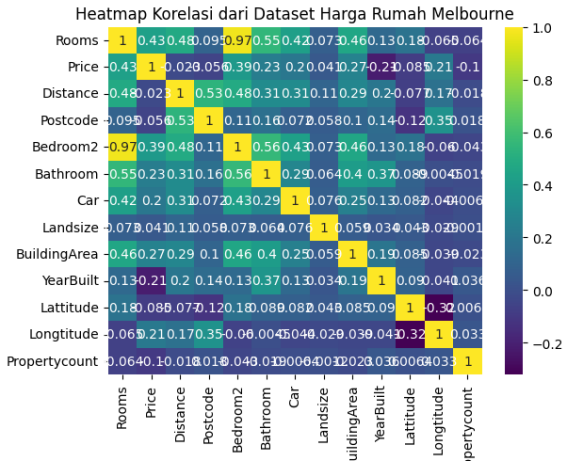


5. Heatmap

   Visualisasi heatmap kali menampilkan korelasi dari dataset harga rumah Melbourne. Bernilai 1 apabila bertemu dengan kolom yang sama, bernilai negatif apabila berbanding terbalik.

   Output:

```
sns.heatmap(df.corr(), annot=True, cmap='viridis')
plt.title('Heatmap Korelasi dari Dataset Harga Rumah Melbourne')
plt.show()
```

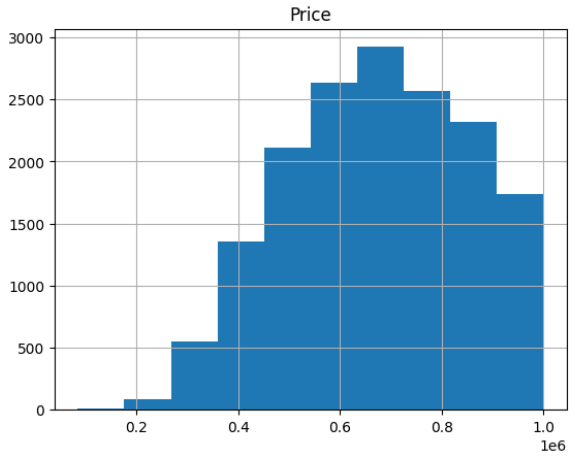


6. Visualisasi Statistik – Histogram

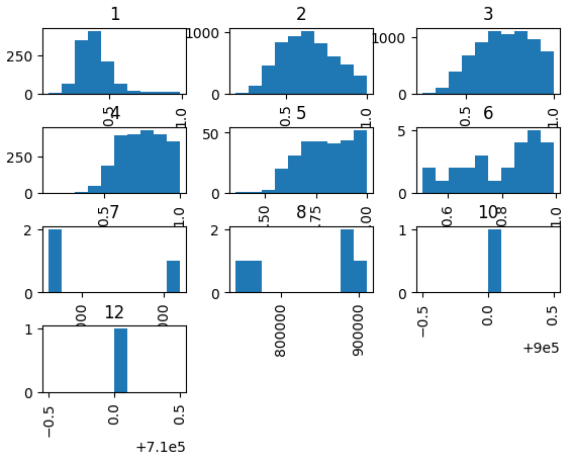
Visualisasi ini memvisualisasikan distribusi data pada kolom ‘Price’ (gambar kiri). Parameter bins=10 dimasukkan untuk membagi nilai-nilai pada kolom ‘Price’ menjadi 10 interval atau kelas yang berbeda pada histogram. Visualisasi dalam gambar kedua menampilkan distribusi data kolom ‘Price’ untuk setiap jumlah kamar pada rumah.

Output:

```
df.hist(column='Price', bins=10);
```




```
df.hist(column='Price', by='Rooms', bins=10);
```



7. Grouping

Step-step yang dilakukan untuk melakukan grouping yang pertama adalah menampilkan nilai unik dalam kolom ‘Regionname’ lalu dibuat dataframe baru sesuai kolom yang dipilih, disini digunakan kolom ‘Regionname’, ‘Rooms’, ‘Price’ baru kemudian dilakukan grouping.

Output:

```
#Menampilkan nilai unik data series
df['Regionname'].unique()
```

```
array(['Northern Metropolitan', 'Western Metropolitan',
       'Southern Metropolitan', 'Eastern Metropolitan',
       'South-Eastern Metropolitan', 'Eastern Victoria',
       'Northern Victoria', 'Western Victoria', nan], dtype=object)
```

```
#Membuat dataframe baru dengan kolom 'Regionname','Rooms', dan 'Price'
df_group_one = df[['Regionname','Rooms','Price']]
```

```
#Melakukan grouping
df_group_one = df_group_one.groupby(['Regionname'],as_index=False).mean()
df_group_one
```

	Regionname	Rooms	Price
0	Eastern Metropolitan	3.396847	1.108723e+06
1	Eastern Victoria	3.539474	7.143282e+05
2	Northern Metropolitan	2.822748	8.614840e+05
3	Northern Victoria	3.512315	6.190512e+05
4	South-Eastern Metropolitan	3.284646	8.776827e+05
5	Southern Metropolitan	2.973978	1.395928e+06
6	Western Metropolitan	3.087072	8.376153e+05
7	Western Victoria	3.295652	4.326068e+05

hasil grouping pertama dilakukan grouping antar kolomnya kemudian hasil yang didapatkan dalam output adalah nilai rata-rata kolom ‘Price’ pada setiap kelompok berdasarakan kolom ‘Regionname’ dan jumlah pada kolom ‘Rooms’.

```
#Melakukan grouping
df_gptest = df[['Regionname','Rooms','Price']]
grouped_test1 = df_gptest.groupby(['Regionname','Rooms'],as_index=False).mean()
grouped_test1
```

	Regionname	Rooms	Price
0	Eastern Metropolitan	1	5.333000e+05
1	Eastern Metropolitan	2	7.289280e+05
2	Eastern Metropolitan	3	1.017051e+06
3	Eastern Metropolitan	4	1.284803e+06
4	Eastern Metropolitan	5	1.533335e+06
...	...	...	...
65	Western Victoria	3	4.295605e+05
66	Western Victoria	4	4.419792e+05
67	Western Victoria	5	5.850000e+05
68	Western Victoria	6	NaN
69	Western Victoria	12	7.100000e+05

70 rows × 3 columns

Selanjutnya melakukan pivot dalam dataframe grouped_test dari hasil grouping kedua. Hasilnya adalah dataframe baru yang merupakan hasil pivot, di mana wilayah ('Regionname') menjadi indeks dan jumlah kamar ('Rooms') menjadi kolom. Setiap sel pada dataframe tersebut berisi nilai rata-rata harga ('Price') pada setiap kelompok yang terbentuk berdasarkan wilayah dan jumlah kamar. Outputnya seperti gambar dibawah ini:

```
#Melakukan pivot data
grouped_pivot = grouped_test1.pivot(index='Regionname',columns='Rooms')
grouped_pivot
```

Python

	Price									
Rooms	1	2	3	4	5	6	7	8	9	10
Regionname										
Eastern Metropolitan	533300.000000	728928.038117	1.017051e+06	1.284803e+06	1.533335e+06	1.664569e+06	2.136667e+06	960000.0	NaN	NaN
Eastern Victoria	462000.000000	479166.666667	6.577866e+05	7.422061e+05	1.048769e+06	NaN	NaN	910000.0	NaN	NaN
Northern Metropolitan	422378.830317	743206.263311	9.051467e+05	1.058603e+06	1.180822e+06	1.244844e+06	1.050375e+06	1773312.5	NaN	1850000.0
Northern Victoria	942000.000000	542700.000000	5.425167e+05	6.828417e+05	8.942857e+05	1.016667e+06	NaN	NaN	NaN	NaN
South-Eastern Metropolitan	396931.818182	621100.515464	8.395236e+05	1.033385e+06	1.133482e+06	1.376000e+06	1.215000e+06	NaN	1380000.0	2115000.0
Southern Metropolitan	446617.100000	840769.046467	1.470444e+06	2.024676e+06	2.637495e+06	2.640057e+06	2.542429e+06	2365353.2	NaN	2137500.0
Western Metropolitan	390100.755725	646524.963360	8.179813e+05	1.013675e+06	1.264379e+06	1.276679e+06	1.242000e+06	1143000.0	NaN	NaN
Western Victoria	NaN	344285.714286	4.295605e+05	4.419792e+05	5.850000e+05	NaN	NaN	NaN	NaN	NaN

Kemudian pada output dibawah ini, dihasilkan dataframe baru yang telah diisi nilai 0 pada sel-sel yang sebelumnya bernilai NaN (missing value).

```
# melakukan pivot data dengan mengisi missing value (NaN) dengan nilai 0
grouped_pivot = grouped_pivot.fillna(0)
grouped_pivot
```

Python

	Price									
Rooms	1	2	3	4	5	6	7	8	9	10
Regionname										
Eastern Metropolitan	533300.000000	728928.038117	1.017051e+06	1.284803e+06	1.533335e+06	1.664569e+06	2.136667e+06	960000.0	0.0	0.0
Eastern Victoria	462000.000000	479166.666667	6.577866e+05	7.422061e+05	1.048769e+06	0.000000e+00	0.000000e+00	910000.0	0.0	0.0
Northern Metropolitan	422378.830317	743206.263311	9.051467e+05	1.058603e+06	1.180822e+06	1.244844e+06	1.050375e+06	1773312.5	0.0	1850000.0
Northern Victoria	942000.000000	542700.000000	5.425167e+05	6.828417e+05	8.942857e+05	1.016667e+06	0.000000e+00	0.0	0.0	0.0
South-Eastern Metropolitan	396931.818182	621100.515464	8.395236e+05	1.033385e+06	1.133482e+06	1.376000e+06	1.215000e+06	0.0	1380000.0	2115000.0
Southern Metropolitan	446617.100000	840769.046467	1.470444e+06	2.024676e+06	2.637495e+06	2.640057e+06	2.542429e+06	2365353.2	0.0	2137500.0
Western Metropolitan	390100.755725	646524.963360	8.179813e+05	1.013675e+06	1.264379e+06	1.276679e+06	1.242000e+06	1143000.0	0.0	0.0
Western Victoria	0.000000	344285.714286	4.295605e+05	4.419792e+05	5.850000e+05	0.000000e+00	0.000000e+00	0.0	0.0	0.0

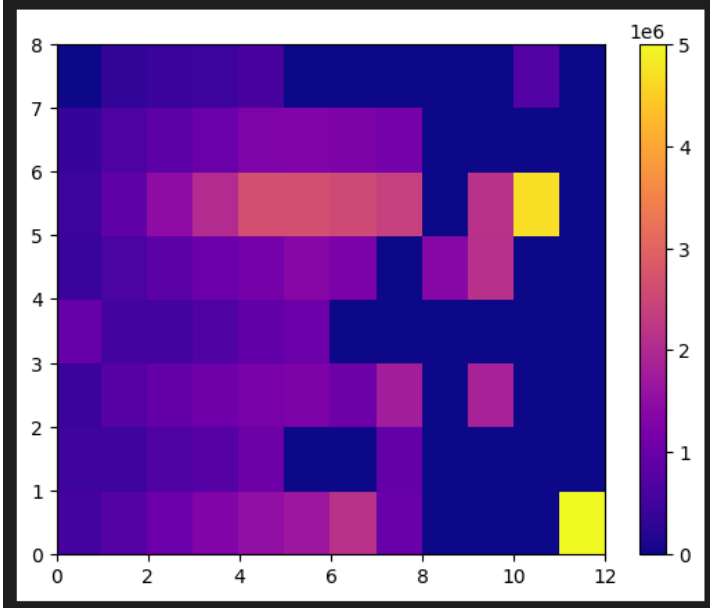
Lalu dilakukan grouping dari kolom ‘Rooms’ pada dataframe berdasarkan rata-rata nilai pada kolom ‘Price’ dengan menggunakan method groupby(). Kemudian hasil grouping disimpan dalam variabel data baru.

```
#Melakukan grouping dari  "Rooms" berdasarkan rata-rata "Price"
df_gptest2 = df[['Rooms','Price']]
grouped_test_bodystyle = df_gptest2.groupby(['Rooms'],as_index= False).mean()
grouped_test_bodystyle
```

	Rooms	Price
0	1	4.328888e+05
1	2	7.594844e+05
2	3	1.028500e+06
3	4	1.369597e+06
4	5	1.818862e+06
5	6	1.882613e+06
6	7	1.791675e+06
7	8	1.716858e+06
8	9	1.380000e+06
9	10	2.018000e+06
10	12	2.705000e+06
11	16	5.000000e+06

Terakhir divisualisasikan pivot table yang sudah dihasilkan sebelumnya dalam visualisasi heatmap. Semakin besar nilainya, maka semakin cerah warnanya. Visualisasinya tertampil dalam output dibawah ini:

```
#Visualisasi dengan Heatmap
plt.pcolor(grouped_pivot, cmap='plasma')
plt.colorbar()
plt.show()
```



8. ANOVA: Analysis of Variance

Jika ANOVA results: F-Score = 695.0002123173488 dan P-Value = 0.0, maka dapat diinterpretasikan bahwa terdapat perbedaan yang signifikan dalam harga rumah antara kelompok Regionname yang berbeda. F-Score yang tinggi menunjukkan bahwa variasi dalam data yang dijelaskan oleh faktor kelompok Regionname jauh lebih besar daripada variasi yang tidak dapat dijelaskan oleh faktor kelompok tersebut. P-Value yang sangat rendah menunjukkan bahwa kemungkinan perbedaan harga rumah antara kelompok Regionname yang berbeda terjadi secara kebetulan sangatlah kecil. Oleh karena itu, dapat disimpulkan bahwa faktor Regionname secara signifikan mempengaruhi harga rumah di Melbourne.

Output:

```
#Mengelompokan data 'Regionname' & 'Price' berdasarkan 'Regionname'
grouped_data=df[['Regionname', 'Price']].groupby(['Regionname'])
grouped_data.head(2)
```

	Regionname	Price
1	Northern Metropolitan	1480000.0
2	Northern Metropolitan	1035000.0
4	Northern Metropolitan	1465000.0
5	Northern Metropolitan	850000.0
6	Northern Metropolitan	1600000.0
66	Western Metropolitan	840000.0
67	Western Metropolitan	730000.0
68	Western Metropolitan	770000.0
70	Western Metropolitan	603000.0
71	Western Metropolitan	700000.0
133	Southern Metropolitan	1275000.0
134	Southern Metropolitan	1455000.0
135	Southern Metropolitan	2850000.0
136	Southern Metropolitan	1850000.0
138	Southern Metropolitan	1436000.0
1628	Eastern Metropolitan	1620000.0
1630	Eastern Metropolitan	645000.0

```
#Menghapus baris atau data yang memiliki nilai kosong atau hilang pada kolom 'Price'
df.dropna(subset=['Price'], inplace=True)
```

```
#Mengelompokkan data berdasarkan 'Regionname'
grouped_data = df[['Regionname', 'Price']].groupby(['Regionname'])
```

```
#ANOVA
f_val, p_val = stats.f_oneway(
    grouped_data.get_group('Northern Metropolitan')['Price'],
    grouped_data.get_group('Western Metropolitan')['Price'],
    grouped_data.get_group('Southern Metropolitan')['Price'],
    grouped_data.get_group('Eastern Metropolitan')['Price'],
    grouped_data.get_group('South-Eastern Metropolitan')['Price'],
    grouped_data.get_group('Northern Victoria')['Price'],
    grouped_data.get_group('Eastern Victoria')['Price'],
    grouped_data.get_group('Western Victoria')['Price']
)
```

```
#Hasil Anova
print("ANOVA results: F-Score =", f_val, ", P-Value =", p_val)
```

```
ANOVA results: F-Score = 695.0002123173488 , P-Value = 0.0
```

Balancing Data Secara Oversampling, Seleksi Fitur, dan Visualisasi (Materi 7)

File Source Code: UTSDataSains4.ipynb

1. Pertama-tama lakukan import library dan pembacaan data set lalu tampilkan datasetnya.
   Output:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.ensemble import ExtraTreesClassifier
```

Python

```
dataset = "test.csv"
df = pd.read_csv(dataset)
df.head()
```

Python

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	tal
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	

5 rows × 21 columns

2. Oversampling
   Tampilkan kelas dalam kolom ‘Wifi’ sebelum dilakukan oversampling (gambar kiri). Lalu dengan metode sampling data 0 yang tadinya bernilai 493 dioversampling menjadi 507 agar nilainya seimbang dan sama dengan nilai 1.
   Output:

```
print("Sebelum Oversampling:")
print(df['wifi'].value_counts())
```

```
Before Oversampling:
wifi
1    507
0    493
Name: count, dtype: int64
```

```
X = df.drop('wifi', axis=1)
y = df['wifi']

#Membuat object RandomOverSampler
ros = RandomOverSampler( random_state=42)

#Oversampling
X_res, y_res = ros.fit_resample(X, y)

print("After Oversampling:")
print(pd.Series(y_res).value_counts())
```

```
After Oversampling:
wifi
0    507
1    507
Name: count, dtype: int64
```

3. Seleksi Unvariative
   Dimulai dari gambar kiri ke kanan, pertama mengambil kolom-kolom dari indeks 0-20 dari dataframe. Lalu mengambil kolom terakhir dari dataframe sebagai target column. Kemudian dibuat objek dengan menggunakan ‘chi2’ sebagai score function dan memilih 10 fitur terbaik berdasarkan kolom ‘Wifi’. Pada gambar kanan output ditampilkan berdasarkan urutan 5 fitur terbaik.
   Output:

```
# memilih data yang dibutuhkan
X = df_resampled.iloc[:,0:21]  #independent colums
y = df_resampled.iloc[:,-1]    # target colum i.e price range
```

✓ 0.1s

```
# menerapkan SelectKBest untuk melakukan ekstraksi
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
df_resampledscores = pd.DataFrame(fit.scores_)
df_resampledcolumns = pd.DataFrame(X.columns)
```

✓ 0.1s

```
# menggabungkan 2 dataframe
featureScores = pd.concat([df_resampledcolumns,df_resampledscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(5,'Score'))
```

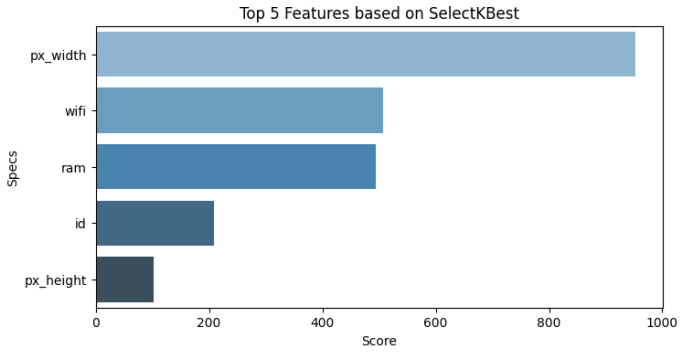
✓ 0.1s

```
        Specs       Score
13  px_width  952.670709
20      wifi  507.000000
14       ram  493.783470
0         id  208.793960
12  px_height  101.983933
```

4. Output dibawah adalah visualisasi dari hasil seleksi unvariative (5 fitur terbaik berdasarkan seleksi unvariative).
   Output:

```
top_features = featureScores.nlargest(5,'Score')
plt.figure(figsize=(8, 4))
sns.barplot(x='Score', y='Specs', data=top_features, palette='Blues_d')
plt.title('Top 5 Features based on SelectKBest')
plt.show()
```

0.2s



5. Feature Importance
   Pertama dilakukan pemilihan fitur atau kolom yang akan digunakan untuk melakukan dikalisifikasikan dengan menyimpan pada variabel x, dan variabel y diisi dengan kolom yang menjadi target klasifikasi. Lalu model Extra Trees Classifier diinisialisasi dengan variabel model, baru kemudian dilakukan elatigan model dengan memanggil method fit pada objek model x dan y. Terakhir dilakukan perhitungan skor pentingnya fitur dalam kolom yang digunakan dalam kasifikasinya. Semakin penting fiturnya, maka semakin besar nilai skor pada suatu fitur.
   Output:

```
X = df_resampled.iloc[:,0:21]
y = df_resampled.iloc[:,-1]
df_resampled.head()
```

0.1s Python

	id	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	sc_w	tal
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12	7	
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6	0	
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17	10	
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10	0	
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15	8	

5 rows × 21 columns

```
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
```

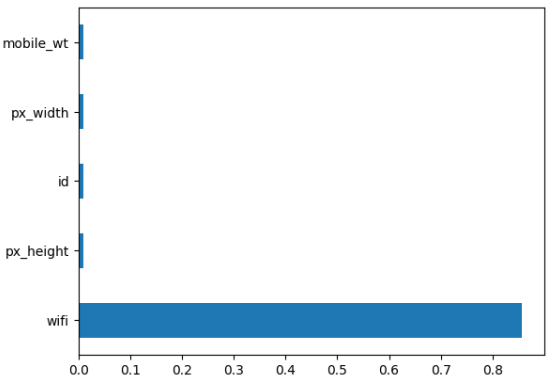
0.3s Python

```
[0.00707105 0.00763526 0.0046391  0.00750704 0.00436597 0.00712516
 0.00391355 0.00654537 0.00616126 0.00792068 0.00606807 0.0071388
 0.00822292 0.00835683 0.00696756 0.00721517 0.00696556 0.00721261
 0.00318803 0.00408511 0.87169487]
```

6. Output atau gambar dibawah ini adalah visualisasi bar chart dari feature importance.
   Output:

```
# melakukan plot
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```

0.3s



7. Matriks Korelasi dengan Heatmap

Pertama mengambil kolom, lalu dibuat matriks korelasinya dari semua fitur dalam dataframe. Kemudian diambil indeks dari ‘corrmat’ dan ditampilkan dalam bentuk plot heatmap berukuran 20x20. Dalam visualisasinya ditampilkan korelasi antara fitur-fitur menggunakan skala warna. Semakin besar nilainya, maka korelasinya semakin erat.

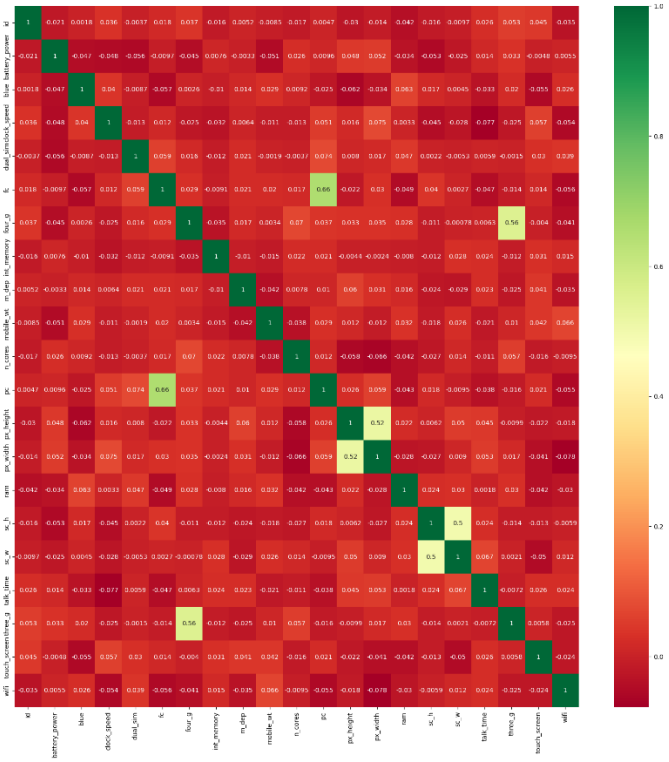
Output:

```
import seaborn as sns
```
0.0s

```
X = df_resampled.iloc[:,0:21]
y = df_resampled.iloc[:,-1]
```
0.0s

```
#Mendapatkan  correlations dari setiap fitur dalam dataset
corrmat = df_resampled.corr()
top_corr_features = corrmat.index
```
0.0s

```
#Plot heatmap
plt.figure(figsize=(20,20))
g=sns.heatmap(df_resampled[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```
2.5s



Cleaning Dataset (Materi 8)

File Source Code: UTSDataSains5.ipynb

1. Membuang (drop) kolom

   Output pertama menampilkan dataframe yang belum di drop. Kemudian dipilih kolom-kolom yang akan di drop atau dibersihkan dan dilakukan drop data yang dipilih. Lalu output terakhir menampilkan dataframe setelah kolom yang dipilih di drop.

   Output:

```
df.head(1)
```

✓ 0.0s Python

	_id	id	date_order	date_finished	mode	from_alamat	from_kelurahan	from_kecamatan	from_lat	from_lng	...	customer_birthdate	drive
0	1	1617	09/03/2019 20.45	09/03/2019 22.05	BIKE	Gang Ikhwan No.16, Sungai Jawi	DARAT SEKIP	PONTIANAK KOTA	-0,0303277	109,297753	...	1994-02-05T00:00:00	

1 rows × 32 columns

```
#Memilih kolom-kolom yang akan di drop
to_drop = ['_id',
           'from_lat',
           'from_lng',
           'to_lat',
           'to_lng',
           'from_kelurahan',
           'customer_gender',
           'amount_delivery',
           'amount_merchant',
           'to_kelurahan',
           'customer_id',
           'driver_id',
           'driver_gender',
           'kendaraan_jenis',
           'kendaraan_merk']

#Melakukan drop pada data yang dipilih
df.drop(to_drop, inplace=True, axis=1)
```

   Setelah di drop

```
df.head(1)
```

✓ 0.0s Python

	id	date_order	date_finished	mode	from_alamat	from_kecamatan	to_alamat	to_kecamatan	distance	transaction_amount_total	customer_nan
0	1617	09/03/2019 20.45	09/03/2019 22.05	BIKE	Gang Ikhwan No.16, Sungai Jawi	PONTIANAK KOTA	Jl. Prof. M.Yamin No.3a, Sungai Bangkong	PONTIANAK SELATAN	5,55	2500	Dian Arya

2. Mengubah indeks di DataFrame

   Pertama-tama dicek apakah setiap nilai di kolom ‘id’ unik, jika iya maka outputnya ‘True’. Lalu kolom ‘id’ diubah menjadi indeks dataframe menggunakan netode ‘set_index’. Pada gambar kanan dataframe yang telah di indeks digunakan untuk memperoleh data yang terkait dengan indeks 2053 dengan metode ‘loc’.

   Output:

```
df = df.set_index('id')
df.head()
```

✓ 0.0s Python Python

id	date_order	date_finished	mode	from_alamat	from_kecamatan	to_alamat	to_kecamatan	distance	transaction_amount_total	customer_name
1617	09/03/2019 20.45	09/03/2019 22.05	BIKE	Gang Ikhwan No.16, Sungai Jawi	PONTIANAK KOTA	Jl. Prof. M.Yamin No.3a, Sungai Bangkong	PONTIANAK SELATAN	5,55	2500	Dian Aryani
1297	09/03/2019 19.55	10/03/2019 01.38	FOOD	Neo Shabu-Shabu Steak & Shake, Johar, Jl. Joha...	PONTIANAK KOTA	Jl. Dare Nandong Villa Ria Indah, Tj. Hulu	PONTIANAK TIMUR	7,08	91000	Ajimin Aditya Pangestu
1394	09/03/2019 19.54	09/03/2019 21.44	SHOP	Alfamart Pontianak Mall, Jl Teuku Umar	PONTIANAK KOTA	Gg. Gn. Malabar No.21, Sungai Jawi	PONTIANAK BARAT	4,02	63500	Wahyu Simanjuntak
1120	09/03/2019 18.56	10/03/2019 00.20	FOOD	Parklife, Jl. Karimata No.64, Sungai Bangkong,...	PONTIANAK KOTA	Unnamed Road, Pal IX	PONTIANAK TENGGARA	8,94	84000	Argono Daru Tarihoran S.Sos
				Jl. Tabrani Ahmad		Pal IX,				

```
df.loc[2053]
```

✓ 0.0s

```
date_order                    09/03/2019 12.28
date_finished                 09/03/2019 17.11
mode                                       CAR
from_alamat          Jl. Tabrani Ahmad No.12,  Sungai Jawi Dalam
from_kecamatan                  PONTIANAK BARAT
to_alamat                    Pal IX,  Kakap River
to_kecamatan                 PONTIANAK TENGGARA
distance                                   7,93
transaction_amount_total                  14400
customer_name                     Oman Prasasta
customer_birthdate          2004-01-23T00:00:00
driver_name                 Maimunah Pudjiastuti
driver_birthdate            1988-05-02T00:00:00
merchant_id                                 NaN
merchant_name                               NaN
merchant_category                           NaN
Name: 2053, dtype: object
```

3. Merapihkan fields dalam data

   Pertama hitung frekuensi nilai objek, integer, dan float. Kemudian dilakukan seleksi data dari kolom ‘customer_birthdate’. Lalu dengan menggunakan regex atau ekspresi rehuler untuk mengekstrak tahun lahir dan dijalankan di datasetnya. Kemudian tampil hasilnya dalam output di gambar kanan.

   Output:

```
#Menghitung frekuensi nilai objek
df.dtypes.value_counts()
```
✓ 0.0s
```
object     14
int64       1
float64     1
Name: count, dtype: int64
```
```
#Seleksi data dari 2004-01-23T00:00:00 dengan output hanya pada tahun
df.loc[2053:, 'customer_birthdate'].head(10)
```
✓ 0.0s
```
id
2053    2004-01-23T00:00:00
1574    1994-10-31T00:00:00
1882    1988-04-12T00:00:00
1958    1992-02-27T00:00:00
2009    2004-04-01T00:00:00
1671    1985-03-28T00:00:00
1847    1974-04-18T00:00:00
1408    1997-01-26T00:00:00
1439    1988-01-21T00:00:00
1766    2005-07-07T00:00:00
Name: customer_birthdate, dtype: object
```

```
#Menggunakan ekspresi reguler (Regex) tunggal untuk mengekstrak tahun lahir
regex = r'^(\d{4})'
```
✓ 0.0s
```
#Jalankan regex di dataset
extr = df['customer_birthdate'].str.extract(r'^(\d{4})', expand=False)
extr.head()
```
✓ 0.0s
```
id
1617    1994
1297    2004
1394    2000
1120    1987
2053    2004
Name: customer_birthdate, dtype: object
```
```
df['customer_birthdate'] = pd.to_numeric(extr)
df['customer_birthdate'].dtype
```
✓ 0.0s
```
dtype('int64')
```

4. Membersihkan kolom dengan kombinasi metode str dengan NumPy

   Pertama-tama tampilkan kolom ‘from_kecamatan’ dalam dataframe. Lalu buat variabel baru yang berisi data dalam kolom ‘from_kecamatan’ dalam dataframe. Kemudian buat variabel baru pbesar untuk menampung hasil pencarian dalam barus pub yang mengandung kata ‘PONTIANAK KOTA’. Pada output sudah berhasil dimanipulasi datanya menjadi ‘Pontianak Kota’.

   Output:

```
df['from_kecamatan'].head(10)
```
✓ 0.0s
```
id
1617         PONTIANAK KOTA
1297         PONTIANAK KOTA
1394         PONTIANAK KOTA
1120         PONTIANAK KOTA
2053        PONTIANAK BARAT
1574    PONTIANAK TENGGARA
1882         PONTIANAK KOTA
1958        PONTIANAK BARAT
2009    PONTIANAK TENGGARA
1671         PONTIANAK KOTA
Name: from_kecamatan, dtype: object
```
```
pub = df['from_kecamatan']
pbesar = pub.str.contains('PONTIANAK KOTA')
pbesar[:4]
```
✓ 0.0s
```
id
1617    True
1297    True
1394    True
1120    True
Name: from_kecamatan, dtype: bool
```

```
pkecil = pub.str.contains('PONTIANAK KOTA')
```
✓ 0.0s
```
df['from_kecamatan'] = np.where(pbesar, 'Pontianak Kota',
                                np.where(pkecil, 'PONTIANAK KOTA',
                                         pub.str.replace('-', ' ')))
df['from_kecamatan'].head(4)
```
✓ 0.0s
```
id
1617    Pontianak Kota
1297    Pontianak Kota
1394    Pontianak Kota
1120    Pontianak Kota
Name: from_kecamatan, dtype: object
```

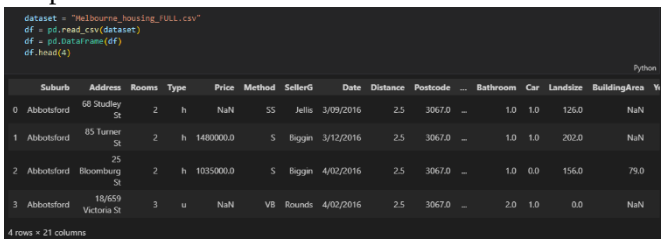
Metoda Imputasi (Materi 9)

File Source Code: UTSDataSains6.ipynb

1. Imputasi Mean

   Teknik ini mengganti nilai atau data yang hilang (NaN) dengan nilai mean (rata-rata). Dalam dataset terdapat data hilang (NaN) pada data ke 0 dan 3 kolom ‘Price’ serta data ke 0,1,3 kolom ‘BuildingArea’, diganti dengan nilai rata-rata dari masing-masing kolom yang terdapat nilai NaN. Gambar kiri adalah dataframe yang belum diimputasi mean. Gambar kanan adalah dataset yang sudah diisi dengan imputasi mean.

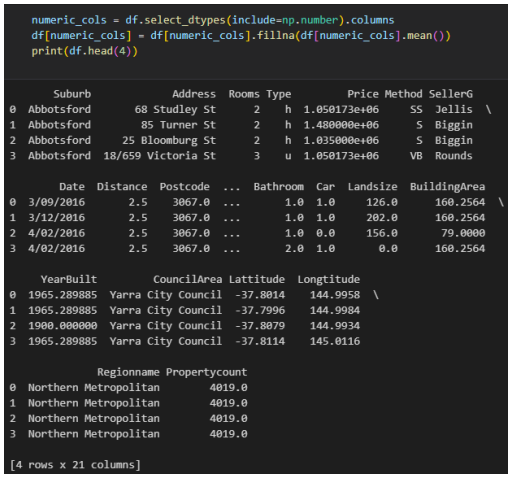
   Output:



```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns



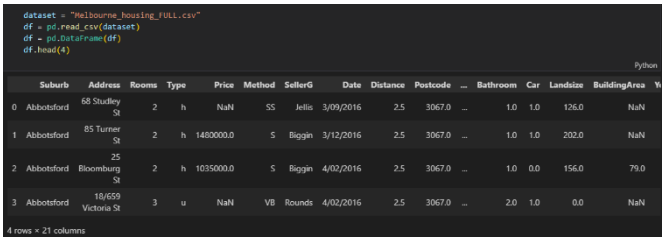
```
numeric_cols = df.select_dtypes(include=np.number).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
print(df.head(4))
```

```
      Suburb            Address  Rooms Type         Price Method SellerG  \
0  Abbotsford       68 Studley St      2    h  1.050173e+06     SS  Jellis
1  Abbotsford       85 Turner St      2    h  1.480000e+06      S  Biggin
2  Abbotsford  25 Bloomburg St      2    h  1.035000e+06      S  Biggin
3  Abbotsford  18/659 Victoria St      3    u  1.050173e+06     VB  Rounds

        Date  Distance  Postcode  ...  Bathroom  Car  Landsize  BuildingArea  \
0  3/09/2016       2.5    3067.0  ...       1.0  1.0     126.0      160.2564
1  3/12/2016       2.5    3067.0  ...       1.0  1.0     202.0      160.2564
2  4/02/2016       2.5    3067.0  ...       1.0  0.0     156.0       79.0000
3  4/02/2016       2.5    3067.0  ...       2.0  1.0       0.0      160.2564

     YearBuilt        CouncilArea Lattitude  Longtitude
0  1965.289885  Yarra City Council  -37.8014    144.9958
1  1965.289885  Yarra City Council  -37.7996    144.9984
2  1900.000000  Yarra City Council  -37.8079    144.9934
3  1965.289885  Yarra City Council  -37.8114    145.0116

             Regionname  Propertycount
0  Northern Metropolitan         4019.0
1  Northern Metropolitan         4019.0
2  Northern Metropolitan         4019.0
3  Northern Metropolitan         4019.0

[4 rows x 21 columns]
```

2. Imputasi Nilai Suka-suka (Arbitrary)

   Teknik menggantikan data yang hilang atau NaN dengan inputan tipe data numerik. Dalam dataset, dalam kolom ‘Price’ terdapat data yang hilang digantikan dengan nilai 88 sesuai dengan nilai suka-suka yang diberikan. Gambar kiri adalah dataframe yang belum diimputasi nilai suka-suka. Gambar kanan adalah dataset yang sudah diisi dengan imputasi nilai suka-suka.

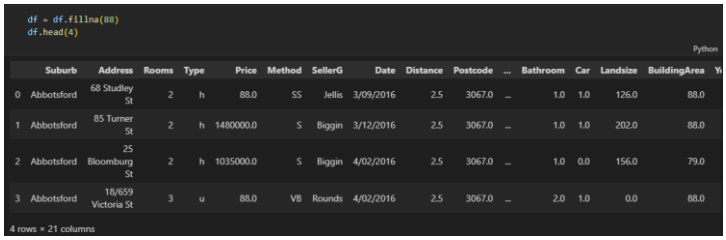
   Output:



```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns



```
df = df.fillna(88)
df.head(4)
```

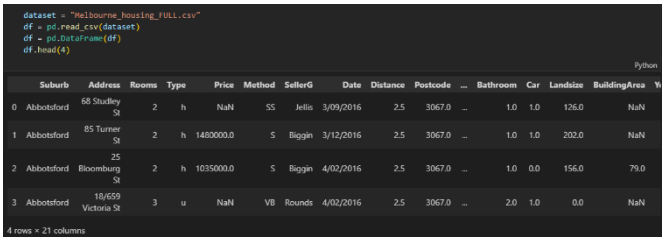
	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	88.0	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	88.0	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	88.0	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	88.0	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	88.0	

4 rows × 21 columns

3. Imputasi End of Tail

   Teknik untuk menggantikan nilai atau data yang hilang atau dalam bentuk NaN. NaN digantikan dengan menggunakan distribusi normal (Gaussian distribution) pada ekor kanan yang dipilih. Gambar kiri adalah dataframe yang belum imputasi end of tail. Gambar kanan adalah dataset yang sudah diisi dengan imputasi end of tail.

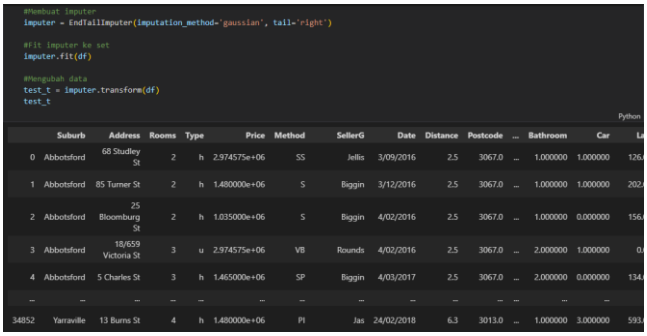
   Output:



```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns



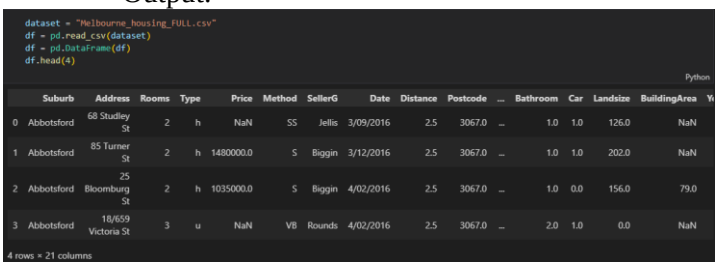
```
#Membuat imputer
imputer = EndTailImputer(imputation_method='gaussian', tail='right')

#Fit imputer ke set
imputer.fit(df)

#Mengubah data
test_t = imputer.transform(df)
test_t
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	La
0	Abbotsford	68 Studley St	2	h	2.974575e+06	SS	Jellis	3/09/2016	2.5	3067.0	...	1.000000	1.000000	126.
1	Abbotsford	85 Turner St	2	h	1.480000e+06	S	Biggin	3/12/2016	2.5	3067.0	...	1.000000	1.000000	202.
2	Abbotsford	25 Bloomburg St	2	h	1.035000e+06	S	Biggin	4/02/2016	2.5	3067.0	...	1.000000	0.000000	156.
3	Abbotsford	18/659 Victoria St	3	u	2.974575e+06	VB	Rounds	4/02/2016	2.5	3067.0	...	2.000000	1.000000	0.
4	Abbotsford	5 Charles St	3	h	1.465000e+06	SP	Biggin	4/03/2017	2.5	3067.0	...	2.000000	0.000000	134.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	h	1.480000e+06	PI	Jas	24/02/2018	6.3	3013.0	...	1.000000	3.000000	593.

4. Imputasi Frequent Category atau Modus

   Teknik untuk menggantikan nilai atau data yang hilang atau dalam bentuk NaN dan digunakan bagi tipe data kategori. Dalam output terlihat data dalam kolom yang ada data NaN digantikan dengan data yang ada dalam satu kategori dan digantikan dengan nilai modus. Gambar kiri adalah dataframe yang belum imputasi modus. Gambar kanan adalah dataset yang sudah diisi dengan imputasi modus.

   Output:

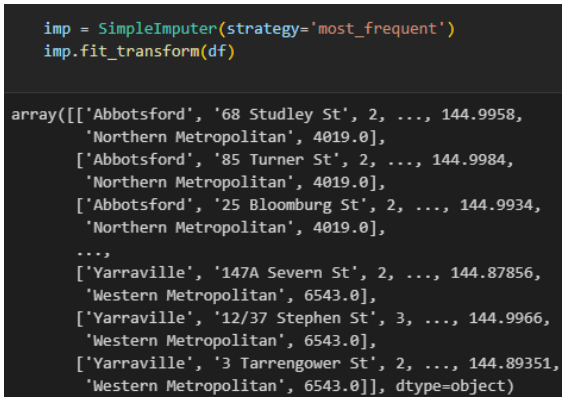


```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns



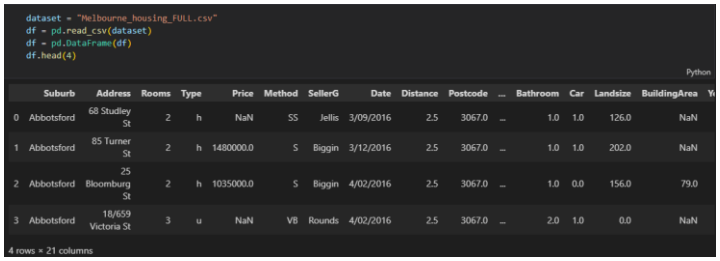
```
imp = SimpleImputer(strategy='most_frequent')
imp.fit_transform(df)
```

```
array([['Abbotsford', '68 Studley St', 2, ..., 144.9958,
        'Northern Metropolitan', 4019.0],
       ['Abbotsford', '85 Turner St', 2, ..., 144.9984,
        'Northern Metropolitan', 4019.0],
       ['Abbotsford', '25 Bloomburg St', 2, ..., 144.9934,
        'Northern Metropolitan', 4019.0],
       ...,
       ['Yarraville', '147A Severn St', 2, ..., 144.87856,
        'Western Metropolitan', 6543.0],
       ['Yarraville', '12/37 Stephen St', 3, ..., 144.9966,
        'Western Metropolitan', 6543.0],
       ['Yarraville', '3 Tarrengower St', 2, ..., 144.89351,
        'Western Metropolitan', 6543.0]], dtype=object)
```

5. Imputasi Random Sample

   Teknik imputasi ini menggantikan nilai yang hilang (NaN) dengan nilai acak yang diambil dari data yang ada. Random sample menghasilkan variasi data yang lebih banyak sehingga bisa menghindari kemungkinan overfitting. Gambar kiri adalah dataframe yang belum imputasi random sample. Gambar kanan adalah dataset yang sudah diisi dengan imputasi random sample.

   Output:

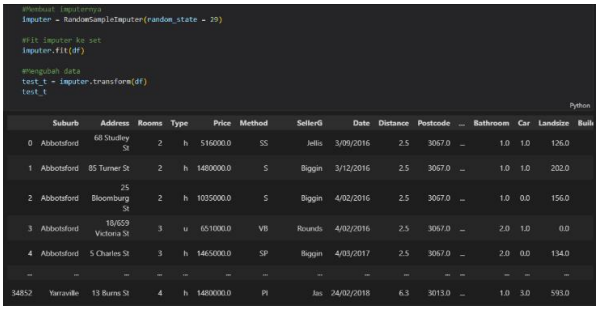


```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns



```
#Membuat imputernya
imputer = RandomSampleImputer(random_state = 29)

#fit imputer ke set
imputer.fit(df)

#Mengubah data
test_t = imputer.transform(df)
test_t
```

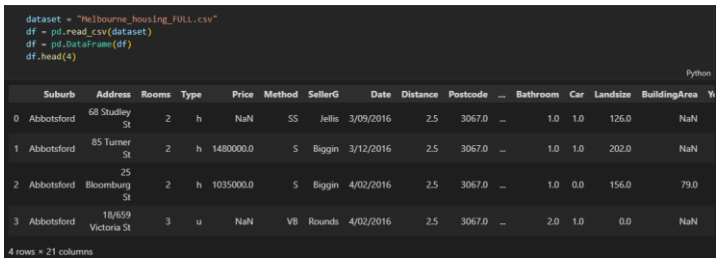
Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	Build
0	Abbotsford	68 Studley St	2	h	516000.0	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	
3	Abbotsford	18/659 Victoria St	3	u	651000.0	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	h	1480000.0	PI	Jas	24/02/2018	6.3	3013.0	...	1.0	3.0	593.0	

6. Imputasi Nilai Nol/Konstanta

   Digunakan untuk menggantikan nilai yang hilang dengan imputasi nilai nol atau konstanta. Dalam output yang data terdapat NaN diganti dengan nilai 0. Gambar kiri adalah dataframe yang belum imputasi nilai 0. Gambar kanan adalah dataset yang sudah diisi dengan imputasi nilai 0.

   Output:



```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns

```
df.fillna(0)
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	Build
0	Abbotsford	68 Studley St	2	h	0.0	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	
3	Abbotsford	18/659 Victoria St	3	u	0.0	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	h	1480000.0	PI	Jas	24/02/2018	6.3	3013.0	...	1.0	3.0	593.0	
34853	Yarraville	29A Murray St	2	h	888000.0	SP	Sweeney	24/02/2018	6.3	3013.0	...	2.0	1.0	98.0	
34854	Yarraville	147A Severn St	2	t	705000.0	S	Jas	24/02/2018	6.3	3013.0	...	1.0	2.0	220.0	
34855	Yarraville	12/37 Stephen St	3	h	1140000.0	SP	hockingstuart	24/02/2018	6.3	3013.0	...	0.0	0.0	0.0	

7. Imputasi Regresi: Deterministik

Teknik imputasi ini mengisi nilai kosong dengan hasil regresi dan nilai yang masih memiliki korelasi. Pertama baca terlebih dahulu dataframenya, baru kemudian tampilkan dataframenya. Lalu akan terlihat data apa saja yang kosong, data kosong ini dihitung nilai nullnya. Kemudian lakukan perhitungan regresi dan tampilkan hasil dalam visualisasi.

Output:

```
dataset = "Melbourne_housing_FULL.csv"
df = pd.read_csv(dataset)
df = pd.DataFrame(df)
df.head(4)
```

Python

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	

4 rows × 21 columns

```
df.loc[df["Price"] == 0.0, "Price"] = np.NAN
df.loc[df["Bedroom2"] == 0.0, "Bedroom2"] = np.NAN
df.loc[df["Bathroom"] == 0.0, "Bathroom"] = np.NAN
df.loc[df["Car"] == 0.0, "Car"] = np.NAN
df.loc[df["Landsize"] == 0.0, "Landsize"] = np.NAN
df.loc[df["BuildingArea"] == 0.0, "BuildingArea"] = np.NAN
df.loc[df["YearBuilt"] == 0.0, "YearBuilt"] = np.NAN
df.loc[df["Lattitude"] == 0.0, "Lattitude"] = np.NAN
df.loc[df["Longtitude"] == 0.0, "Longtitude"] = np.NAN

#Menghitung nilai null dari dataframe, dan hanya 9 kolom dengan menambahkan [0:9]
df.isnull().sum()[0:9]
```

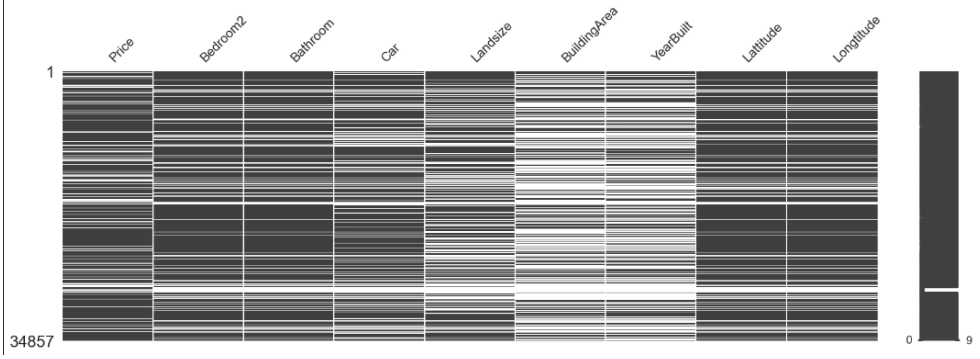
```
Price           7610
Bedroom2        8234
Bathroom        8272
Car            10359
Landsize       14247
BuildingArea   21191
YearBuilt      19306
Lattitude       7976
Longtitude      7976
dtype: int64
```

Visualisasi matriks dibawah ini terlihat masih ada beberapa missing value atau nilai yang hilang.

```
mno.matrix(df, figsize = (20, 6))
```

Pyth

```
<Axes: >
```

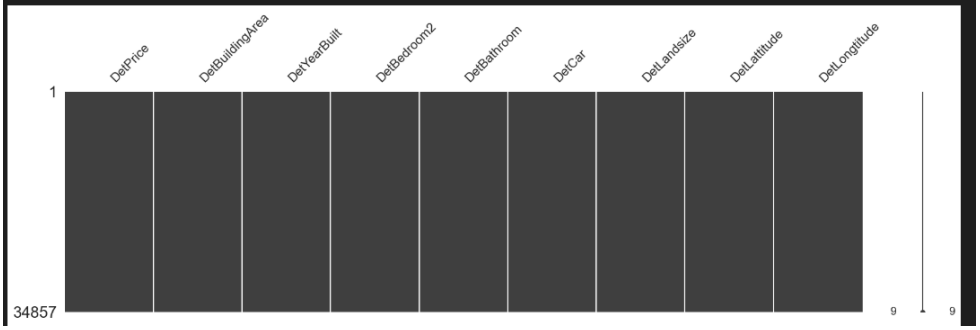


Bentuk visualisasi matriks yang sudah diisi nilai yang hilangnya dengan metode regresi deterministik.

```
#Membuat matriks
mno.matrix(deter_df, figsize = (20,5))
```

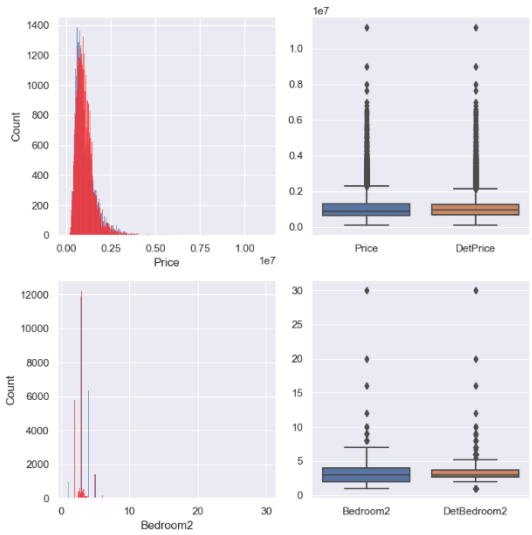
Python

```
<Axes: >
```



Visualisasi chart yang menunjukan distribusi dari variabel ‘Price’ dan ‘Bedroom2’ dan distribusi setelah dilakukan imputasi dengan menggunakan Teknik imputasi regresi deterministik.

```
#Membuat chart
sns.set()
fig, axes = plt.subplots(nrows = 2, ncols = 2)
fig.set_size_inches(8, 8)

for index, variable in enumerate(["Price", "Bedroom2"]):
    sns.histplot(df[variable].dropna(), kde = False, ax = axes[index, 0])
    sns.histplot(deter_df["Det" + variable], kde = False, ax = axes[index, 0], color = 'red')

    sns.boxplot(data = pd.concat([df[variable], deter_df["Det" + variable]], axis = 1),
                ax = axes[index, 1])

plt.tight_layout()
```



Output dibawah menampilkan perhitungan statistika untuk kolom ‘Price’ dan ‘Bedroom2’ yang sudah dan belum di imputasi dengan metode regresi deterministik (‘DetPrice’ dan ‘DetBedroom2’.

```
#Menampilkan nilai perhitungan statistika untuk kolom tertentu
pd.concat([df[["Price", "Bedroom2"]], deter_df[["DetPrice", "DetBedroom2"]]], axis = 1).describe().T
```

	count	mean	std	min	25%	50%	75%	max
Price	27247.0	1.050173e+06	641467.130105	85000.0	635000.000000	870000.0	1.295000e+06	11200000.0
Bedroom2	26623.0	3.086617e+00	0.977899	1.0	2.000000	3.0	4.000000e+00	30.0
DetPrice	34857.0	1.060287e+06	578282.280121	85000.0	690000.000000	945000.0	1.270000e+06	11200000.0
DetBedroom2	34857.0	3.080773e+00	0.882384	1.0	2.682897	3.0	3.724045e+00	30.0

8. Imputasi Regresi: Stokastik

   Hampir mirip dengan regresi determinasi untuk langkah-langkahnya. Bedanya untuk regresi stokastik ini memprediksi nilai yang hilang dengan mempertimbangkan variasi dalam data yang ada. Teknik imputasi ini memperhitungkan ketidakpastian yang ada dalam model regresinya dan membuat beberapa prediksinya.

   Output:

```
df = pd.read_csv("Melbourne_housing_FULL.csv")
df.head(10)
```

Python

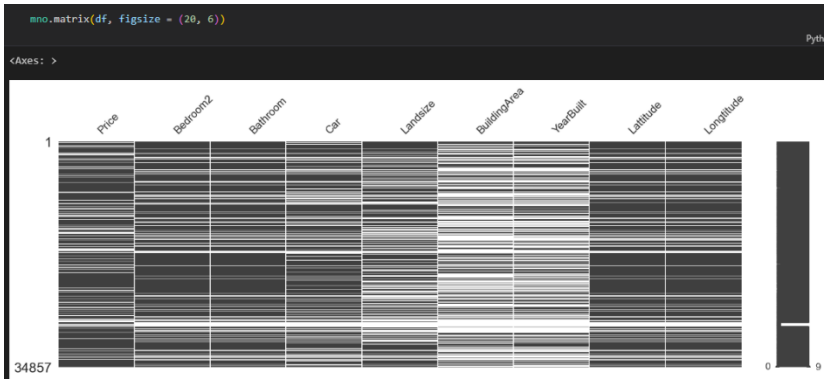
	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	Y
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	126.0	NaN	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	202.0	NaN	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	156.0	79.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	0.0	NaN	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	134.0	150.0	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3067.0	...	2.0	1.0	94.0	NaN	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3067.0	...	1.0	2.0	120.0	142.0	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	3067.0	...	2.0	2.0	400.0	220.0	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	3067.0	...	1.0	2.0	201.0	NaN	

```
df.loc[df["Price"] == 0.0, "Price"] = np.NAN
df.loc[df["Bedroom2"] == 0.0, "Bedroom2"] = np.NAN
df.loc[df["Bathroom"] == 0.0, "Bathroom"] = np.NAN
df.loc[df["Car"] == 0.0, "Car"] = np.NAN
df.loc[df["Landsize"] == 0.0, "Landsize"] = np.NAN
df.loc[df["BuildingArea"] == 0.0, "BuildingArea"] = np.NAN
df.loc[df["YearBuilt"] == 0.0, "YearBuilt"] = np.NAN
df.loc[df["Lattitude"] == 0.0, "Lattitude"] = np.NAN
df.loc[df["Longtitude"] == 0.0, "Longtitude"] = np.NAN

#Menghitung nilai null dari dataframe, dan hanya 9 kolom dengan menambahkan [0:9]
df.isnull().sum()[0:9]
```

```
Price           7610
Bedroom2        8234
Bathroom        8272
Car            10359
Landsize       14247
BuildingArea   21191
YearBuilt      19306
Lattitude       7976
Longtitude      7976
dtype: int64
```

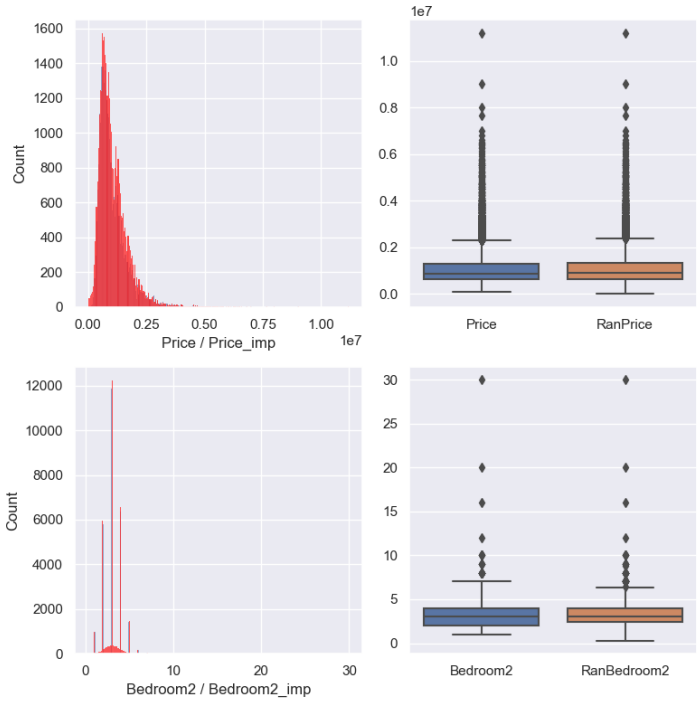
Visualisasi matriks dibawah ini terlihat masih ada beberapa missing value atau nilai yang hilang.

```
msno.matrix(df, figsize = (20, 6))
```



Visualisasi chart yang menunjukan distribusi dari variabel ‘Price’ dan ‘Bedroom2’ dan distribusi setelah dilakukan imputasi dengan menggunakan Teknik imputasi regresi deterministik.

```
sns.set()
fig, axes = plt.subplots(nrows = 2, ncols = 2)
fig.set_size_inches(8, 8)

for index, variable in enumerate(["Price", "Bedroom2"]):
    sns.histplot(df[variable].dropna(), kde = False, ax = axes[index, 0])
    sns.histplot(random_data["Ran" + variable], kde = False, ax = axes[index, 0], color = 'red')
    axes[index, 0].set(xlabel = variable + " / " + variable + '_imp')

    sns.boxplot(data = pd.concat([df[variable], random_data["Ran" + variable]], axis = 1),
                ax = axes[index, 1])

    plt.tight_layout()
```



Output dibawah menampilkan perhitungan statistika untuk kolom ‘Price’ dan ‘Bedroom2’ yang sudah dan belum di imputasi dengan metode regresi deterministik (‘RanPrice’ dan ‘RanBedroom2’.

```
pd.concat([df[["Price", "Bedroom2"]], random_data[["RanPrice", "RanBedroom2"]]], axis = 1).describe().T
```

	count	mean	std	min	25%	50%	75%	max
Price	27247.0	1.050173e+06	641467.130105	85000.000000	635000.000000	870000.0	1295000.0	11200000.0
Bedroom2	26623.0	3.086617e+00	0.977899	1.000000	2.000000	3.0	4.0	30.0
RanPrice	34857.0	1.067367e+06	628491.999776	385.076779	645000.000000	906000.0	1340000.0	11200000.0
RanBedroom2	34857.0	3.078863e+00	0.952660	0.297721	2.435392	3.0	4.0	30.0

Rekonstruksi Data (Materi 9) Yaitu Proses Scaling, Standarisasi, Normalisasi

File Source Code: UTSDataSains7.ipynb

1. Scaling (Penskalaan)

   Teknik melakukan rekonstruksi data numerik agar nilainya berada dalam skala yang sama atau setara. Scaling ini dilakukan untuk menghindari masalah yang timbul akibat perbedaan skala data yang signifikan antara kolomnya. Dalam kolom yang digunakan disini menggunakan kolom ‘Landsize’ dan kolom ‘BuildingArea’. Hasil scalingnya ditampilkan dalam gambar kanan.

   Output:

```
# Pilih kolom-kolom yang akan dijadikan fitur
fitur = ["Landsize", "BuildingArea"]

# Cek apakah terdapat nilai NaN pada dataset
data.isna().sum()

# Isi nilai NaN dengan nilai rata-rata dari kolom yang bersangkutan
imputer = SimpleImputer(strategy='mean')
data[fitur] = imputer.fit_transform(data[fitur])
data[fitur]
```

	Landsize	BuildingArea
0	126.000000	160.2564
1	202.000000	160.2564
2	156.000000	79.0000
3	0.000000	160.2564
4	134.000000	150.0000
...	...	...
34852	593.000000	160.2564
34853	98.000000	104.0000
34854	220.000000	120.0000
34855	593.598993	160.2564
34856	250.000000	103.0000

34857 rows × 2 columns

```
# Buat objek normalizer
max_abs = Normalizer(norm='l2')

# Sesuaikan normalizer dengan data fitur yang dipilih
max_abs.fit(data[fitur])

# Mengubah data fitur yang telah dinormalisasi
train_scaled = max_abs.transform(data[fitur])
```

```
train_scaled
```

```
array([[0.61807707, 0.78611751],
       [0.7834042 , 0.62151256],
       [0.89212798, 0.45178276],
       ...,
       [0.87789557, 0.47885213],
       [0.96543536, 0.26064262],
       [0.92460149, 0.38093581]])
```

2. Standarisasi

   Standarisasi adalah metode scaling yang digunakan untuk mengubah data menjadi distribusi normal dengan mean =0. Metode ini membuat objek scaler dan menyesuaikan dengan data supaya nilai antar kolomnya tidak berbanding terlalu jauh atau menghasilkan nilai-nilai ekstrim. Kolom yang digunakan yaitu kolom ‘Landsize’ dan kolom ‘BuildingArea’.

   Output:

```
# Load/memuat/membaca data
data = pd.read_csv("Melbourne_housing_FULL.csv")

# Pilih kolom-kolom yang akan dijadikan fitur
fitur = ["Landsize", "BuildingArea"]

# Cek apakah terdapat nilai NaN pada dataset
data.isna().sum()

# Isi nilai NaN dengan nilai rata-rata dari kolom yang bersangkutan
imputer = SimpleImputer(strategy='mean')
data[fitur] = imputer.fit_transform(data[fitur])
data[fitur].describe()
```

	Landsize	BuildingArea
count	34857.000000	34857.000000
mean	593.598993	160.256400
std	2763.694121	251.943934
min	0.000000	0.000000
25%	357.000000	160.000000
50%	593.598993	160.256400
75%	598.000000	160.256400
max	433014.000000	44515.000000

```
#Buat objek scaler
scaler = StandardScaler()

#Sesuaikan scaler dengan data
scaler.fit(data[fitur])

#Mengubah data kereta
train_scaled = scaler.transform(data[fitur])
```

```
#Tampilkan hasil
train_scaled
```

```
array([[-1.69195895e-01, -8.32865021e-01],
       [-1.41696075e-01, -8.32865021e-01],
       [-1.58340703e-01, -1.97558347e+00],
       ...,
       [-1.35182960e-01,  3.09853427e-01],
       [ 0.00000000e+00,  2.53734466e-16],
       [-1.24327768e-01, -1.97558347e+00]])
```

3. Normalisasi

   Normasilasi merekonstruksi data ke skala yang baru, seperti yang ada dalam output, nilanya antar -1 sampai 1, bahkan ada yang nilainya di bawah -0. Metode ini digunakan untuk memudahkan perbandingan data dan memperbaikinya. Teknik yang digunakan dalam metode ini yaitu min-max scaling. Terlihat dalam prosesnya dihitung terlebih dahulu rata-ratanya lalu dihitung nilai max dan minya.

   Output:

```
#Cek nilai NaN pada dataset
data.isna().sum()

#Isi nilai NaN dengan nilai rata-rata dari kolom yang bersangkutan
imputer = SimpleImputer(strategy='mean')
data[fitur] = imputer.fit_transform(data[fitur])
print("Sebelum Normalisasi")
data[fitur]
```

Sebelum Normalisasi

	Landsize	BuildingArea
0	126.000000	160.2564
1	202.000000	160.2564
2	156.000000	79.0000
3	0.000000	160.2564
4	134.000000	150.0000
...	...	...
34852	593.000000	160.2564
34853	98.000000	104.0000
34854	220.000000	120.0000
34855	593.598993	160.2564
34856	250.000000	103.0000

34857 rows × 2 columns

```
#Menghitung mean
means = data[fitur].mean(axis = 0)

# menghitung max - min
max_min = data[fitur].max(axis = 0) - data[fitur].min(axis = 0)

# menerapkan transformasi ke data
train_scaled = (data[fitur] - means) / max_min
```

```
#Tampilkan hasil
print("Setelah Normalisasi")
train_scaled
```

Setelah Normalisasi

	Landsize	BuildingArea
0	-0.001080	6.384749e-19
1	-0.000904	6.384749e-19
2	-0.001011	-1.825371e-03
3	-0.001371	6.384749e-19
4	-0.001061	-2.304032e-04
...	...	...
34852	-0.000001	6.384749e-19
34853	-0.001145	-1.263763e-03