

Data Preparation dari Sumber Open Source

*disusun untuk memenuhi tugas mata
kuliah Pembelajaran Mesin*

oleh:

Arif Maulana (2208107010067)

M. Nouval Rifqi (2208107010075)

Azzariyat Azra (2208107010079)

Tiara Agustin (2208107010004)



FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

2025

1. Data Description

Pada tahap ini, kita akan memahami karakteristik dasar dari dataset Car Price Prediction Challenge yang diambil dari Kaggle. Tahapan ini bertujuan untuk memberikan gambaran awal mengenai struktur data, jumlah sampel, jenis fitur yang tersedia, serta nilai-nilai yang ada dalam dataset. Data Description merupakan proses menjelaskan struktur dan karakteristik dataset yang akan digunakan dalam analisis atau pemodelan.

1) Nama Dataset dan Sumbernya

a) Nama Dataset : Car Price Prediction Challenge

b) Sumber Dataset :
<https://www.kaggle.com/datasets/deepcontractor/car-price-prediction-challenge/data>

c) Tujuan Dataset : Dataset ini digunakan untuk membangun model machine learning yang dapat memprediksi harga jual mobil berdasarkan fitur-fitur tertentu seperti tahun produksi, jarak tempuh, jenis bahan bakar, dan lainnya.

2) Deskripsi Singkat Dataset

Dataset ini berisi informasi mengenai mobil bekas yang dijual di pasar, termasuk harga jualnya serta berbagai spesifikasi kendaraan. Dengan dataset ini, kita dapat melakukan analisis eksploratif dan membangun model prediktif untuk menentukan harga jual mobil berdasarkan fitur-fiturnya.

a) Beberapa informasi penting yang terkandung dalam dataset ini meliputi:

- i) `selling_price` : Target utama yang ingin diprediksi.
- ii) `year` : Tahun kendaraan diproduksi, yang memengaruhi harga jual.
- iii) `km_driven` : Total kilometer yang sudah ditempuh kendaraan
- iv) `fuel` : Misalnya, Bensin, Diesel, CNG, dll
- v) `seller_type` : Apakah mobil dijual oleh individu atau dealer
- vi) `transmission` : Manual atau otomatis
- vii) `owner` : Seberapa banyak kendaraan ini berpindah tangan
- viii) `mileage` : Konsumsi bahan bakar per liter
- ix) `engine` : Ukuran mesin dalam CC (Cubic Centimeter)
- x) `max_power` : Tenaga yang dapat dikeluarkan mesin dalam satuan BHP (Brake Horse Power)

3) Jumlah Data (Sampel, Fitur, dan Label jika ada)

Untuk mengetahui jumlah sampel dan fitur dalam dataset, kita menggunakan fungsi *df.shape*

Dataset ini berisi informasi tentang 19.237 mobil dengan 18 fitur yang mencakup berbagai aspek seperti spesifikasi kendaraan, kondisi, dan faktor lain yang dapat memengaruhi harga jualnya. Jika terdapat kolom yang menunjukkan harga mobil, maka kolom tersebut dapat digunakan sebagai label (target) dalam model prediksi, sementara fitur lainnya berperan sebagai variabel independen.

4) Format Data

Dataset ini tersedia dalam format CSV (Comma-Separated Values). CSV adalah format yang umum digunakan dalam analisis data karena mudah diakses menggunakan berbagai pustaka Python seperti Pandas.

2. Data Loading

Dalam proses Data Loading, dataset dimuat ke dalam lingkungan pemrograman menggunakan Pandas, yang merupakan library populer untuk manipulasi dan analisis data dalam Python. Sebelum memuat dataset, kita perlu memastikan bahwa Google Drive sudah terhubung ke Google Colab agar bisa mengakses file yang disimpan di dalamnya. Hal ini dilakukan menggunakan perintah berikut :

```
from google.colab import drive  
  
drive.mount('/content/drive')
```

Kode di atas melakukan mounting Google Drive, sehingga kita bisa mengakses file dalam Google Drive melalui path `/content/drive/`. Setelah menjalankan perintah ini, pengguna akan diminta untuk memberikan izin akses ke Google Drive. Setelah Google Drive terhubung, kita bisa membaca dataset yang tersimpan dalam format CSV menggunakan Pandas

```
import pandas as pd  
  
# Memuat dataset ke dalam DataFrame  
df = pd.read_csv('/content/drive/My  
Drive/Tugas1_ML/1/car_price_prediction.csv  

```

Pertama, library Pandas diimpor agar dapat digunakan dalam proses manipulasi data. Selanjutnya, fungsi `pd.read_csv()` digunakan untuk membaca file CSV dari lokasi penyimpanan di Google Drive. Hasil dari proses ini disimpan dalam variabel `df`, yang merupakan objek `DataFrame` yang berisi seluruh data dari file tersebut. Setelah data berhasil dimuat, fungsi `df.head()` digunakan untuk menampilkan lima baris pertama dari dataset, sehingga kita bisa memverifikasi bahwa data telah dimuat dengan benar. Jika data berhasil muncul dalam bentuk tabel, itu berarti proses pemuatan dataset telah berhasil.

3. Data Understanding

Pada tahap Data Understanding, tujuan utamanya adalah memahami karakteristik dataset, termasuk tipe data, distribusi nilai, keberadaan nilai yang hilang, serta pola atau anomali dalam data. Langkah ini penting karena akan mempengaruhi bagaimana data diproses lebih lanjut dalam tahapan Data Preparation dan analisis model Machine Learning. Salah satu cara untuk memahami struktur dataset adalah dengan menggunakan perintah `df.head()`

`df.head()`

	ID	Price	Levy	Manufacturer	Model	Prod. year	Category	Leather interior	Fuel type	Engine volume	Mileage	Cylinders	Gear box type	Drive wheels	Doors	Wheel	Color	Airbags
0	45654403	13328	1399	LEXUS	RX 450	2010	Jeep	Yes	Hybrid	3.5	186005 km	6.0	Automatic	4x4	04-May	Left wheel	Silver	12
1	44731507	16621	1018	CHEVROLET	Equinox	2011	Jeep	No	Petrol	3	192000 km	6.0	Tiptronic	4x4	04-May	Left wheel	Black	8
2	45774419	8467	-	HONDA	FIT	2006	Hatchback	No	Petrol	1.3	200000 km	4.0	Variator	Front	04-May	Right-hand drive	Black	2
3	45769185	3607	862	FORD	Escape	2011	Jeep	Yes	Hybrid	2.5	168966 km	4.0	Automatic	4x4	04-May	Left wheel	White	0
4	45809263	11726	446	HONDA	FIT	2014	Hatchback	Yes	Petrol	1.3	91901 km	4.0	Automatic	Front	04-May	Left wheel	Silver	4

Gambar 3.1 Output dari `df.head()`

Setiap baris mewakili satu kendaraan, dan setiap kolom merepresentasikan fitur tertentu. Kolom **ID** adalah identifikasi unik untuk setiap kendaraan. **Price** menunjukkan harga kendaraan dalam satuan yang belum disebutkan. **Levy** tampaknya menunjukkan semacam pajak atau biaya tambahan, meskipun ada tanda "-" pada beberapa baris, yang bisa mengindikasikan nilai yang hilang atau belum terisi.

```
df['Levy'] = df['Levy'].replace('-', 0)
df['Levy'] = df['Levy'].astype(float)
```

Dalam tahap ini, kita menangani kolom **Levy**, yang sebelumnya berisi tanda "-" sebagai nilai yang tidak terdefinisi.

Langkah pertama adalah mengganti semua nilai "-" dengan 0, yang merupakan pendekatan umum ketika data yang hilang tidak dapat diisi dengan nilai rata-rata atau median yang lebih representatif. Ini bisa bermakna bahwa kendaraan tertentu memang tidak memiliki biaya tambahan atau pajak yang berlaku.

Setelah itu, kita mengonversi tipe data **Levy** menjadi *float*, karena sebelumnya nilai-nilainya kemungkinan dalam bentuk *string* akibat adanya karakter "-". Dengan konversi ini, kita memastikan bahwa kolom **Levy** dapat digunakan dalam analisis numerik atau model prediktif tanpa masalah tipe data.

Langkah ini penting untuk menjaga konsistensi data dan memastikan bahwa semua fitur numerik dapat digunakan dalam proses perhitungan lebih lanjut, seperti normalisasi atau analisis statistik.

`df.describe()`

	ID	Price	Prod. year	Cylinders	Airbags
count	1.923700e+04	1.923700e+04	19237.000000	19237.000000	19237.000000
mean	4.557654e+07	1.855593e+04	2010.912824	4.582991	6.582627
std	9.365914e+05	1.905813e+05	5.668673	1.199933	4.320168
min	2.074688e+07	1.000000e+00	1939.000000	1.000000	0.000000
25%	4.569837e+07	5.331000e+03	2009.000000	4.000000	4.000000
50%	4.577231e+07	1.317200e+04	2012.000000	4.000000	6.000000
75%	4.580204e+07	2.207500e+04	2015.000000	4.000000	12.000000
max	4.581665e+07	2.630750e+07	2020.000000	16.000000	16.000000

Gambar 3.2 Output dari `df.describe()`

Berdasarkan hasil `df.describe()`, kita dapat melihat ringkasan statistik dari dataset, termasuk jumlah data (`count`), rata-rata (`mean`), standar deviasi (`std`), nilai minimum (`min`), persentil 25% hingga 75%, serta nilai maksimum (`max`).

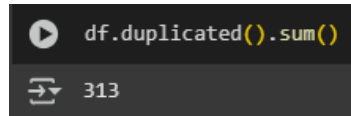
Dari tabel ini, terlihat bahwa tidak ada missing values dalam kolom yang ditampilkan, karena semua kolom memiliki jumlah data yang sama, yaitu **19.237**. Dengan demikian, kita tidak perlu melakukan imputasi atau penghapusan data terkait missing values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19237 entries, 0 to 19236
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                     19237 non-null  int64
1   Price                  19237 non-null  int64
2   Levy                   19237 non-null  object
3   Manufacturer           19237 non-null  object
4   Model                  19237 non-null  object
5   Prod. year             19237 non-null  int64
6   Category               19237 non-null  object
7   Leather interior       19237 non-null  object
8   Fuel type              19237 non-null  object
9   Engine volume          19237 non-null  object
10  Mileage                 19237 non-null  object
11  Cylinders               19237 non-null  float64
12  Gear box type           19237 non-null  object
13  Drive wheels            19237 non-null  object
14  Doors                  19237 non-null  object
15  Wheel                   19237 non-null  object
16  Color                   19237 non-null  object
17  Airbags                 19237 non-null  int64
dtypes: float64(1), int64(4), object(13)
memory usage: 2.6+ MB
```

Gambar 3.3 Output dari `df.info()`

Hasil dari perintah `df.info()` yang ditampilkan pada gambar menunjukkan informasi mengenai struktur dataset, termasuk jumlah total entri, jumlah non-null untuk setiap kolom, serta tipe data yang digunakan dalam setiap kolom. Dataset ini memiliki total 19.237 entri, yang berarti ada 19.237 baris data dengan 18 kolom berbeda.

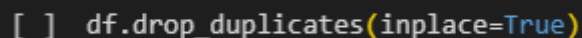
Setiap kolom dalam dataset ini tidak memiliki nilai null, seperti yang ditunjukkan oleh jumlah non-null yang sama dengan jumlah total entri. Hal ini mengindikasikan bahwa tidak ada data yang hilang dalam dataset ini, yang merupakan kondisi yang ideal untuk analisis lebih lanjut tanpa perlu menangani nilai yang hilang.



```
df.duplicated().sum()
313
```

Gambar 3.4 Data duplikat

Dari hasil `df.duplicated().sum()`, kita melihat bahwa terdapat **313 baris duplikat** dalam dataset. Keberadaan data duplikat bisa menyebabkan bias dalam analisis dan pelatihan model, karena informasi yang sama muncul lebih dari sekali, yang dapat mengarah pada hasil yang tidak akurat. Oleh karena itu, langkah selanjutnya adalah **menghapus baris yang duplikat** menggunakan `df.drop_duplicates(inplace=True)`, agar dataset menjadi lebih bersih dan siap untuk analisis lebih lanjut.



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

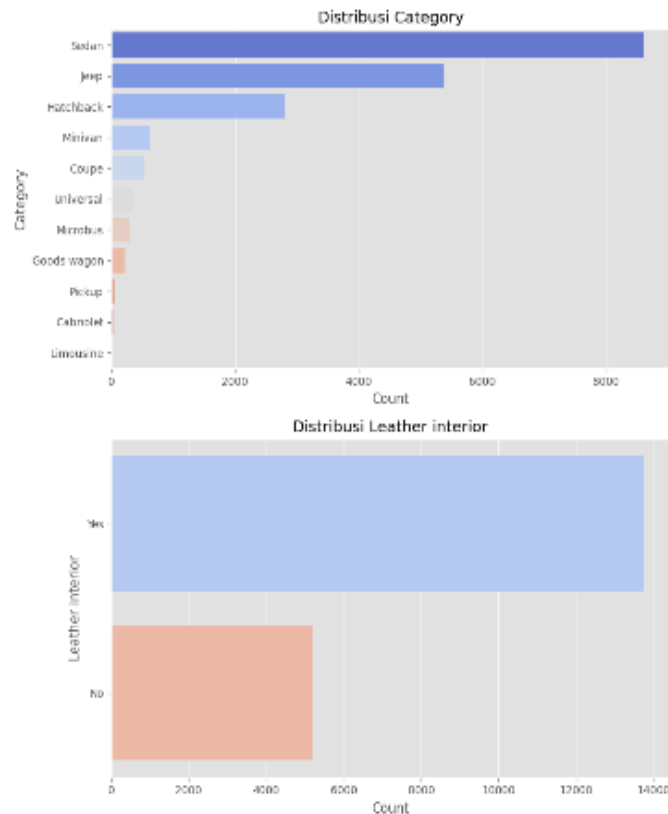
Gambar 3.5 Menghapus baris duplikat

```
numerical_columns = df.select_dtypes(include=['int64',
'float64']).columns
categorical_columns =
df.select_dtypes(include=['object']).columns
```

Kode ini bertujuan untuk mengelompokkan kolom dalam dataset berdasarkan tipe datanya. Kolom numerik dipilih menggunakan `df.select_dtypes(include=['int64', 'float64']).columns`, yang menyaring kolom bertipe integer dan float. Ini berguna untuk analisis statistik atau pemrosesan lebih lanjut.

Sementara itu, kolom kategorikal diidentifikasi dengan `df.select_dtypes(include=['object']).columns`, yang mengambil kolom bertipe string. Kolom ini biasanya perlu diubah ke bentuk numerik melalui encoding sebelum digunakan dalam model machine learning. Dengan pemisahan ini, kita dapat menangani data dengan lebih mudah, baik dalam tahap eksplorasi, preprocessing.

- Distribusi kategori kendaraan



Gambar 3.6 Distribusi dari kolom categorical

Dari grafik distribusi kategori kendaraan, terlihat bahwa jenis kendaraan yang paling dominan adalah Sedan, disusul oleh Jeep dan Hatchback. Ini menunjukkan bahwa mayoritas kendaraan dalam dataset adalah sedan, yang kemungkinan besar lebih populer di pasar dibandingkan jenis kendaraan lainnya.

Di sisi lain, kategori seperti Limousine, Cabriolet, dan Pickup memiliki jumlah yang jauh lebih sedikit, yang mengindikasikan bahwa kendaraan-kendaraan ini mungkin kurang umum atau lebih spesifik dalam penggunaannya.

Secara keseluruhan, pola ini menunjukkan preferensi pasar terhadap kendaraan yang lebih umum digunakan untuk keperluan sehari-hari, seperti sedan dan hatchback, dibandingkan kendaraan khusus seperti limousine atau pickup.

4. Data Preparation

a. Encoding Categorical

```
encoder = LabelEncoder()
for column in categorical_columns:
    df[column] = encoder.fit_transform(df[column])
df.sample(5)
```

	Price	Levy	Manufacturer	Model	Prod. year	Category	Leather interior	Fuel type	Engine volume	Mileage	Cylinders	Gear box type	Drive wheels	Doors	Wheel	Color	Airbags
7632	49410.3	1017.0	23	1334	2017	9	1	5	2.0	30930.0	4	0	1	1	0	14	4
3950	49410.3	1327.0	58	435	2017	9	1	5	2.5	6800.0	6	0	1	1	0	1	8
3250	314.0	0.0	5	1533	2008	4	1	5	3.0	240000.0	6	2	0	1	0	1	8
9427	15053.0	0.0	5	1533	2000	4	1	5	4.0	250000.0	8	2	0	1	0	1	4
16189	16900.0	503.0	58	262	2012	3	1	5	1.5	46756.0	4	0	1	1	0	1	4

Gambar 4.1 Output dari Encoding Categorical

Pada tahap ini, dilakukan proses Label Encoding terhadap kolom-kolom kategorikal. Label Encoding mengonversi nilai kategori menjadi angka numerik. Misalnya, jika suatu kolom memiliki kategori seperti ["Sedan", "SUV", "Hatchback"], maka setelah encoding, kategorinya bisa berubah menjadi [0, 1, 2].

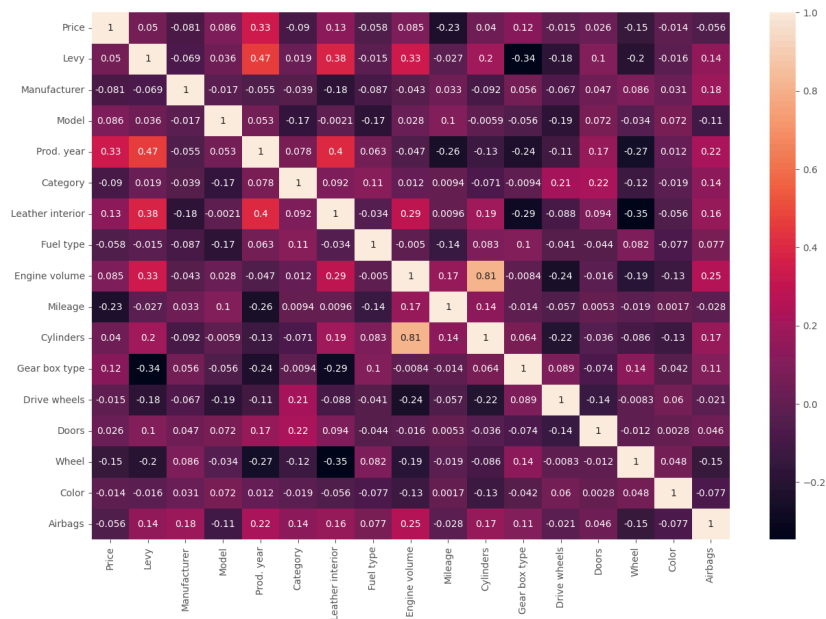
Langkah ini diambil karena sebagian besar model Machine Learning tidak dapat bekerja langsung dengan data kategorikal dalam bentuk string. Dengan mengonversinya ke bentuk numerik, model dapat lebih mudah memahami dan memproses informasi.

Pada langkah sebelumnya, semua kolom kategorikal diubah menjadi angka menggunakan Label Encoding. Ini menyebabkan setiap kategori direpresentasikan dengan angka, misalnya:

- **Fuel type** awalnya bisa berupa kategori seperti "Petrol", "Diesel", "Hybrid", dan sekarang telah diubah menjadi angka (**misalnya 1, 2, 3, dst.**).
- **Gear box type** mungkin awalnya berupa teks seperti "Manual", "Automatic", dan sekarang telah dikonversi menjadi angka (**misalnya 0 dan 1**).

Keuntungan dari pendekatan ini adalah model dapat bekerja dengan lebih baik tanpa harus menangani data dalam bentuk string. Namun, kelemahannya adalah jika kategori tidak memiliki urutan (nominal), Label Encoding dapat menyebabkan model salah menginterpretasikan hubungan antara angka yang diberikan.

```
plt.figure(figsize=(15, 10))
sns.heatmap(corr, annot=True)
plt.show()
```

Gambar 4.1 Heatmap

Berdasarkan heatmap korelasi di atas, kolom 'Color', 'Doors', 'Drive wheels', 'Fuel type', dan 'Airbags' memiliki korelasi yang sangat rendah terhadap harga mobil (Price). Korelasi yang rendah menunjukkan bahwa variabel-variabel ini tidak memiliki pengaruh yang signifikan terhadap target yang ingin kita prediksi.

Oleh karena itu, untuk menyederhanakan model dan menghindari fitur yang tidak memberikan banyak informasi, kita menghapus kolom-kolom tersebut dari dataset. Hal ini juga membantu mengurangi kompleksitas model dan meningkatkan efisiensi pelatihan tanpa kehilangan informasi yang relevan.

```
# Daftar kolom yang akan dihapus
columns_to_drop = ['Color', 'Doors', 'Drive wheels', 'Fuel type',
'Airbags']

# Menghapus kolom dari DataFrame
df.drop(columns=columns_to_drop, axis=1, inplace=True)

# Menampilkan DataFrame setelah penghapusan
df.head()
```

	Price	Levy	Manufacturer	Model	Category	Leather interior	Engine volume	Mileage	Cylinders	Gear box type	Wheel	Age
0	13328.0	1399.0	32	1242	4	1	3.5	186005.0	6	0	0	15
1	16621.0	1018.0	8	658	4	0	3.0	192000.0	6	2	0	14
2	8467.0	0.0	21	684	3	0	1.4	200000.0	4	3	1	19
3	3607.0	862.0	16	661	4	1	2.5	168966.0	4	0	0	14
4	11726.0	446.0	21	684	3	1	1.4	91901.0	4	0	0	11

Gambar 4.2 Dataframe setelah penghapusan

Dapat disimpulkan bahwa beberapa fitur kategorikal telah dihapus, yaitu Color, Doors, Drive wheels, Fuel type, dan Airbags. Penghapusan ini kemungkinan dilakukan karena fitur-fitur tersebut dianggap kurang relevan, memiliki korelasi rendah dengan target prediksi, atau dapat menyebabkan redundansi dalam analisis.

Sekarang DataFrame lebih fokus pada fitur numerik dan beberapa fitur kategorikal yang sudah dikodekan secara numerik, seperti Leather interior, Gear box type, dan Wheel. Ini akan memudahkan proses pemodelan, terutama jika model yang digunakan adalah model berbasis angka seperti regresi atau jaringan saraf.

b. Feature Extraction

```
now_year = dt.datetime.now().year
df['Age'] = now_year - df['Prod. year']
df.drop('Prod. year', axis=1, inplace=True)
```

Kode ini digunakan untuk menghitung usia (Age) dari kendaraan berdasarkan tahun produksinya dan kemudian menghapus kolom Prod. year setelah konversi. Berikut adalah penjelasan setiap langkahnya:

1. `now_year = dt.datetime.now().year`
 - Baris ini mengambil tahun saat ini secara dinamis menggunakan `datetime` agar tidak perlu diperbarui secara manual setiap tahun.
2. `df['Age'] = now_year - df['Prod. year']`
 - Baris ini membuat kolom baru bernama Age, yang dihitung sebagai selisih antara tahun sekarang dengan tahun produksi kendaraan.
 - Dengan ini, kita bisa mendapatkan usia kendaraan dalam satuan tahun.
3. `df.drop('Prod. year', axis=1, inplace=True)`
 - Setelah usia kendaraan dihitung, kolom Prod. year dihapus dari dataset karena informasi yang dikandungnya sudah diwakili oleh Age.
 - `axis=1` menunjukkan bahwa kita menghapus kolom (bukan baris).
 - `inplace=True` berarti perubahan dilakukan langsung pada DataFrame tanpa perlu menyimpan ke variabel baru.

Alasan melakukan tahapan ini

- Usia kendaraan lebih relevan dibandingkan tahun produksi. Model machine learning lebih mudah memahami hubungan data dalam bentuk numerik yang lebih bermakna, seperti "usia kendaraan" daripada "tahun produksi".
- Mengurangi redundansi fitur. Setelah kolom Prod. year diubah menjadi Age, kolom lama tidak lagi diperlukan.
- Model lebih generalisasi. Jika kita tetap menggunakan tahun produksi, nilai tersebut terus berubah setiap tahun, yang dapat menyebabkan model kesulitan menangkap pola

yang stabil. Dengan mengonversinya menjadi usia, kita memperoleh fitur yang lebih statis dan interpretatif.

c. Clipping

```
# Menentukan batas bawah dan atas berdasarkan persentil
lowerBound = df[numerical_columns].quantile(0.05) # Persentil 5%
upperBound = df[numerical_columns].quantile(0.95) # Persentil 95%

# Menerapkan clipping pada setiap kolom numerik
df[numerical_columns] = df[numerical_columns].apply(lambda x:
x.clip(lower=lowerBound[x.name], upper=upperBound[x.name]))
```

Pada tahap ini, dilakukan penentuan batas bawah dan atas untuk nilai-nilai dalam kolom numerik menggunakan metode persentil. Batas bawah ditetapkan pada persentil ke-5, sedangkan batas atas pada persentil ke-95. Tujuan dari langkah ini adalah untuk menangani outlier dengan cara yang lebih fleksibel dibandingkan dengan metode penghapusan langsung.

Setelah batas atas dan bawah ditentukan, dilakukan *clipping*, yaitu membatasi nilai-nilai yang berada di luar rentang tersebut agar tidak melewati ambang yang telah ditentukan. Jika suatu nilai lebih kecil dari persentil ke-5, maka nilai tersebut akan diubah menjadi nilai pada persentil ke-5. Begitu juga sebaliknya, jika suatu nilai lebih besar dari persentil ke-95, maka akan dipotong pada nilai persentil ke-95.

Langkah ini penting karena keberadaan outlier dapat mempengaruhi hasil analisis dan performa model Machine Learning, terutama model berbasis regresi atau algoritma yang sensitif terhadap skala data. Dengan menggunakan metode clipping ini, data tetap dipertahankan tetapi tidak terlalu dipengaruhi oleh nilai ekstrem yang bisa mengganggu analisis lebih lanjut.