LAPORAN PROJECT BASED PEMBELAJARAN MESIN

MEMPREDIKSI TINGKAT KEHEMATAN BAHAN BAKAR KENDARAAN DENGAN METODE *BAGGING*

Disusun untuk Memenuhi Project Based Mata Kuliah Pembelajaran Mesin

Disusun Oleh:

Hilman Taris Muttaqin	1301204208
Khalilullah Al Faath	1301204376
Muhammad Erlangga Arsadi	1301204346
Naufal Abdurrahman Burhani	1301204008

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

2022/2023

DAFTAR ISI

1. FORMULASI MASALAH

Terdapat dataset autos_mpg.csv yang berisikan data-data dari banyak kendaraan serta konsumsi bahan bakarnya. Lalu akan dilakukan pula regresi pada dataset, yaitu memprediksi tingkat kehematan bahan bakar dari kendaraan yang ada berdasarkan profil kendaraan yang diberikan yang diwakili oleh beberapa atribut seperti silinder, daya (tenaga kuda), tahun keluaran, dll. Untuk keperluan regresi pada dataset, metode *ensemble* yang akan digunakan adalah metode *bagging*. Jadi, masalah inti yang akan kami selesaikan pada project based kali ini adalah bagaimana cara memprediksi tingkat kehematan bahan bakar kendaraan dengan metode *bagging*.

2. EKSPLORASI DAN PRA-PEMROSESAN DATA

2.1. Dataset

Pada project based kali ini, dataset yang akan digunakan adalah dataset autos MPG yang berisikan data kendaraan serta konsumsi bahan bakarnya. Dataset yang sudah disebutkan sebelumnya dapat diakses dari tautan berikut.

https://github.com/khalilullahalfaath/Project_Based_ML/blob/fa1ad703e8ad687e46747bbfbd8e5cef6270aa11/autos_mpg.csv?raw=true

Pada dataset autos MPG terdapat beberapa atribut yang digunakan, diantaranya adalah sebagai berikut:

Atribut	Tipe	Keterangan
mpg	continuous (target attribute)	Miles per gallon, konsumsi bahan bakar per mil
cylinders	multi-valued discrete	Banyaknya silinder pada mobil dari 4 sampai 8
displacement	continuous	Inci kubik (cu. In.) atau sentimeter kubik (cc) volume yang digantikan atau berapa banyak udara yang dipindahkan oleh semua piston
horsepower	continuous	Tenaga kuda mesin
weight	continuous	Berat mobil dalam pon
acceleration	continuous	Waktu yang dibutuhkan untuk akselerasi dari 0 sampai 60 mph (dalam detik)
model_year	multi-valued discrete	Tahun model (modulo 100)
origin	multi-valued discrete	Asal dari mobil {1: Amerika, 2: Eropa, 3: Jepang}
car_name	string (unique for each instance)	Nama kendaraan

Berikut adalah tampilan dari isi dataset yang ditampilkan menggunakan bentuk tabel beserta index dari masing-masing row tabel.

[3] df

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

Dataset autos MPG

Dapat dilihat bahwa, sepertinya dataset ini terurut berdasarkan kolom model_yearnya. Kita akan mengeceknya dengan fungsi .equals yang tersedia di pandas.

```
df_terurut = df.sort_values(by=["model_year"])

df.equals(df_terurut)
```

```
False
```

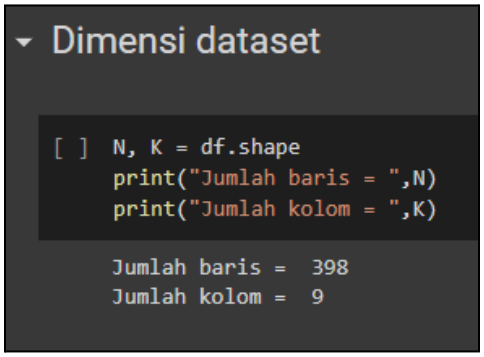
Pengecekan Dataset

Setelah dilakukannya pengecekan dengan fungsi .equals, ternyata asumsi tersebut keliru dan dataset tidak terurut berdasarkan model_year.

2.2. Data Understanding

2.2.1. Dimensi Dataset

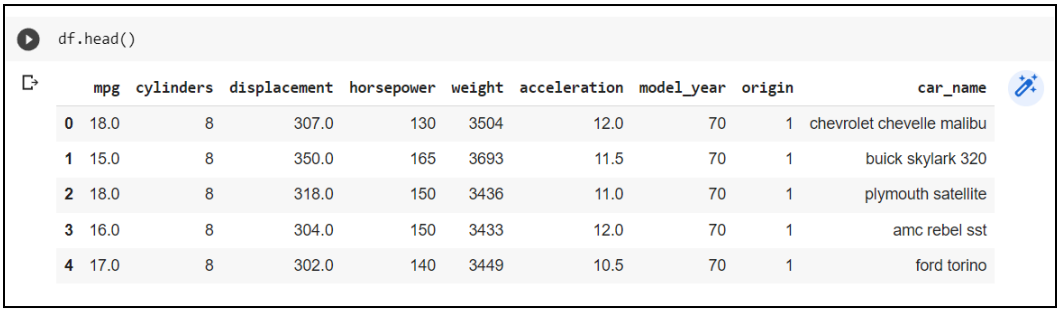
Seperti yang dapat kita lihat di gambar, data memiliki record sebanyak 398 baris dan jumlah kolom sebanyak 9 kolom.



Dimensi dari Dataset

2.2.2. Lima data terawal

Berikut adalah lima data terawal dari dataset.

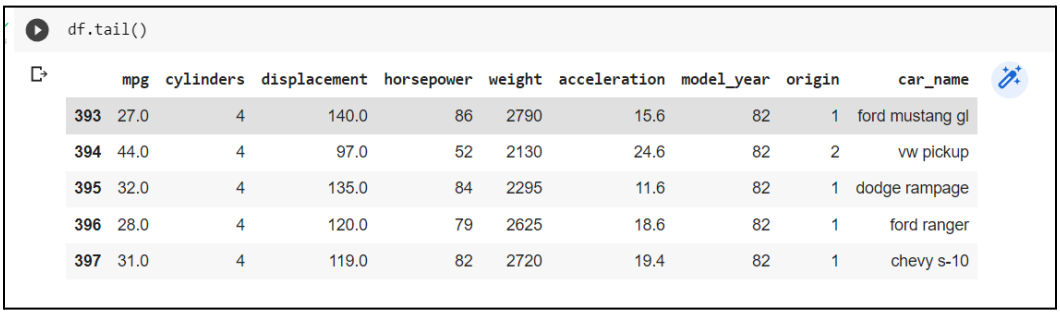


	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Lima data terawal dari Dataset

2.2.3. Lima data terakhir

Berikut adalah lima data terakhir dari dataset.



	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

Lima data terakhir dari Dataset

2.2.4. Lima data random

Berikut adalah lima data yang dipilih secara random.

```
[7] df.sample(5)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
284	20.6	6	225.0	110	3360	16.6	79	1	dodge aspen 6
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii
186	27.0	4	101.0	83	2202	15.3	76	2	renault 12tl
264	18.1	8	302.0	139	3205	11.2	78	1	ford futura
323	27.9	4	156.0	105	2800	14.4	80	1	dodge colt

Lima data random dari Dataset

2.2.5. Analisis Deskriptif pada Dataset

Pada gambar di bawah ini, kita bisa melihat sekilas terkait gambaran besar dari dataset yang digunakan pada tugas kali ini.

Deskripsi terkait dataset

```
[ ] df.describe()
```

	mpg	cylinders	displacement	weight	acceleration	model_year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

Deskripsi terkait Dataset

Dapat dilihat bahwa:

- Banyak dari record untuk setiap atribut jumlahnya sama.
- Kolom weight memiliki persebaran data yang terbesar.
- Kebanyakan data memiliki kemencengan atau tidak terdistribusi secara normal.

2.2.6. Tipe Data setiap Kolom dari Dataset

Berikut adalah tipe data setiap kolom dari dataset.

Tipe setiap kolom dari dataset

```
[9]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 398 entries, 0 to 397
     Data columns (total 9 columns):
      #   Column        Non-Null Count  Dtype
     ---  ------        --------------  -----
      0   mpg           398 non-null    float64
      1   cylinders     398 non-null    int64
      2   displacement  398 non-null    float64
      3   horsepower    398 non-null    object
      4   weight        398 non-null    int64
      5   acceleration  398 non-null    float64
      6   model_year    398 non-null    int64
      7   origin        398 non-null    int64
      8   car_name      398 non-null    object
     dtypes: float64(3), int64(4), object(2)
     memory usage: 28.1+ KB
```

Tipe dari setiap kolom pada Dataset

Analisis terkait tipe data setiap kolom dari dataset dapat diuraikan sebagai berikut:

- Tidak terdeteksi adanya nilai yang NULL dalam bentuk np.nan.
- Mpg, displacement, acceleration bertipe data float (kontinu).
- Cylinders, acceleration, model_year, origin bertipe integer.
- Cylinders, model_year, origin walaupun bentuknya angka, tetapi sebenarnya merupakan data kategorikal.
- Horsepower karena seharusnya bertipe numerik, di sini terdeteksi bertipe data object. Menandakan bahwa ada data yang salah di sini.

2.3. Pra-Pemprosesan Data

Tindakan selanjutnya yang kami lakukan adalah pra-pemrosesan data setelah mempelajari kondisi data yang disediakan. Pembersihan data dan transformasi data adalah dua contoh dari beberapa bentuk pra-pemrosesan data yang dilakukan dalam prosedur ini. Prosedur ini digunakan untuk membersihkan data dan menghilangkan gangguan, ketidakkonsistenan, dan informasi yang tidak lengkap.

2.3.1. Pembersihan Data

2.3.1.1. Data NULL

Penanganan data yang hilang atau null penting karena banyak algoritma machine learning yang tidak bisa menangani nilai null dan akan menghasilkan error jika nilai null ada dalam data. Selain itu, nilai null bisa mempengaruhi hasil analisis dan model jika tidak ditangani dengan benar.

Penting juga untuk menangani nilai null karena mereka bisa menunjukkan data yang hilang atau salah. Jika Anda tidak menangani nilai null, Anda mungkin akan mencakup data yang salah atau bias dalam analisis Anda, yang bisa mengarah pada kesimpulan yang salah.

Ada beberapa cara untuk menangani nilai null dalam sebuah dataset, tergantung pada konteks dan jumlah data null. Beberapa metode yang umum digunakan adalah:

- Menghapus baris atau kolom dengan nilai null: Ini adalah metode yang sederhana, tetapi bisa menjadi masalah jika nilai null tidak terdistribusi secara acak atau jika data NULL terlalu besar maka akan membuat jumlah data lebih sedikit dan mempengaruhi akurasi.
- Mengganti nilai null dengan nilai placeholder: Ini bisa berguna jika ingin menyimpan nilai null dalam data, tapi perlu menggantinya dengan nilai yang bisa digunakan oleh algoritma machine learning, seperti mean, median, atau modus.
- Interpolasi nilai null: Ini melibatkan estimasi nilai yang hilang berdasarkan nilai yang ada dalam data. Ini bisa dilakukan dengan menggunakan teknik seperti interpolasi linier atau interpolasi spline.

Langkah-langkah:

a. Mencari data NULL

```
print(df.isna().sum())

mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

Pencarian Data NULL

Dapat dilihat bahwa data NULL tidak terdeteksi dengan isna(), artinya data NULL tidak dalam bentuk np.nan. Padahal, kita mencurigai bahwa terdapat data NULL di dataset berdasarkan kolom horsepower yang tidak sesuai dengan tipe data yamg seharusnya. Oleh karena itu, kita mencari data NULL dengan "?"

```
[11] df.columns[df.isin(['?']).any()]

     Index(['horsepower'], dtype='object')

data NULL ada di kolom horsepower

[12] df.replace("?",np.nan,inplace=True)

     print(df.isna().sum())

mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

Pencarian Data NULL dengan "?"

Didapatkan bahwa data NULL dalam bentuk "?" terdapat pada kolom horsepower. Kita kemudian mengubahnya menjadi np.nan untuk dihandling selanjutnya.

b. Handling data NULL

Menurut sumber yang kami dapatkan dari situs https://www.naukri.com/learning/articles/handling-missing-data-mean-median-mode/, untuk mengimpute nilai data NULL memiliki ketentuan sebagai berikut:

- Gunakan nilai mean, jika datanya numerik dan tidak menceng (terdistribusi normal).
- Gunakan nilai median, jika datanya numerik dan menceng.
- Gunakan nilai modus, jika datanya objek (string) atau numerik, selain dari yang di atas.

Oleh karena itu, kita akan mengecek *skewness* dari datanya terlebih dahulu.

c. Buat df temporary yang hanya berisi data horsepower untuk mempermudah perhitungan kemencengan.

```
[14] temp_df = df['horsepower'].copy().dropna().astype(int).to_numpy()

temp_df

array([130, 165, 150, 150, 140, 198, 220, 215, 225, 190, 170, 160, 150,
       225,  95,  95,  97,  85,  88,  46,  87,  90,  95, 113,  90, 215,
       200, 210, 193,  88,  90,  95, 100, 105, 100,  88, 100, 165, 175,
       153, 150, 180, 170, 175, 110,  72, 100,  88,  86,  90,  70,  76,
        65,  69,  60,  70,  95,  80,  54,  90,  86, 165, 175, 150, 153,
       150, 208, 155, 160, 190,  97, 150, 130, 140, 150, 112,  76,  87,
        69,  86,  92,  97,  80,  88, 175, 150, 145, 137, 150, 198, 150,
       158, 150, 215, 225, 175, 105, 100, 100,  88,  95,  46, 150, 167,
       170, 180, 100,  88,  72,  94,  90,  85, 107,  90, 145, 230,  49,
        75,  91, 112, 150, 110, 122, 180,  95, 100, 100,  67,  80,  65,
        75, 100, 110, 105, 140, 150, 150, 140, 150,  83,  67,  78,  52,
        61,  75,  75,  75,  97,  93,  67,  95, 105,  72,  72, 170, 145,
       150, 148, 110, 105, 110,  95, 110, 110, 129,  75,  83, 100,  78,
        96,  71,  97,  97,  70,  90,  95,  88,  98, 115,  53,  86,  81,
        92,  79,  83, 140, 150, 120, 152, 100, 105,  81,  90,  52,  60,
        70,  53, 100,  78, 110,  95,  71,  70,  75,  72, 102, 150,  88,
       108, 120, 180, 145, 130, 150,  68,  80,  58,  96,  70, 145, 110,
       145, 130, 110, 105, 100,  98, 180, 170, 190, 149,  78,  88,  75,
        89,  63,  83,  67,  78,  97, 110, 110,  48,  66,  52,  70,  60,
       110, 140, 139, 105,  95,  85,  88, 100,  90, 105,  85, 110, 120,
       145, 165, 139, 140,  68,  95,  97,  75,  95, 105,  85,  97, 103,
       125, 115, 133,  71,  68, 115,  85,  88,  90, 110, 130, 129, 138,
       135, 155, 142, 125, 150,  71,  65,  80,  80,  77, 125,  71,  90,
        70,  70,  65,  69,  90, 115, 115,  90,  76,  60,  70,  65,  90,
```

Data horsepower

```
print(skew(temp_df))

1.0831611646869432
```

Kode Print skewness horsepower

Karena data horsepower adalah data numerik dan menceng (condong dan condongnya ke arah kanan) maka kita akan mengimpute data NULL dengan nilai median.

d. Mengembalikan tipe data dari horsepower ke float.

```
[17] col_num = ['horsepower']

[18] df = df.astype({'horsepower': 'float'})
```

Pengembalian tipe data horsepower ke float

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car_name      398 non-null    object
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

Hasil pengembalian tipe data horsepower ke float

```
[21] df['horsepower']

     0      130.0
     1      165.0
     2      150.0
     3      150.0
     4      140.0
            ...
     393     86.0
     394     52.0
     395     84.0
     396     79.0
     397     82.0
     Name: horsepower, Length: 398, dtype: float64
```

Data horsepower

e. Mengisi data NULL pada kolom horsepower dengan nilai mediannya.

```
imputer = SimpleImputer(strategy = 'median', missing_values = np.nan)
imputer.fit(df[col_num])
df[col_num] = imputer.transform(df[col_num])
```

```
[24] print(df.isna().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

Pengisian data NULL pada kolom horsepower

Data Null sudah terisi dengan nilai mediannya.

2.3.1.2. Data Duplikat

Duplikat data harus dihilangkan sebagai tahap awal dalam proses pembersihan data. Kualitas data untuk analisis dapat ditingkatkan dengan menghilangkan duplikasi data serta noise. Untuk mengecek data duplikat kita dapat menggunakan fungsi duplicated pada pandas. Tidak terdeteksi adanya duplikasi data.

```
[26] bool_series = df.duplicated()
     print(bool_series)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
393    False
394    False
395    False
396    False
397    False
Length: 398, dtype: bool
```

```
[27] print(type(bool_series))
     bool_series.value_counts()
```

```
<class 'pandas.core.series.Series'>
False    398
dtype: int64
```

Mendeteksi Duplikasi Data

2.3.1.3. Mengecek Data yang hanya memiliki banyak nila uniknya satu

Setelah memeriksa duplikasi data, kami menentukan apakah data mengandung varian sederhana atau tidak. Data tidak memiliki data yang diserap atau cenderung memiliki nilai yang sama jika varian kolom mendekati 0.

Hal ini perlu dilakukan karena berarti kolom tersebut tidak penting dalam bisa didrop karena tidak mempengaruhi kolom target.

```
df.var()
```

```
<ipython-input-28-28ded241fd7c>:1:
  df.var()
mpg                61.089611
cylinders           2.893415
displacement    10872.199152
horsepower       1460.969052
weight         717140.990526
acceleration        7.604848
model_year         13.672443
origin              0.643292
dtype: float64
```

Pengecekan nilai dari kolom

Ternyata tidak ada data yang memiliki nilai hanya 1 nilai. Sehingga tidak perlu ada handling dalam hal ini.

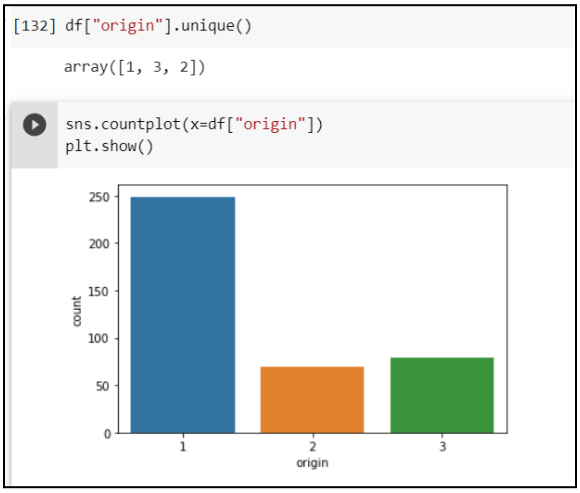
Untuk membuktikan bahwa tidak ada data yang hanya bernilai 1, kita bisa mencoba mengecek dengan kode ini.

```
col1len = []
for col in df:
    if (len(df[col].unique()) == 1):
      col1len.append(col)
print(col1len)
```

```
[]
```

Kode Pembuktian tidak ada nilai 1

Kita bisa mengecek origin yang memiliki variansi terkecil.



Pengecekan kolom origin

Dapat dilihat bahwa origin memiliki jenis data lebih dari satu.

2.4. Keseluruhan Alur Pra-Pemrosesan

Berikut adalah alur prapemrosesan yang kami lakukan:

1) Pembersihan data: Dilakukan handling data NULL, pengecekan nilai duplikat, mengecek atribut yang jenis nilainya hanya ada 1 tipe.
2) Karena data kategorikal sudah dalam bentuk numerik, maka tidak perlu ada encoding.

2.5. Eksplorasi Data Kategorikal

2.5.1. Pendefinisian Data Kategorikal

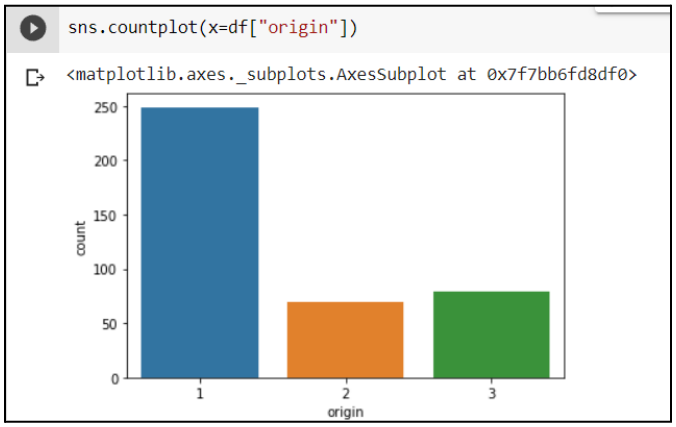
Data kategorik adalah data yang menjelaskan karakteristik dari data tersebut. Misalnya jenis kelamin, Bahasa, Kewarganegaraan, dan lain sebagainya. Data kategorik juga dapat menggunakan nilai numerik. Data kategorik dibagi menjadi dua yaitu data nominal dan data ordinal.

Dilihat dari dataset, setidaknya ada beberapa data kategorik di sini, yaitu origin, model_year, dan cylinders. Di sini akan digunakan dua plot, yaitu:

- Count plot adalah tipe plot yang sering digunakan dalam visualisasi data untuk menunjukkan jumlah atau frekuensi suatu kelas dalam data. Count plot biasanya menggunakan satu variabel kategorik yang diplot pada sumbu x dan satu variabel numerik yang diplot pada sumbu y. Tipe plot ini mirip dengan bar chart, namun biasanya lebih sederhana dan hanya menampilkan satu set data. Contoh penggunaan count plot adalah untuk menampilkan jumlah orang dalam setiap kelompok umur, jumlah produk yang terjual dalam setiap bulan, atau jumlah peserta dalam setiap kelas pelatihan. Dalam beberapa kasus, count plot juga bisa menampilkan distribusi data yang lebih detail dengan menggunakan histogram atau bar chart stacked.
- Pie chart adalah tipe diagram yang digunakan untuk menampilkan bagaimana persentase keseluruhan suatu variabel dibagi menjadi bagian-bagian yang lebih kecil. Pie chart terdiri dari sebuah lingkaran yang terbagi menjadi beberapa bagian atau "pie slices", masing-masing pie slice mewakili persentase dari keseluruhan. Pie chart sering digunakan untuk menampilkan data yang bersifat part-to-whole, di mana setiap bagian pie slice merupakan bagian dari keseluruhan.

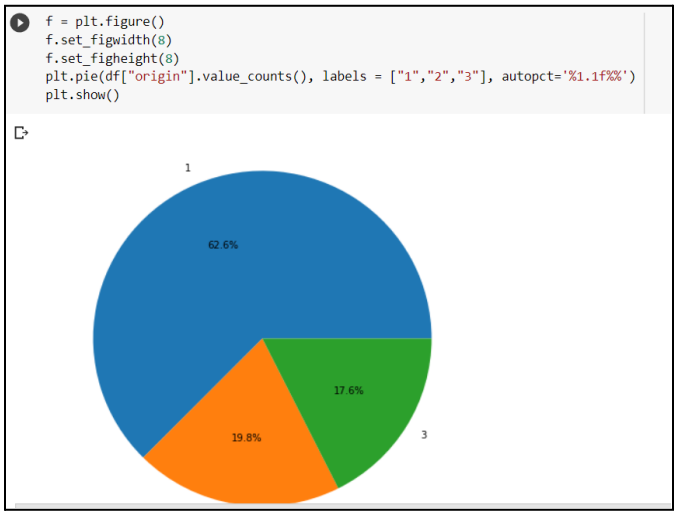
2.5.2. Kolom origin

2.5.2.1. Count plot



Count plot dari kolom origin

2.5.2.2. Pie chart

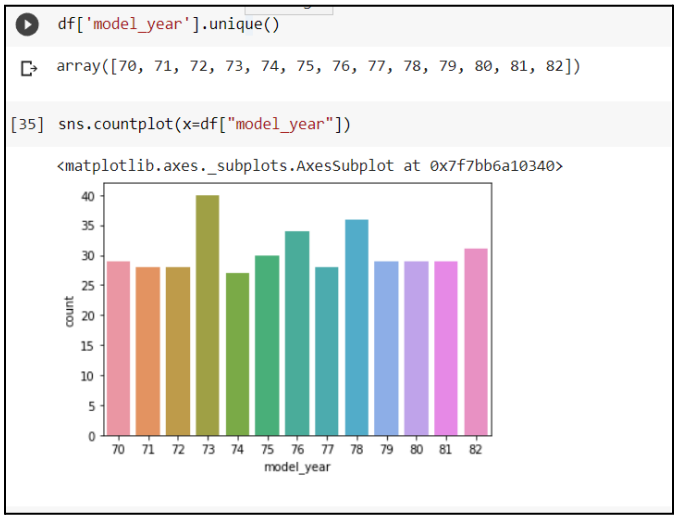


Pie chart dari kolom origin

Dapat dilihat bahwa modus dari data adalah nilai 1, dengan banyak 62%, sementara untuk 2 dan 3 lebih seimbang banyaknya dengan selisih yang sedikit.

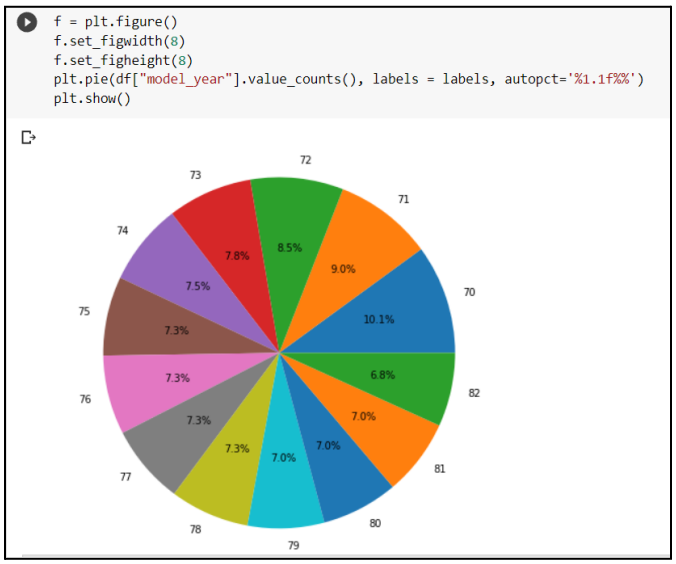
2.5.3. Kolom model_year

2.5.3.1. Count plot



Count plot kolom model_year

2.5.3.2. Pie chart



Pie chart kolom model_year

Dapat dilihat bahwa data ini seragam banyaknya untuk setiap model_year. Dengan nilai terbanyak adalah tahun 1973.

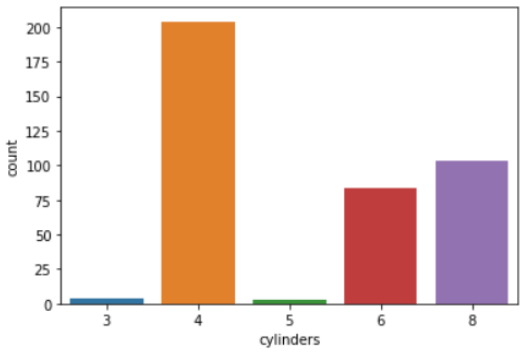
2.5.4. Kolom cylinders

2.5.4.1. Count plot

```
df['cylinders'].unique()
```

```
array([8, 4, 6, 3, 5])
```

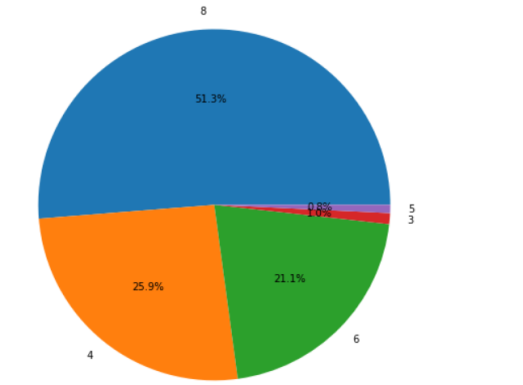
```
[39] sns.countplot(x=df["cylinders"])
     plt.show()
```



Count plot kolom cylinders

2.5.4.2. Pie chart

```
labels = [8, 4, 6, 3, 5]
f = plt.figure()
f.set_figwidth(8)
f.set_figheight(8)
plt.pie(df["cylinders"].value_counts(), labels = labels, autopct='%1.1f%%')
plt.show()
```



Pie chart kolom cylinders

Dapat dilihat bahwa kolom cylinders memiliki kemencengan data yang sangat besar. Setengah dari data terkumpul pada silender sebanyak 8.

2.6. Eksplorasi Data Numerik

2.6.1. Pendefinisian Data Numerik

Data numerik adalah data kuantitatif yang nilainya dalam bentuk numerik (angka). Data kategorik merupakan kumpulan kategori dan setiap nilai mewakili beberapa kategori, data kategorik disebut juga data kualitatif yang berbentuk tidak beraturan (Bhagat et al., 2013)

Data numerik yang digunakan di sini adalah:

```
col_numeric = ["mpg", "displacement", "horsepower", "weight", "acceleration"]
```

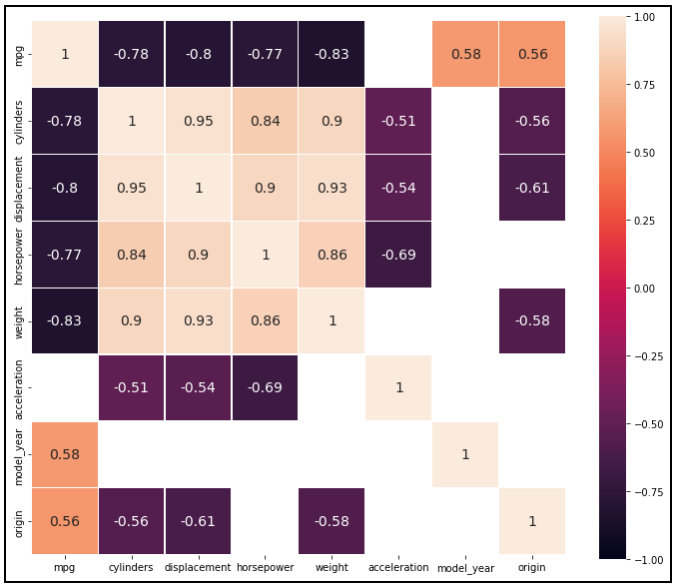
Data kolom numerik yang akan digunakan

2.6.2. Korelasi Antaratribut

Korelasi adalah hubungan atau asosiasi antara dua variabel atau lebih. Jika dua variabel terkorelasi, artinya perubahan pada salah satu variabel akan mempengaruhi perubahan pada variabel lainnya. Tingkat korelasi bisa negatif atau positif. Jika korelasi negatif, artinya ketika salah satu variabel meningkat, variabel lainnya akan menurun, dan sebaliknya. Jika korelasi positif, artinya ketika salah satu variabel meningkat, variabel lainnya juga akan meningkat. Korelasi bisa diukur dengan rumus korelasi yang disebut dengan koefisien korelasi. Ada beberapa jenis koefisien korelasi yang sering digunakan, di antaranya adalah koefisien korelasi Pearson, koefisien korelasi Spearman, dan koefisien korelasi Kendal.

```
corr = df.corr()
plt.figure(figsize=(12,10))
sns.heatmap(corr[(corr>=0.5) | (corr <= -0.5)],vmax=1.0, vmin=-1.0,linewidths=0.1,annot=True,
            annot_kws={"size":14},square=True)
plt.show()
```

Kode korelasi antara atribut



Heatmap korelasi antaratribut

Dapat dilihat bahwa:

- Hampir semua variabel memiliki korelasi kuat terhadap semua variabel lainnya. Kecuali model_year yang hanya berkorelasi kuat terhadap mpg.
- Kolom acceleration tidak mempengaruhi nilai mpg.
- Yang paling banyak mempengaruhi atribut lainnya adalah kolom mpg, cylinders, displacement, dan horsepower.
- Cylinders, displacement, dan horsepower mempengaruhi nilai mpg secara kebalikannya. Ini wajar, karena semakin kecil cylinders maka mpg akan semakin besar sebab yang tenaga (horsepower) yang dihasilkan lebih sedikit yang disebabkan oleh chamber untuk membakar bahan bakarnya lebih sedikit. Karena cylinders berhubungan dengan displacement maka hubungan keduanya ini wajar.

2.6.3. Distribusi Data dengan Boxplot

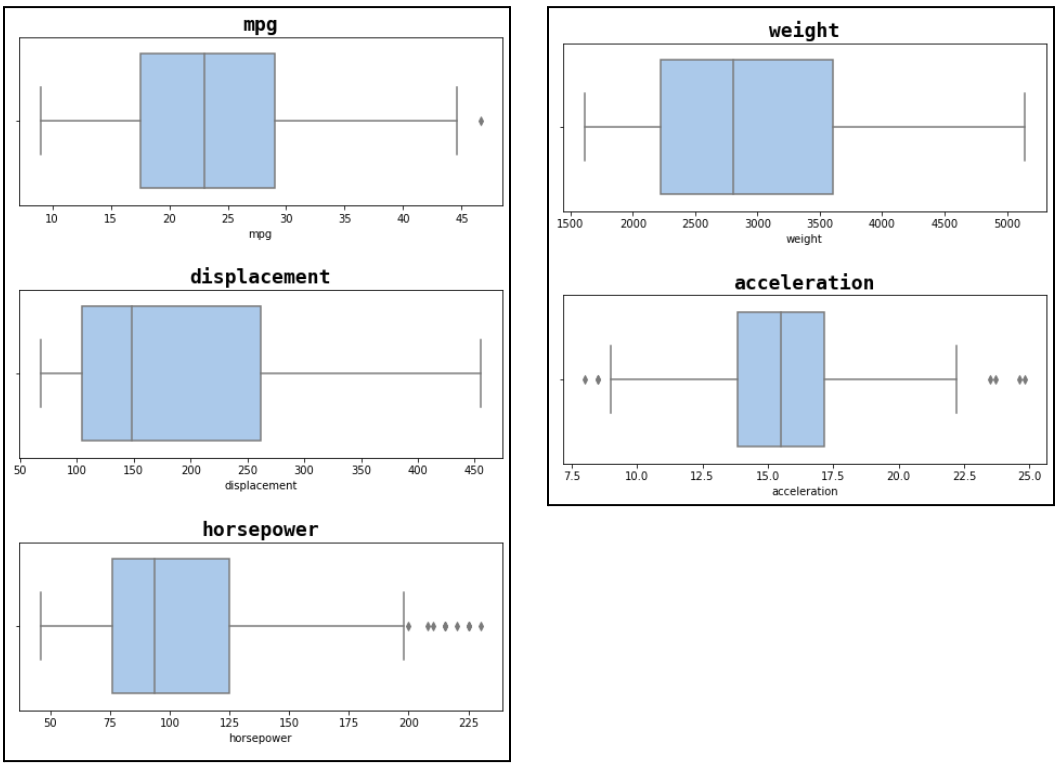
Boxplot atau box-and-whisker plot adalah tipe plot yang digunakan untuk menampilkan distribusi data dengan menggambarkan keempat quartil (q1, q2, q3, dan q4) dan batas bawah (minimum) dan batas atas (maksimum) dari sekumpulan data. Boxplot memberikan informasi tentang kisaran data (range), titik tengah (median), dan simpangan baku (standar deviasi) dari sekumpulan data. Boxplot sangat berguna untuk membandingkan distribusi data dari beberapa kelompok atau untuk menemukan outlier

(nilai yang sangat berbeda dari kebanyakan nilai dalam sekumpulan data). Boxplot juga bisa digunakan untuk menemukan pola dalam data atau untuk mengetahui seberapa tersebar data tersebut.

```
fig, ax = plt.subplots(5, 1, figsize = (8, 20))

for i, col in enumerate(col_numeric):
    plt.subplots_adjust(hspace=0.5)
    g = sns.boxplot(data = df, x = col, ax = ax[i], palette = "pastel")
    g.set_title(col, weight = "bold", fontsize = 18, fontname = "monospace")
plt.show()
```

Kode distribusi data dengan boxplot



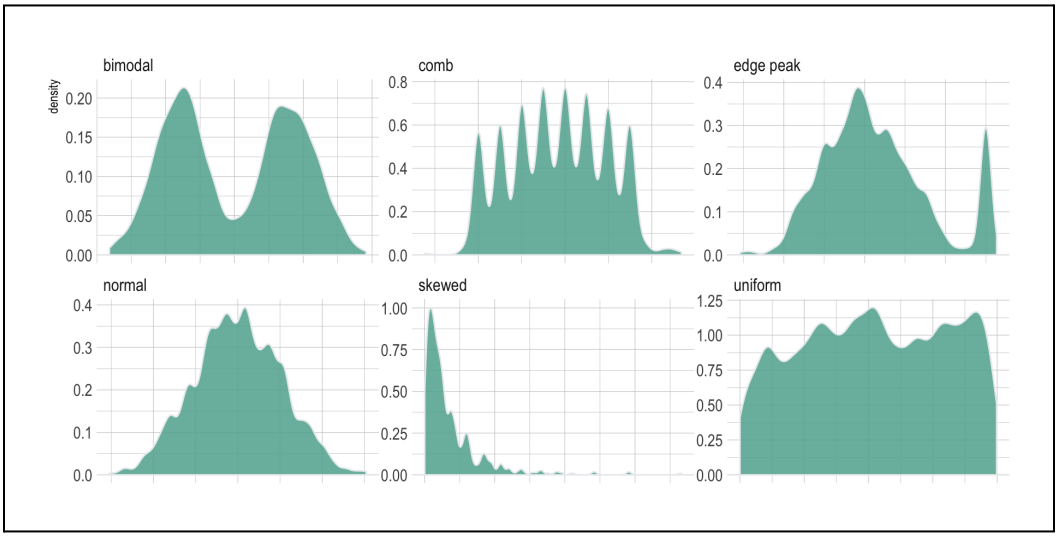
Hasil distribusi data dengan boxplot

Dapat dilihat bahwa beberapa kolom yang memiliki outlier adalah kolom mpg, horsepower, dan acceleration. Karena outliers bukan merupakan noice dari data, maka outliers tersebut tidak perlu dihapus.

2.6.4. Distribusi Data dengan Density Plot

Density plot adalah tipe plot yang digunakan untuk menampilkan distribusi data dengan menggambarkan kepadatan data pada setiap titik. Density plot mirip dengan

histogram, namun lebih halus dan tidak terpusat pada titik-titik tertentu seperti bin dalam histogram. Density plot bisa digunakan untuk menampilkan distribusi data dari variabel numerik atau kategorik. Density plot sangat berguna untuk mengetahui pola dalam data atau untuk membandingkan distribusi data dari beberapa kelompok.

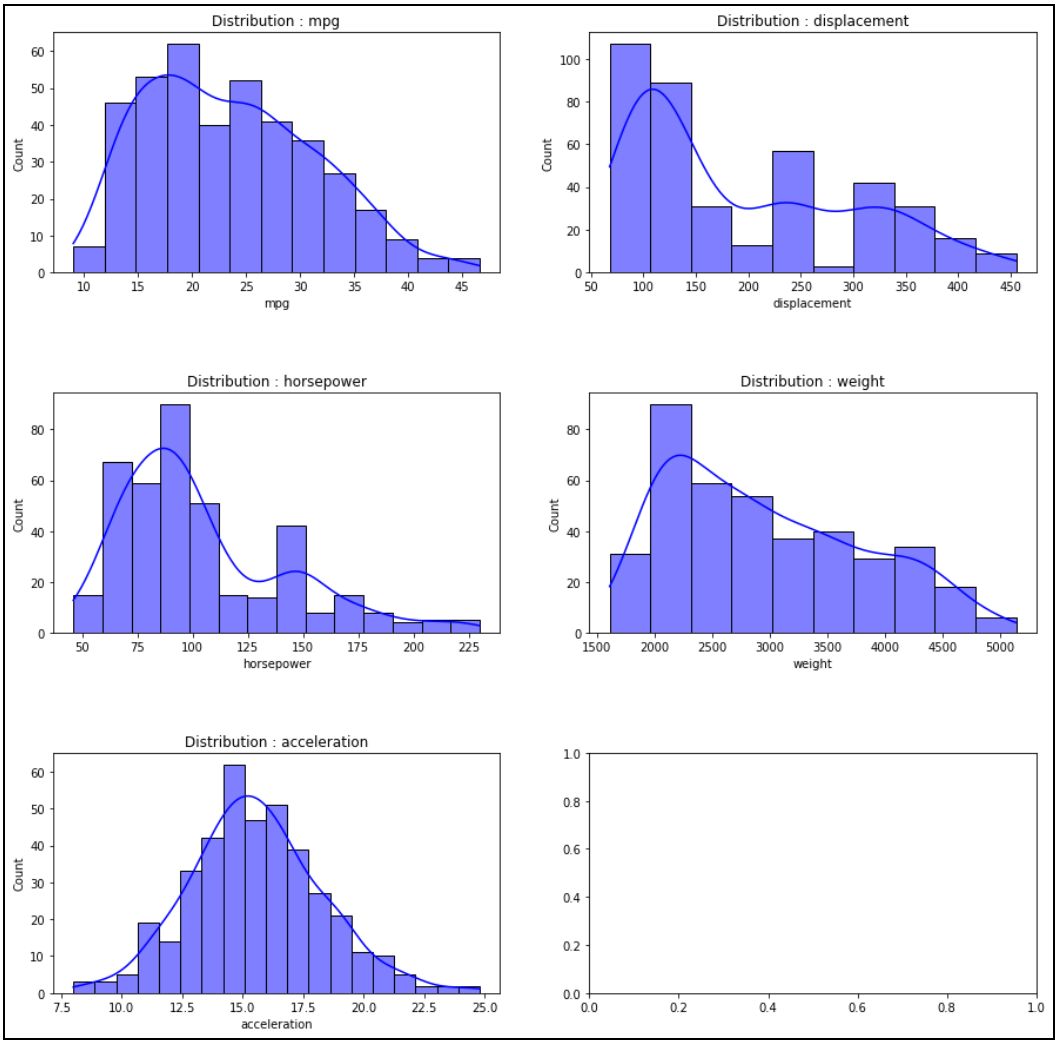


Tipe-tipe density plot

```
45]  fig, ax = plt.subplots(nrows = 3,ncols = 2,figsize = (15,15))


    for i in range(len(col_numeric)):
        plt.subplots_adjust(hspace=0.5)
        plt.subplot(3,2,i+1)
        sns.histplot(df[col_numeric[i]],color = 'b',kde=True)
        title = 'Distribution : ' + col_numeric[i]
        plt.title(title)
    plt.show()
```

Kode density plot



Hasil density plot

- acceleration: normal.
- displacement, horsepower, weight: left-skewed.

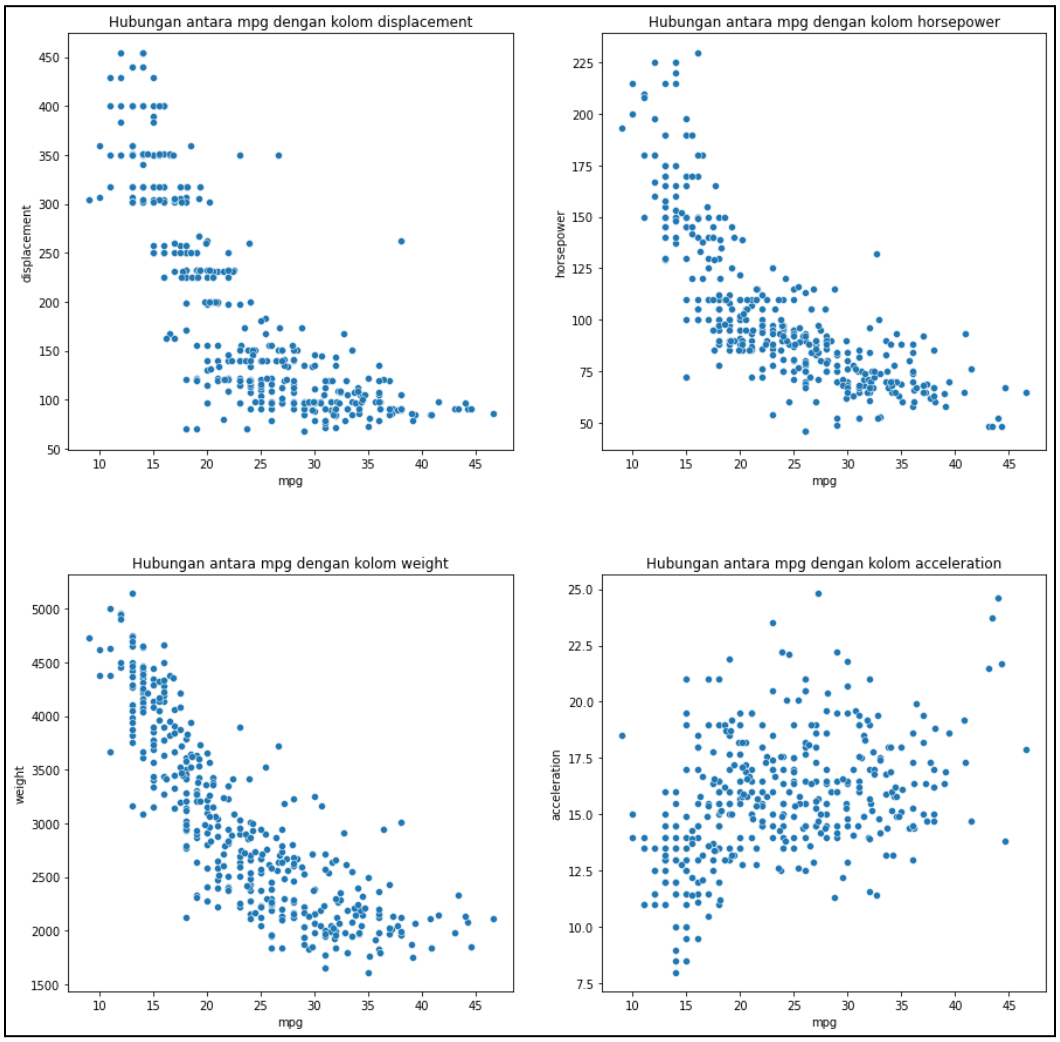
2.6.5. Hubungan mpg dengan Kolom Numerik yang lain dengan Scatter Plot

Scatter plot adalah tipe plot yang digunakan untuk menampilkan hubungan antara dua variabel dengan menggambarkan setiap pasangan nilai (x, y) pada sebuah kartesian (koordinat). Scatter plot menggunakan satu variabel pada sumbu x dan satu variabel pada sumbu y. Scatter plot sangat berguna untuk menampilkan hubungan atau korelasi antara dua variabel, serta untuk menemukan pola dalam data.

```
colm = ["displacement", "horsepower", "weight", "acceleration"]
fig, ax = plt.subplots(nrows = 2,ncols = 2,figsize = (15,15))

for i in range(len(colm)):
  plt.subplot(2,2,i+1)
  plt.subplots_adjust(hspace=0.3)
  sns.scatterplot(data = df,x="mpg",y=colm[i],palette = 'pastel',alpha=1)
  title = "Hubungan antara mpg dengan kolom " + colm[i]
  plt.title(title)
plt.show()
```

Kode scatter plot



Hasil scatter plot

Terdapat korelasi kuat negatif antara kolom mpg, dengan semua kolom numerik kecuali acceleration.

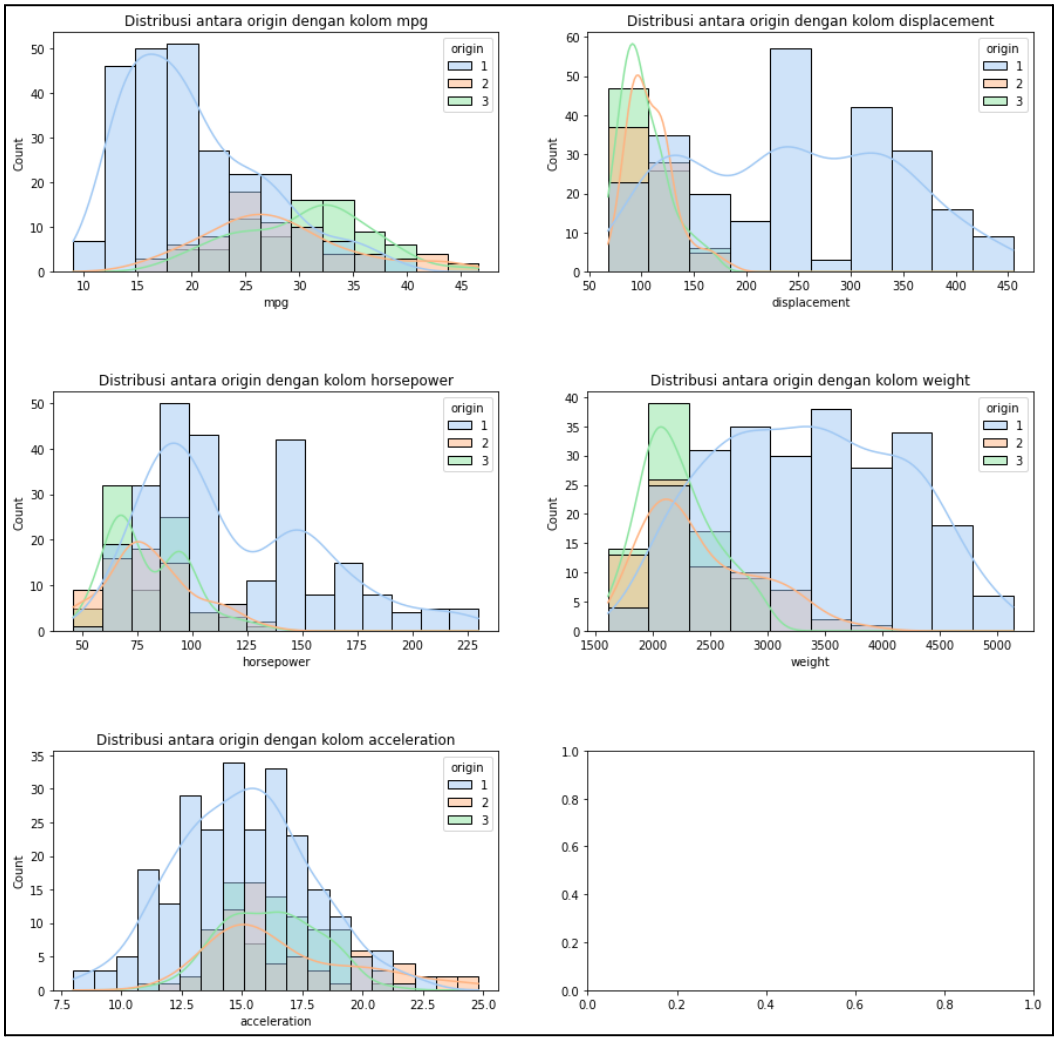
2.7. Eksplorasi Data Numerik dan Kategorikal

2.7.1. Hubungan antara kolom origin dengan kolom numerik

```
fig, ax = plt.subplots(nrows = 3,ncols = 2,figsize = (15,15))

for i in range(len(col_numeric)):
  plt.subplot(3,2,i+1)
  plt.subplots_adjust(hspace=0.5)
  sns.histplot(data = df,x=df[col_numeric[i]],palette = 'pastel',kde=True,hue="origin")
  title = "Hubungan antara origin dengan kolom " + col_numeric[i]
  plt.title(title)
plt.show()
```

Kode Eksplorasi hubungan kolom origin dan kolom lain



Hasil Eksplorasi hubungan kolom origin dan kolom lain

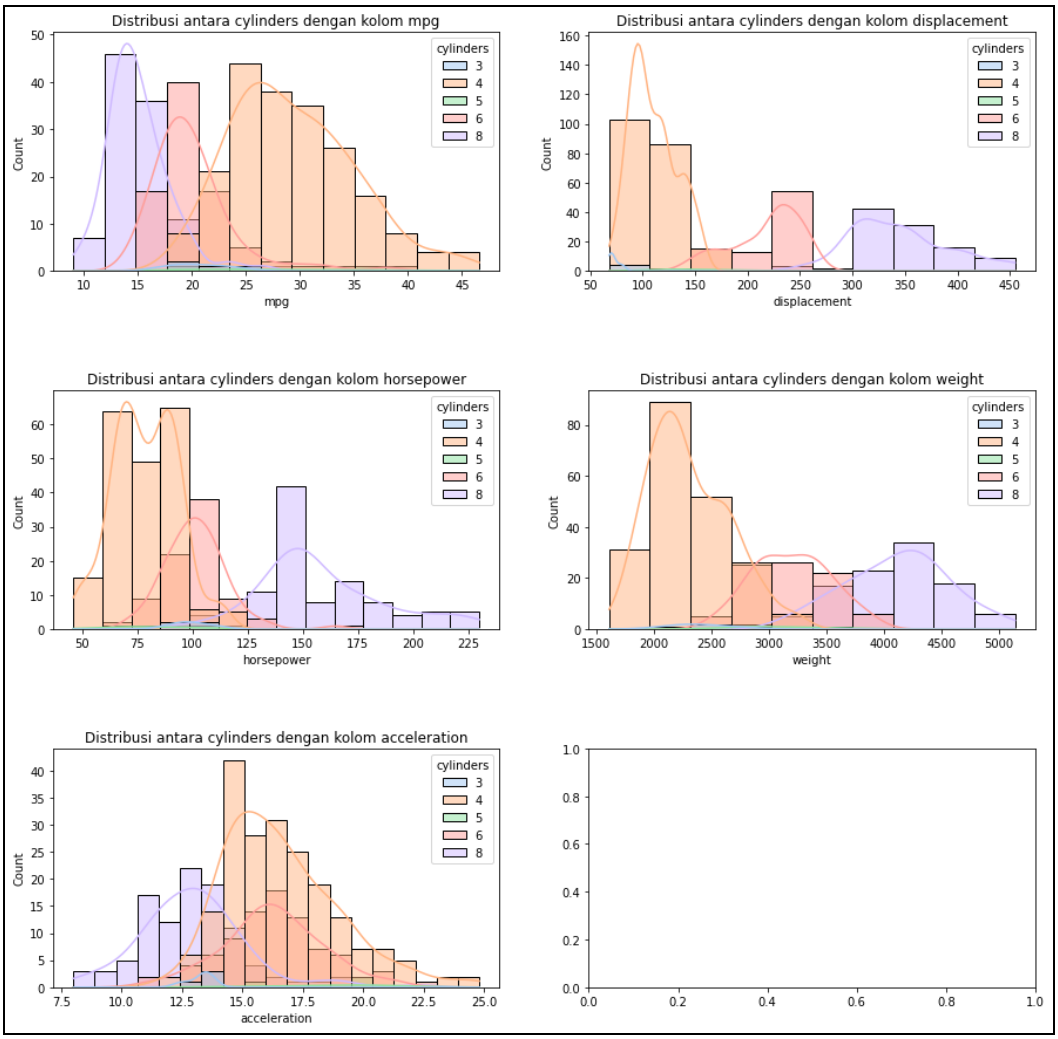
- record origin 1 memiliki mpg yang kecil, displacement yang besar, horsepower dan weight daripada origin 2 dan 3.
- origin 1 terdistribusi seragam pada weight dan displacement.
- Distribusi untuk origin 2 dan 3 relatif sama.
    - mpg, weight : normal.
    - displacement : right-skewed.
    - Pada horsepower, origin 2 bimodal, sementara origin 2 terdistribusi normal
- Semua origin berdistribusi normal pada kolom acceleration

2.7.2. Hubungan antara kolom cylinders dengan kolom yang lain

```
fig, ax = plt.subplots(nrows = 3,ncols = 2,figsize = (15,15))

for i in range(len(col_numeric)):
  plt.subplot(3,2,i+1)
  plt.subplots_adjust(hspace=0.5)
  sns.histplot(data = df,x=df[col_numeric[i]],palette = 'pastel',kde=True,hue="cylinders")
  title = "Hubungan antara cylinders dengan kolom " + col_numeric[i]
  plt.title(title)
plt.show()
```

Kode Eksplorasi hubungan kolom cylinders dan kolom lain

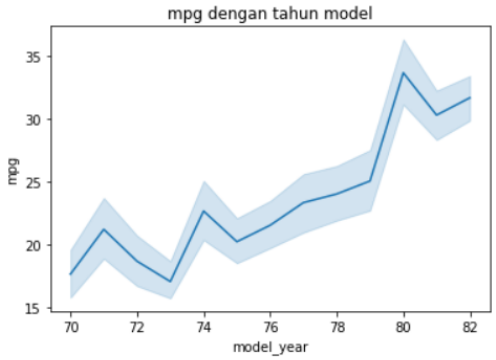


Hasil Eksplorasi hubungan kolom cylinders dan kolom lain

cylinders 4, 6, dan 8 terdistribusi normal jika dihubungkan dengan mpg, weight, dan acceleration.

2.7.3. Kenaikan kolom mpg terhadap kolom model_year

```
[49] plt.title("mpg dengan tahun model")
     sns.lineplot(data = df, x="model_year",y="mpg")
     plt.show()
```



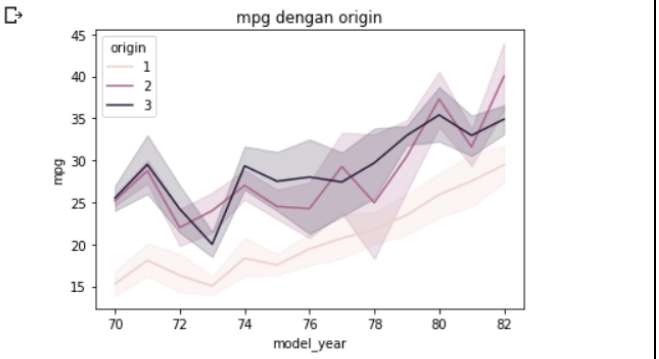
Semakin tinggi model_year maka mpg-nya juga semakin meningkat

Kolom mpg berpengaruh terhadap model_year

Jika semakin tinggi nilai dari model_year, maka akan semakin meningkat pula nilai mpg-nya.

2.7.4. Kenaikan kolom mpg terhadap kolom model_year yang ditinjau juga dari kolom origin

```
plt.title("mpg dengan origin")
sns.lineplot(data = df, x="model_year",y="mpg",hue="origin")
plt.show()
```



Kolom mpg berpengaruh terhadap model_year dan ditinjau juga dari kolom origin

- mpg bertambah untuk setiap origin.
- mpg origin 1 selalu di bawah origin 2 dan 3.

2.8. Menghitung Outliers

Outliers adalah nilai-nilai yang sangat berbeda dari kebanyakan nilai lain dalam sekumpulan data. Outliers bisa terjadi karena kesalahan pengukuran, kejadian yang tidak biasa, atau karena adanya variabel yang tidak tercakup dalam model.

```
Q1 = df[col_numeric].quantile(0.25)
Q3 = df[col_numeric].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

filt = ((df[col_numeric] < (lower_bound)) | (df[col_numeric] > (upper_bound)))
filt.sum()
```

```
mpg             1
displacement    0
horsepower     11
weight          0
acceleration    7
dtype: int64
```

Jumlah outliers

```
df.loc[filt["mpg"]]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
322	46.6	4	86.0	65.0	2110	17.9	80	3	mazda glc

Outliers pada kolom mpg

Dapat dilihat bahwa value outliers di sini wajar sebab jika dilihat antara korelasi tiap variabel, mpg sangat berbanding terbalik dengan jumlah cylinders, displacement, horsepower, dan weightnya. sementara untuk record ke-322 ini, nilai dari kolom-kolom tersebut merupakan nilai yang sangat minimum.

```
[54] df.loc[filt["acceleration"]]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
7	14.0	8	440.0	215.0	4312	8.5	70	1	plymouth fury iii
9	15.0	8	390.0	190.0	3850	8.5	70	1	amc ambassador dpl
11	14.0	8	340.0	160.0	3609	8.0	70	1	plymouth 'cuda 340
59	23.0	4	97.0	54.0	2254	23.5	72	2	volkswagen type 3
299	27.2	4	141.0	71.0	3190	24.8	79	2	peugeot 504
326	43.4	4	90.0	48.0	2335	23.7	80	2	vw dasher (diesel)
394	44.0	4	97.0	52.0	2130	24.6	82	2	vw pickup

Outliers pada kolom acceleration

Pada kasus di atas, ini juga bukan merupakan noice outliers karena masih berkorelasi dengan kolom-kolom yang mempengaruhi nilai acceleration, yaitu cylinders, displacement, dan horsepower.

```
55] df.loc[filt["horsepower"]]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
6	14.0	8	454.0	220.0	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215.0	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225.0	4425	10.0	70	1	pontiac catalina
13	14.0	8	455.0	225.0	3086	10.0	70	1	buick estate wagon (sw)
25	10.0	8	360.0	215.0	4615	14.0	70	1	ford f250
26	10.0	8	307.0	200.0	4376	15.0	70	1	chevy c20
27	11.0	8	318.0	210.0	4382	13.5	70	1	dodge d200
67	11.0	8	429.0	208.0	4633	11.0	72	1	mercury marquis
94	13.0	8	440.0	215.0	4735	11.0	73	1	chrysler new yorker brougham
95	12.0	8	455.0	225.0	4951	11.0	73	1	buick electra 225 custom
116	16.0	8	400.0	230.0	4278	9.5	73	1	pontiac grand prix

Outliers pada kolom horsepower

pada kasus di atas, nilai dari horsepowernya juga sangat mungkin, karena cylinders-nya bernilai sangat besar, yaitu 8.

```
[56] # persentase outliers
     filt = ((df[col_numeric] < (lower_bound)) | (df[col_numeric] > (upper_bound)))
     filt = filt/len(df[col_numeric])*100
     filt.sum()

     mpg             0.251256
     displacement    0.000000
     horsepower      2.763819
     weight          0.000000
     acceleration    1.758794
     dtype: float64
```

Persentase outliers

Karena selain outliers-nya yang mungkin, persentase outliers dari masing-masing atribut sangat sedikit. Sehingga di sini kami tidak melakukan handling untuk nilai outliers.

2.9. Transformasi data

Scaling data adalah proses memformat data agar memiliki nilai yang sama dengan data lain dalam rentang yang sama. Ini biasanya dilakukan untuk menghindari bias dalam analisis data. Ada beberapa cara untuk melakukan scaling data, termasuk normalisasi, standarisasi, dan min-max scaling.

- Normalisasi adalah proses mengubah setiap nilai dalam data menjadi nilai yang relatif terhadap nilai tertinggi dan terendah dari data. Ini biasanya dilakukan dengan mengurangi nilai tertinggi dan terendah dari data dengan satuan yang sama, sehingga data memiliki rentang 0 hingga 1.
- Standarisasi adalah proses mengubah setiap nilai dalam data menjadi nilai yang relatif terhadap rata-rata dan standar deviasi dari data. Ini biasanya dilakukan dengan mengurangi rata-rata dari data dengan standar deviasi, sehingga data memiliki rata-rata 0 dan standar deviasi 1.
- Min-max scaling adalah proses mengubah setiap nilai dalam data menjadi nilai yang relatif terhadap nilai tertinggi dan terendah dari data. Ini biasanya dilakukan dengan mengurangi nilai tertinggi dan terendah dari data dengan satuan yang sama, sehingga data memiliki rentang yang ditentukan.

Scaling data sering digunakan dalam machine learning untuk membantu algoritma mengelola data dengan lebih baik dan memprediksi hasil yang lebih akurat.

```
scaler = preprocessing.MinMaxScaler()
names = df.columns
d = scaler.fit_transform(df)
minMax_df = pd.DataFrame(d, columns=names)
minMax_df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	0.239362	1.0	0.617571	0.456522	0.536150	0.238095	0.0	0.0
1	0.159574	1.0	0.728682	0.646739	0.589736	0.208333	0.0	0.0
2	0.239362	1.0	0.645995	0.565217	0.516870	0.178571	0.0	0.0
3	0.186170	1.0	0.609819	0.565217	0.516019	0.238095	0.0	0.0
4	0.212766	1.0	0.604651	0.510870	0.520556	0.148810	0.0	0.0

Hasil scaling dari Dataset

2.10. Splitting Dataset

```
[ ]  df_model = minMax_df.copy()

[ ]  y = df_model["mpg"]
     df_model.drop(['mpg'], axis=1, inplace=True)

     y

     0      0.239362
     1      0.159574
     2      0.239362
     3      0.186170
     4      0.212766
              ...
     393    0.478723
     394    0.930851
     395    0.611702
     396    0.505319
     397    0.585106
     Name: mpg, Length: 398, dtype: float64
```

Kolom y target

```
df_model
```

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	1.0	0.617571	0.456522	0.536150	0.238095	0.0	0.0
1	1.0	0.728682	0.646739	0.589736	0.208333	0.0	0.0
2	1.0	0.645995	0.565217	0.516870	0.178571	0.0	0.0
3	1.0	0.609819	0.565217	0.516019	0.238095	0.0	0.0
4	1.0	0.604651	0.510870	0.520556	0.148810	0.0	0.0
...	...	...	...	...	...	...	...
393	0.2	0.186047	0.217391	0.333711	0.452381	1.0	0.0
394	0.2	0.074935	0.032609	0.146583	0.988095	1.0	0.5
395	0.2	0.173127	0.206522	0.193365	0.214286	1.0	0.0
396	0.2	0.134367	0.179348	0.286929	0.630952	1.0	0.0
397	0.2	0.131783	0.195652	0.313864	0.678571	1.0	0.0

398 rows × 7 columns

Kolom selain y target

```
[ ] X_train, X_test, y_train, y_test = train_test_split(df_model, y, test_size=0.33, random_state=42)

[ ] X_train
```

	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
324	0.2	0.043928	0.103261	0.140913	0.666667	0.833333	1.0
176	0.6	0.423773	0.239130	0.453076	0.535714	0.416667	0.0
119	0.2	0.118863	0.244565	0.274738	0.357143	0.250000	0.5
192	0.6	0.470284	0.320652	0.493337	0.386905	0.500000	0.0
202	0.6	0.490956	0.266304	0.447973	0.583333	0.500000	0.0
...	...	...	...	...	...	...	...
71	0.0	0.005168	0.277174	0.203289	0.327381	0.166667	1.0
106	1.0	0.728682	0.728261	0.818259	0.267857	0.250000	0.0
270	0.2	0.170543	0.266304	0.255741	0.404762	0.666667	1.0
348	0.2	0.054264	0.086957	0.123901	0.553571	0.916667	1.0
102	0.2	0.074935	0.000000	0.095549	0.773810	0.250000	0.5

266 rows × 7 columns

Splitting dataset

Di sini dataset terbagi menjadi data training dan data testing. Sehingga model dapat belajar dengan baik.

3. PEMODELAN

Untuk membuat model yang optimal akan kumpulan data mobil ini, berbagai model pembelajaran mesin dan metode *begging-regressor* akan digunakan untuk mengatasi tantangan klasifikasi yang menantang ini.

3.1. Decision Tree Regressor

Decision Tree Regressor adalah model machine learning yang menggunakan struktur tree untuk membuat prediksi numerik pada data. Tree ini terdiri dari node yang mewakili pertanyaan atau kondisi, dan setiap node memiliki cabang yang mengarah ke node lainnya yang mewakili jawaban atau hasil dari kondisi tersebut. Proses memprediksi nilai dengan decision tree dimulai dari root node dan kemudian melakukan perjalanan melalui tree sesuai dengan jawaban dari setiap pertanyaan di setiap node hingga mencapai leaf node, yang merupakan node terakhir di tree dan memberikan nilai prediksi.

Decision tree regressor sering digunakan untuk memecahkan masalah regresi, yaitu masalah yang berkaitan dengan memprediksi nilai numerik berdasarkan data masukan. Misalnya, decision tree regressor dapat digunakan untuk memprediksi harga rumah berdasarkan fitur-fitur seperti luas tanah, jumlah kamar tidur, dan lokasi.

Salah satu keuntungan dari decision tree regressor adalah mudah dipahami dan diinterpretasikan, karena strukturnya yang jelas dan terbuka. Namun, decision tree juga memiliki beberapa kelemahan, seperti mudah membentuk tree yang overfit pada data yang digunakan untuk pelatihannya, sehingga kurang akurat untuk data baru. Oleh karena itu, decision tree sering digunakan dengan teknik pemodelan lainnya, seperti random forest, untuk meningkatkan akurasi prediksi.

3.2. SVR Model

Support Vector Regression (SVR) adalah salah satu teknik machine learning yang digunakan untuk memecahkan masalah regresi, yaitu masalah yang berkaitan dengan memprediksi nilai numerik berdasarkan data masukan. SVR menggunakan teknik yang disebut "support vector machine" (SVM) untuk membuat model yang dapat memprediksi nilai dengan akurasi yang tinggi.

SVR mencoba menemukan garis atau hyperplane yang paling baik memisahkan data ke dua kelas. Hyperplane tersebut ditentukan oleh support vectors, yaitu data-data yang paling dekat dengan hyperplane. Setelah hyperplane teridentifikasi, SVR kemudian menggunakan rumus khusus untuk menghitung nilai prediksi untuk setiap titik data.

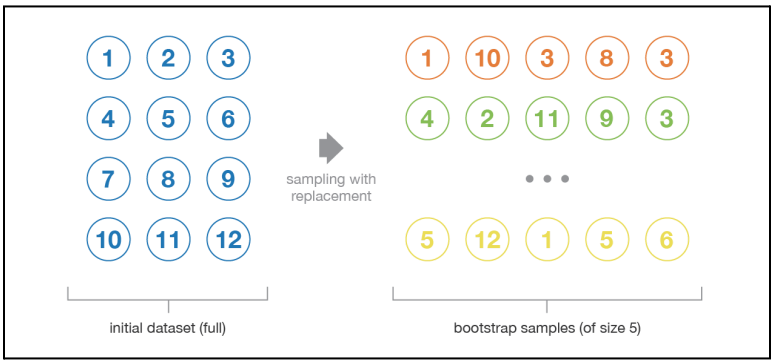
SVR memiliki beberapa keunggulan dibandingkan teknik regresi lainnya, seperti mudah diimplementasikan dan memiliki performa yang baik untuk data yang tidak terlalu rumit. Namun, SVR juga memiliki beberapa kelemahan, seperti membutuhkan waktu yang cukup lama untuk pelatihan dan tidak cocok untuk data dengan banyak fitur.

3.3. Gradient Boosting Regressor Model

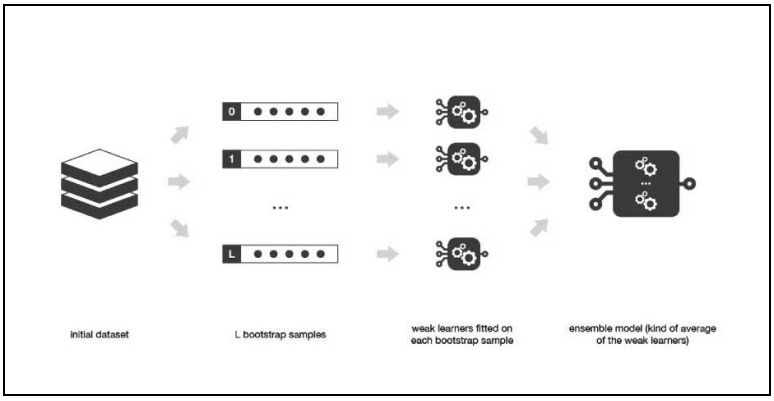
Gradient Boosting Regressor adalah salah satu algoritma pembelajaran mesin yang digunakan untuk melakukan pemodelan regresi terhadap data. Algoritma ini bekerja dengan cara membangun sejumlah model yang saling terkait dan bertumpuk, dimana setiap model dibangun untuk memperbaiki kesalahan dari model sebelumnya.

3.4. Bagging Regressor

Bagging Regressor adalah teknik machine learning yang digunakan untuk meningkatkan akurasi model regresi dengan menggunakan beberapa model regresi yang dilatih secara independen dan kemudian menggabungkan hasil dari masing-masing model tersebut. Bagging merupakan singkatan dari "bootstrap aggregating", yang merujuk pada proses mengambil sample acak dari data dengan menggunakan sampling dengan replacement. Setiap model dilatih dengan menggunakan sample data yang berbeda, sehingga model yang dihasilkan akan bervariasi satu sama lain.



Sampling with replacement



Ilustrasi Bagging Regresor

Bagging regressor biasanya digunakan dengan model regresi yang sederhana, seperti decision tree atau linear regression. Dengan menggunakan banyak model yang bervariasi, bagging regressor dapat mengurangi overfitting dan meningkatkan akurasi prediksi secara keseluruhan. Namun, bagging regressor membutuhkan waktu yang cukup lama untuk pelatihan karena harus melatih banyak model secara independen.

3.5. Proses Pemodelan

Karena kita menggunakan tiga pemodelan, maka kita membuat model sebanyak tiga, yaitu Decision Tree regressor, SVR, dan Gradient Boosting Regressor.

3.5.1. Decision Tree Regressor

```
[74] reggr_tree = BaggingRegressor(base_estimator = None, n_estimators=10, random_state=0).fit(X_train, y_train)

     y_predict_tree = reggr_tree.predict(X_test)
```

Pembangunan Model

Di sini dibangun sebuah model, dengan base_estimator = None. Karena default Bagging Regressor di scikitlearn adalah Decision Tree Regressor. Kemudian dengan base model tersebut kita membuat model bagging regressor dengan menggunakan data latih yang didefinisikan sebelumnya.

```
print(y_predict_tree)
```

```
[0.58244681 0.55319149 0.36728723 0.14760638 0.13829787 0.48271277
 0.48670213 0.07978723 0.20984043 0.28590426 0.11968085 0.78297872
 0.5212766  0.125      0.42154255 0.09840426 0.57074468 0.30611702
 0.16888298 0.80159574 0.42952128 0.27393617 0.59654255 0.50797872
 0.18218085 0.78803191 0.43484043 0.40824468 0.3        0.09308511
 0.53085106 0.67898936 0.23723404 0.44148936 0.7912234  0.11170213
 0.35319149 0.25026596 0.13164894 0.43351064 0.42978723 0.51595745
 0.28723404 0.07180851 0.37952128 0.68404255 0.45345745 0.36702128
 0.41755319 0.48271277 0.43351064 0.79734043 0.63297872 0.07978723
 0.41223404 0.1037234  0.18191489 0.52074468 0.38962766 0.26329787
 0.1356383  0.57978723 0.41117021 0.31382979 0.26595745 0.41595745
 0.36489362 0.63031915 0.50797872 0.15026596 0.68909574 0.10638298
 0.1037234  0.2893617  0.53138298 0.29920213 0.26382979 0.53537234
 0.51595745 0.22579787 0.20744681 0.47606383 0.76489362 0.67446809
 0.39361702 0.10106383 0.6962766  0.69920213 0.45478723 0.06914894
 0.18085106 0.76542553 0.46143617 0.58351064 0.32898936 0.73271277
 0.54202128 0.33138298 0.4125     0.14361702 0.62074468 0.40026596
 0.28457447 0.13829787 0.19148936 0.10638298 0.51329787 0.12234043
 0.66914894 0.75718085 0.58510638 0.3981383  0.28430851 0.78829787
 0.57473404 0.37234043 0.49069149 0.31569149 0.84547872 0.29414894
 0.13430851 0.43351064 0.32393617 0.2643617  0.34255319 0.43218085
 0.74654255 0.46037234 0.65132979 0.58723404 0.78404255 0.58244681]
```

Hasil prediksi model bagging - Decision Tree Regressor

3.5.2. SVR

```
[81] reggr_SVR = BaggingRegressor(base_estimator = SVR(),n_estimators=10, random_state=0).fit(X_train, y_train)

[105] y_predict_SVR = reggr_SVR.predict(X_test)
```

Pembangunan model

Di sini dibangun sebuah model, dengan base_estimator = SVR(). Kemudian dengan base model tersebut kita membuat model bagging regressor dengan menggunakan data latih yang didefinisikan sebelumnya.

```
print(y_predict_SVR)
```

```
[0.63182933 0.57797121 0.31734454 0.16571257 0.11176673 0.4360987
 0.51419548 0.16565505 0.20165504 0.26778896 0.18844708 0.75624259
 0.40193804 0.13466785 0.39979617 0.14185727 0.57952891 0.29989978
 0.21167569 0.77364224 0.37867595 0.29023494 0.47929499 0.55933621
 0.19538341 0.84542782 0.40179539 0.43639473 0.27130011 0.14015684
 0.53807908 0.74981477 0.28584071 0.37548085 0.72846229 0.17730062
 0.33743271 0.2533271  0.12958028 0.44453604 0.41399566 0.48946873
 0.28547383 0.12902048 0.32807236 0.69628687 0.42022171 0.33539502
 0.38841538 0.49834566 0.35560628 0.68318039 0.67778101 0.12109799
 0.46465636 0.13597464 0.26791137 0.55139469 0.39196222 0.2381362
 0.14915194 0.60216859 0.40117133 0.31639598 0.26504877 0.37621343
 0.4349834  0.67335989 0.52499246 0.13704285 0.64894781 0.11466572
 0.16867334 0.2705003  0.43821835 0.26782133 0.22842415 0.57044487
 0.50370143 0.25822747 0.18039679 0.42688963 0.77309338 0.68015325
 0.36242435 0.12758895 0.7203286  0.63376453 0.45147717 0.15639262
 0.20228598 0.58415273 0.42320165 0.69302612 0.32681731 0.66566004
 0.58652274 0.31588601 0.48600687 0.11693574 0.63576424 0.37902446
 0.29278053 0.15566809 0.20850758 0.19771609 0.49505929 0.12696906
 0.69208051 0.77816441 0.6158729  0.4532787  0.2649642  0.76209912
 0.52365839 0.34745636 0.44506806 0.29410198 0.6510712  0.25662915
 0.14259629 0.37901002 0.31223032 0.30332002 0.31361156 0.46568295
 0.74657144 0.49807049 0.66920008 0.66149128 0.75811248 0.56748358]
```

Hasil prediksi Bagging - SVR

3.5.3. Gradient Boosting Regressor

```
[88] reggr_Gradient = BaggingRegressor(base_estimator = GradientBoostingRegressor()).fit(X_train, y_train)

     y_predict_Gradient = reggr_Gradient.predict(X_test)
```

Pembangunan Model

Di sini dibangun sebuah model, dengan base_estimator = GradientBoostingRegressor. Kemudian dengan base model tersebut kita membuat model bagging regressor dengan menggunakan data latih yang didefinisikan sebelumnya.

```
y_predict_Gradient
```

```
array([0.55849307, 0.55699476, 0.30548346, 0.16802877, 0.11985102,
       0.48221358, 0.47485955, 0.07561546, 0.22558536, 0.27448218,
       0.10722988, 0.76644308, 0.51901739, 0.13045842, 0.41024519,
       0.09628835, 0.58570403, 0.31556435, 0.18059006, 0.75721074,
       0.40379801, 0.27655254, 0.46694862, 0.52142688, 0.17463703,
       0.69661449, 0.39769156, 0.38113453, 0.23631613, 0.0986899 ,
       0.48199434, 0.67135787, 0.21042108, 0.39853295, 0.81435816,
       0.10318493, 0.36588427, 0.24784337, 0.13081053, 0.43530589,
       0.48185284, 0.51102306, 0.29718964, 0.06780345, 0.38475289,
       0.68897306, 0.45133834, 0.3648812 , 0.41646771, 0.46773011,
       0.39011782, 0.74903426, 0.66433649, 0.08599663, 0.50429011,
       0.10647247, 0.2129748 , 0.54886921, 0.38781943, 0.25903174,
       0.14202906, 0.55272933, 0.38689673, 0.29981546, 0.26162423,
       0.43086525, 0.36958373, 0.66004211, 0.4820288 , 0.13007129,
       0.65395796, 0.12240587, 0.10679689, 0.23373167, 0.47901049,
       0.29245986, 0.25184108, 0.55588954, 0.54743012, 0.20060156,
       0.17731859, 0.47112475, 0.77311221, 0.62722617, 0.38787639,
       0.10613219, 0.67003297, 0.66270522, 0.3888659 , 0.0761477 ,
       0.1652217 , 0.71611644, 0.45575741, 0.59846753, 0.33024671,
       0.69857516, 0.57404402, 0.30032625, 0.4485829 , 0.12193965,
       0.61916418, 0.40806394, 0.27241118, 0.15617136, 0.22586001,
       0.1057517 , 0.57134034, 0.11754242, 0.69843821, 0.76949972,
       0.52911775, 0.41406771, 0.23443833, 0.74612027, 0.54928221,
       0.33960403, 0.48232202, 0.29911785, 0.71263829, 0.308614  ,
       0.14063687, 0.36609846, 0.30908327, 0.29389076, 0.32607636,
       0.51050991, 0.73941972, 0.47071472, 0.66388381, 0.59460816,
       0.76838343, 0.5523474 ])
```

Hasil prediksi - Gradient Boosting Regressor

4. EVALUASI

4.1. R2 Score

R square merupakan suatu nilai yang memperlihatkan seberapa besar variabel independen (eksogen) mempengaruhi variabel dependen (endogen). R squared merupakan angka yang berkisar antara 0 sampai 1 yang mengindikasikan besarnya kombinasi variabel independen secara bersama – sama mempengaruhi nilai variabel dependen. R2 Score dihitung dengan membandingkan variasi data aktual dengan variasi data yang diprediksi oleh model.

Secara matematis, R2 Score dihitung dengan menggunakan rumus:

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

Rumus R2 score

Di mana:

- Residual sum of squares adalah jumlah kuadrat error (selisih antara nilai aktual dan nilai prediksi).
- Total sum of squares adalah jumlah kuadrat selisih antara nilai aktual dan nilai rata-rata.

Semakin tinggi nilai R2 Score, semakin baik model tersebut dalam memprediksi data. Namun, perlu diingat bahwa R2 Score tidak selalu merupakan ukuran yang tepat untuk semua jenis data. Jadi, sebaiknya juga melakukan perbandingan dengan metrik lain untuk menilai kinerja model.

4.2. Mean Absolute Error

Mean Absolute Error (MAE) adalah metrik yang digunakan untuk mengukur keakuratan model regresi. MAE dihitung dengan menjumlahkan selisih antara nilai aktual dan nilai prediksi, kemudian membagi jumlah tersebut dengan jumlah data yang ada. Secara matematis, MAE dihitung dengan menggunakan rumus:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - y_j|$$

Rumus MAE

Di mana:

- y adalah nilai aktual.
- ŷ adalah nilai prediksi.
- Σ adalah operator penjumlahan.
- |x| adalah nilai absolut dari x.
- n adalah jumlah data.

MAE mengukur seberapa besar rata-rata error yang dihasilkan oleh model. Semakin kecil nilai MAE, semakin baik model tersebut dalam memprediksi data. Namun, perlu diingat bahwa MAE tidak selalu merupakan metrik yang tepat untuk semua jenis data. Jadi, sebaiknya juga melakukan perbandingan dengan metrik lain untuk menilai kinerja model.

4.3. Mean Squared Error

Mean Squared Error (MSE) adalah metrik yang digunakan untuk mengukur keakuratan model regresi. MSE dihitung dengan menjumlahkan kuadrat selisih antara nilai aktual dan nilai prediksi, kemudian membagi jumlah tersebut dengan jumlah data yang ada. Secara matematis, MSE dihitung dengan menggunakan rumus:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Rumus MSE

Di mana:

- y adalah nilai aktual.
- ŷ adalah nilai prediksi.
- Σ adalah operator penjumlahan.
- n adalah jumlah data.

MSE mengukur seberapa besar rata-rata kuadrat error yang dihasilkan oleh model. Semakin kecil nilai MSE, semakin baik model tersebut dalam memprediksi data. Namun, perlu diingat bahwa MSE tidak selalu merupakan metrik yang tepat untuk semua jenis data. Jadi, sebaiknya juga melakukan perbandingan dengan metrik lain untuk menilai kinerja model.

Berikut adalah hasil evaluasi yang kita lakukan.

	R2 Score	Mean Absolute Error	Mean Squared Error
Decision Tree Regressor	0.861185	0.053359	0.005651
SVR	0.913433	0.046152	0.003524
Gradient Boosting Regressor	0.885653	0.049273	0.004655

Hasil Evaluasi

Dapat dilihat bahwa untuk hasil evaluasi, SVR memiliki akurasi yang lebih baik dengan nilai 0.913433 untuk R2 Scorenya, sedangkan untuk tingkat ke-error-annya SVR memiliki tingkat error yang lebih sedikit.

5. EKSPERIMEN

Untuk proses eksperimen, di sini kita melakukan tiga eksperimen.

1) Untuk setiap model, kita melakukan tuning hyperparameter
2) Eksperimen dengan algoritma lain
3) Eksperimen tuning hyperparameter untuk proses baggingnya.

5.1. Tuning Hyperparameter Model

5.1.1. Decision Tree Regressor

Model Decision Tree Regressor di scikit-learn memiliki beberapa hyperparameter yang dapat disesuaikan untuk meningkatkan performa. Berikut ini adalah daftar beberapa hyperparameter yang paling penting:

- Hyperparameter max_depth merupakan kontrol kompleksitas model dengan mengatur kedalaman maksimum dari tree.
- Hyperparameter min_samples_split merupakan kontrol jumlah data yang dibutuhkan pada setiap node dengan mengatur jumlah sampel minimum yang diperlukan untuk membagi node internal.
- Hyperparameter min_samples_leaf merupakan kontrol jumlah data yang dibutuhkan pada setiap leaf node dengan mengatur jumlah sampel minimum yang diperlukan untuk berada pada leaf node.
- Max_features adalah hyperparameter untuk mengontrol jumlah fitur atau kolom data yang dipertimbangkan pada setiap pemutusan node. Max_features mengontrol bagaimana model memecah data menjadi bagian-bagian kecil yang terpisah dengan mengatur jumlah fitur yang dipertimbangkan pada setiap pemutusan.

Berikut adalah hyperparameter yang kita gunakan:

```
parameters = {
    "max_depth": [3, 5, 7, None],
    "min_samples_leaf": [1, 2, 4],
    "max_leaf_nodes": [5, 10, 20, None],
    "max_features": ["auto", "sqrt", "log2", None,1,2,3,4,5,6,7]
}
```

Kode Hyperparameter yang digunakan

```
parameters = {
    "max_depth": [3, 5, 7, None],
    "min_samples_leaf": [1, 2, 4],
    "max_leaf_nodes": [5, 10, 20, None],
    "max_features": ["auto", "sqrt", "log2", None,1,2,3,4,5,6,7]
}

dtr = DecisionTreeRegressor()
clf_dtr = GridSearchCV(dtr, parameters, cv=5)
clf_dtr.fit(X_train,y_train)
print(clf_dtr.best_params_)
print(clf_dtr.score(X_train,y_train))
```

```
{'max_depth': 5, 'max_features': 6, 'max_leaf_nodes': None, 'min_samples_leaf': 1}
0.9321499553571382
```

```
[79] bg_regg_dtr = BaggingRegressor(base_estimator=clf_dtr,n_estimators=10,random_state=0).fit(X_train, y_train)
```

```
[80] bg_regg_dtr.score(X_test,y_test)
```

```
0.868228515503453
```

Hasil Hyperparameter yang digunakan

Didapatkan bahwa hyperparameter terbaik adalah {'max_depth': 5, 'max_features': 6, 'max_leaf_nodes': None, 'min_samples_leaf': 1}. Dengan akurasi training sebesar 0.9321499553571382. Untuk akurasi testingnya adalah `0.868228515503453`.

5.1.2. Support Vector Regressor

Model Support Vector Regressor di scikit-learn memiliki beberapa hyperparameter yang dapat disesuaikan untuk meningkatkan performa. Berikut ini adalah daftar beberapa hyperparameter yang paling penting:

- Hyperparameter C merupakan parameter penalty dari error term yang mengontrol trade-off antara meningkatkan ukuran margin dan memastikan model dapat memfit data.
- Hyperparameter kernel merupakan tipe kernel yang digunakan dalam model yang menentukan tipe fungsi yang digunakan untuk memetakan data ke dalam ruang dimensi yang lebih tinggi.
- Hyperparameter gamma merupakan koefisien kernel untuk kernel 'rbf', 'poly', dan 'sigmoid' yang mengontrol kompleksitas model dan dapat membantu mencegah overfitting.
- Hyperparameter epsilon adalah hyperparameter pada Support Vector Regression (SVR) yang mengontrol seberapa dekat nilai yang dihasilkan oleh model harus

sesuai dengan nilai yang sebenarnya. Epsilon memberikan "toleransi kesalahan" kepada model, sehingga model tidak perlu benar-benar sesuai dengan data pada setiap titik.

Berikut adalah hyperparameter yang kita gunakan:

```
parameters = {'kernel': ('linear', 'rbf','poly'),
              'C':[1.5, 1.0, 10],'gamma': ['scale','auto',1e-7, 1e-4]
              'epsilon':[0.1,0.2,0.5,0.3]}
```

Kode Hyperparameter yang akan digunakan

```
parameters = {'kernel': ('linear', 'rbf','poly'),
              'C':[1.5, 1.0, 10],'gamma': ['scale','auto',1e-7, 1e-4],
              'epsilon':[0.1,0.2,0.5,0.3]}
svr = SVR()
clf_svr = GridSearchCV(svr, parameters, cv=5)
clf_svr.fit(X_train,y_train)
print(clf_svr.best_params_)
print(clf_svr.score(X_train,y_train))

{'C': 1.0, 'epsilon': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
0.8901769976787334

bg_regg_svr = BaggingRegressor(base_estimator=clf_svr,n_estimators=10,random_state=0).fit(X_train, y_train)

[87] bg_regg_svr.score(X_test,y_test)

0.9121810927140912
```

Hasil Hyperparameter yang digunakan

Didapatkan bahwa hyperparameter terbaik adalah `{'C': 1.0, 'epsilon': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}`. Dengan akurasi training sebesar `0.8901769976787334`. Untuk akurasi testingnya adalah `0.9121810927140912`.

5.1.3. Gradient Boosting Regressor

Model Gradient Boosting Regressor di scikit-learn memiliki beberapa hyperparameter yang dapat disesuaikan untuk meningkatkan performa. Berikut ini adalah daftar beberapa hyperparameter yang paling penting:

- Loss adalah hyperparameter pada Gradient Boosting Regressor yang mengontrol bagaimana model mengevaluasi kesalahan pada setiap tree dan memperbaikinya pada tree berikutnya. Loss menentukan rumus yang digunakan untuk mengukur kesalahan model dan menyesuaikan bobot tree berikutnya untuk memperbaikinya.

- Hyperparameter max_depth merupakan kedalaman maksimum dari masing-masing tree yang mengontrol kompleksitas tree dan dapat membantu mencegah overfitting.
- Min_samples_leaf adalah hyperparameter pada Gradient Boosting Regressor yang mengontrol jumlah sampel minimum yang diperlukan untuk berada pada leaf node dalam tree. Min_samples_leaf dapat membantu mencegah overfitting dengan membatasi kemampuan model untuk mempelajari pola-pola kecil dalam data.

Berikut adalah hyperparameter yang kita gunakan:

```
parameters = {
    "loss": ["squared_error", "absolute_error", "huber", "quantile"],
    "max_depth": [3, 5],
    "min_samples_leaf": [1, 2]
}
```

Kode Hyperparameter yang digunakan

```
parameters = {
    "loss": ["squared_error", "absolute_error", "huber", "quantile"],
    "max_depth": [3, 5],
    "min_samples_leaf": [1, 2]
}
gbr = GradientBoostingRegressor()
clf_gbr = GridSearchCV(gbr, parameters, cv=5)
clf_gbr.fit(X_train,y_train)
print(clf_gbr.best_params_)
print(clf_gbr.score(X_train,y_train))
```

```
{'loss': 'huber', 'max_depth': 3, 'min_samples_leaf': 1}
0.9730819466749582
```

```
[92] bg_regg_gbr = BaggingRegressor(base_estimator=clf_gbr,n_estimators=10,random_state=0).fit(X_train, y_train)
```

```
[93] bg_regg_gbr.score(X_test,y_test)
```

```
0.8890287318202585
```

Hasil hyperparameter yang digunakan

Didapatkan bahwa hyperparameter terbaik adalah `{'loss': 'huber', 'max_depth': 3, 'min_samples_leaf': 1}`. Dengan akurasi training sebesar `0.9730819466749582`. Untuk akurasi testingnya adalah `0.8890287318202585`.

5.2. Memilih Algoritma Lain

Berikut adalah codenya:

```
def test_regression():
    hasil_arr = []
    grid = ParameterGrid({"max_samples": [0.5, 1.0],
                          "max_features": [0.5, 1.0],
                          "bootstrap": [True, False],
                          "bootstrap_features": [True, False]})

    for base_estimator in [None,LinearRegression(),
                           DummyRegressor(),QuantileRegressor(),Ridge(),Lasso(),
                           DecisionTreeRegressor(),
                           KNeighborsRegressor(),
                           SVR()]:
        for params in grid:
            para = []
            para.append(str(base_estimator))
            para_val = list(params.values())
            for i in para_val:
              para.append(i)
            reggr = BaggingRegressor(base_estimator=base_estimator,
                                     random_state=0,
                                     **params).fit(X_train, y_train)
            y_predict= reggr.predict(X_test)
            para.append(reggr.score(X_test,y_test))
            para.append(mean_absolute_error(y_test,y_predict))
            para.append(mean_squared_error(y_test,y_predict))
            hasil_arr.append(para)
    return hasil_arr
```

Kode program

```
hasil.loc[(hasil["bootstrap"]==True)&(hasil["bootstrap_features"]==False)&(hasil["max_features"]==1.0)&(hasil["max_samples"]==1.0)]
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
7	None	True	False	1.0	1.0	0.861185	0.053359	0.005651
23	LinearRegression()	True	False	1.0	1.0	0.841424	0.062926	0.006456
39	DummyRegressor()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
55	QuantileRegressor()	True	False	1.0	1.0	-0.028829	0.168396	0.041886
71	Ridge()	True	False	1.0	1.0	0.834370	0.064097	0.006743
87	Lasso()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
103	DecisionTreeRegressor()	True	False	1.0	1.0	0.861185	0.053359	0.005651
119	KNeighborsRegressor()	True	False	1.0	1.0	0.880996	0.053232	0.004845
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524

Akurasi untuk algoritma yang berbeda

```
df_awal.sort_values(by=["score"],ascending=False)
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
119	KNeighborsRegressor()	True	False	1.0	1.0	0.880996	0.053232	0.004845
7	None	True	False	1.0	1.0	0.861185	0.053359	0.005651
103	DecisionTreeRegressor()	True	False	1.0	1.0	0.861185	0.053359	0.005651
23	LinearRegression()	True	False	1.0	1.0	0.841424	0.062926	0.006456
71	Ridge()	True	False	1.0	1.0	0.834370	0.064097	0.006743
39	DummyRegressor()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
87	Lasso()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
55	QuantileRegressor()	True	False	1.0	1.0	-0.028829	0.168396	0.041886

Akurasi score

```
df_awal.sort_values(by=["MSE"])
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
119	KNeighborsRegressor()	True	False	1.0	1.0	0.880996	0.053232	0.004845
7	None	True	False	1.0	1.0	0.861185	0.053359	0.005651
103	DecisionTreeRegressor()	True	False	1.0	1.0	0.861185	0.053359	0.005651
23	LinearRegression()	True	False	1.0	1.0	0.841424	0.062926	0.006456
71	Ridge()	True	False	1.0	1.0	0.834370	0.064097	0.006743
39	DummyRegressor()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
87	Lasso()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
55	QuantileRegressor()	True	False	1.0	1.0	-0.028829	0.168396	0.041886

Akurasi dengan MSE

```
[189] df_awal.sort_values(by=["MAE"])
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
119	KNeighborsRegressor()	True	False	1.0	1.0	0.880996	0.053232	0.004845
7	None	True	False	1.0	1.0	0.861185	0.053359	0.005651
103	DecisionTreeRegressor()	True	False	1.0	1.0	0.861185	0.053359	0.005651
23	LinearRegression()	True	False	1.0	1.0	0.841424	0.062926	0.006456
71	Ridge()	True	False	1.0	1.0	0.834370	0.064097	0.006743
39	DummyRegressor()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
87	Lasso()	True	False	1.0	1.0	-0.001482	0.167412	0.040772
55	QuantileRegressor()	True	False	1.0	1.0	-0.028829	0.168396	0.041886

Akurasi dengan MAE

Dapat dilihat bahwa algoritma SVR adalah algoritma terbaik untuk bagging regressor dengan parameter default.

5.3. Tuning Hyperparameter Bagging Regressor

```
[178] hasil
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
0	None	True	True	0.5	0.5	0.868426	0.058287	0.005357
1	None	True	True	0.5	1.0	0.868438	0.055648	0.005356
2	None	True	True	1.0	0.5	0.877589	0.053991	0.004984
3	None	True	True	1.0	1.0	0.858754	0.058259	0.005750
4	None	True	False	0.5	0.5	0.867318	0.057315	0.005402
...	...	...	...	...	...	...	...	...
139	SVR()	False	True	1.0	1.0	0.908015	0.047912	0.003745
140	SVR()	False	False	0.5	0.5	0.875096	0.055404	0.005085
141	SVR()	False	False	0.5	1.0	0.889283	0.052722	0.004508
142	SVR()	False	False	1.0	0.5	0.908508	0.047410	0.003725
143	SVR()	False	False	1.0	1.0	0.913136	0.046469	0.003536

144 rows × 8 columns

Hasil

```
hasil.sort_values(by=["score"],ascending=False)
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
143	SVR()	False	False	1.0	1.0	0.913136	0.046469	0.003536
142	SVR()	False	False	1.0	0.5	0.908508	0.047410	0.003725
139	SVR()	False	True	1.0	1.0	0.908015	0.047912	0.003745
138	SVR()	False	True	1.0	0.5	0.903865	0.049046	0.003914
...	...	...	...	...	...	...	...	...
57	QuantileRegressor()	False	True	0.5	1.0	-0.067044	0.170031	0.043441
63	QuantileRegressor()	False	False	1.0	1.0	-0.067044	0.170031	0.043441
59	QuantileRegressor()	False	True	1.0	1.0	-0.067044	0.170031	0.043441
58	QuantileRegressor()	False	True	1.0	0.5	-0.068416	0.170120	0.043497
60	QuantileRegressor()	False	False	0.5	0.5	-0.068416	0.170120	0.043497

144 rows × 8 columns

Akurasi score

```
hasil.sort_values(by=["MAE"])
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
143	SVR()	False	False	1.0	1.0	0.913136	0.046469	0.003536
142	SVR()	False	False	1.0	0.5	0.908508	0.047410	0.003725
139	SVR()	False	True	1.0	1.0	0.908015	0.047912	0.003745
110	DecisionTreeRegressor()	False	False	1.0	0.5	0.895829	0.048241	0.004241
...	...	...	...	...	...	...	...	...
57	QuantileRegressor()	False	True	0.5	1.0	-0.067044	0.170031	0.043441
63	QuantileRegressor()	False	False	1.0	1.0	-0.067044	0.170031	0.043441
59	QuantileRegressor()	False	True	1.0	1.0	-0.067044	0.170031	0.043441
58	QuantileRegressor()	False	True	1.0	0.5	-0.068416	0.170120	0.043497
60	QuantileRegressor()	False	False	0.5	0.5	-0.068416	0.170120	0.043497

144 rows × 8 columns

Akurasi MAE

```
hasil.sort_values(by=["MSE"])
```

	Base Estimator	bootstrap	bootstrap_features	max_features	max_samples	score	MAE	MSE
135	SVR()	True	False	1.0	1.0	0.913433	0.046152	0.003524
143	SVR()	False	False	1.0	1.0	0.913136	0.046469	0.003536
142	SVR()	False	False	1.0	0.5	0.908508	0.047410	0.003725
139	SVR()	False	True	1.0	1.0	0.908015	0.047912	0.003745
138	SVR()	False	True	1.0	0.5	0.903865	0.049046	0.003914
...	...	...	...	...	...	...	...	...
57	QuantileRegressor()	False	True	0.5	1.0	-0.067044	0.170031	0.043441
63	QuantileRegressor()	False	False	1.0	1.0	-0.067044	0.170031	0.043441
59	QuantileRegressor()	False	True	1.0	1.0	-0.067044	0.170031	0.043441
58	QuantileRegressor()	False	True	1.0	0.5	-0.068416	0.170120	0.043497
60	QuantileRegressor()	False	False	0.5	0.5	-0.068416	0.170120	0.043497

144 rows × 8 columns

Akurasi MSE

Dapat dilihat bahwa dengan tuning hyperparameter untuk bagging regressor, SVR masih merupakan algoritma yang terbaik.

6. RESOURCES

Link Github	https://github.com/khalilullahalfaath/Project_Based_ML/blob/main/ML_Project_based.ipynb
Link Colab	https://colab.research.google.com/drive/1y_9TYBHfYRvfQjhUcWWhhOmvEdF2-baU?usp=sharing
Link docs	https://docs.google.com/document/d/1sTwCo4rVp0e5AjvYx5N7tiGE0HLZ5AHZQOBs2gys4Jk/edit?usp=sharing
Link Video	https://youtu.be/RzCjaJTozNw

7. KESIMPULAN

- Terdapat tiga macam metode evaluasi yang ada di sini, yaitu R2 Score, MSE, dan MAE.
- Tuning hyperparameter tidak terlalu berpengaruh pada kasus ini, sebab hyperparameter default sudah merupakan akurasi yang terbesar.
- Model yang paling cocok di kasus ini adalah SVR. Dibuktikan dengan akurasi yang konsisten maksimal di semua kondisi, baik itu ketika dilakukan tuning hyperparameter pada bagging regressor ataupun tidak. Dengan akurasi sebesar:

Score	MAE	MSE
0.913433	0.046152	0.003524

DAFTAR PUSTAKA

Ahmed, Mazen. "Ways to Evaluate your Regression Model | by Mazen Ahmed | Medium." *Mazen Ahmed*, 2021, https://linguisticmaz.medium.com/evaluating-regression-models-cb02ba075e16. Diakses pada 8 Januari 2023.

AnkanDas22. "Python | Decision Tree Regression using sklearn." *GeeksforGeeks*, 23 Agustus 2022, https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/. Diakses pada 8 Januari 2023.

Brownlee, Jason. "How to Develop a Bagging Ensemble with Python - MachineLearningMastery.com." *Machine Learning Mastery*, 27 April 2020, https://machinelearningmastery.com/bagging-ensemble-with-python/. Diakses pada 8 Januari 2023.

Frost, Jim. "Mean Squared Error (MSE) - Statistics By Jim." *Statistics by Jim*, https://statisticsbyjim.com/regression/mean-squared-error-mse/. Diakses pada 8 Januari 2023.

Gauss, Carl Friedrich. "Support Vector Regression (SVR) | Analytics Vidhya." *Medium*, 19 November 2020, https://medium.com/analytics-vidhya/support-vector-regression-svr-model-a-regression-based-machine-learning-approach-f4641670c5bb. Diakses pada 8 Januari 2023.

GISGeography. "How to Calculate Mean Absolute Error (MAE) in Excel - GIS Geography." *GISGeography*, 9 November 2022, https://gisgeography.com/mean-absolute-error-mae-gis/. Diakses pada 8 Januari 2023.

Gold, Aaron. “Cylinder? What's a Cylinder?” *VroomGirls*, https://www.vroomgirls.com/cylinder-whats-a-cylinder/. Diakses pada 8 Januari 2023.

Ismail, Jul. “Bagging dan Boosting.” *Jul Ismail*, 5 September 2021, https://julismail.staff.telkomuniversity.ac.id/bagging-dan-boosting/. Diakses pada 8 Januari 2023.

Kurama, Vihar. “Introduction to Bagging and Ensemble Methods.” *Paperspace Blog*, 2020, https://blog.paperspace.com/bagging-ensemble-methods/. Diakses pada 8 Januari 2023.

Meiryani. “MEMAHAMI R SQUARE (KOEFISIEN DETERMINASI) DALAM PENELITIAN ILMIAH.” *BINUS Accounting*, 12 Agustus 2021, https://accounting.binus.ac.id/2021/08/12/memahami-r-square-koefisien-determinasi-dalam-penelitian-ilmiah/. Diakses pada 8 Januari 2023.

Montoya, Anna. “House Prices - Advanced Regression Techniques.” *House Prices - Advanced Regression Techniques | Kaggle*, 2016, https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data. Diakses pada 8 Januari 2023.

NumPy. “NumPy documentation — NumPy v1.24 Manual.” *NumPy*, https://numpy.org/doc/stable/. Diakses pada 8 Januari 2023.

pandas. “pandas documentation — pandas 1.5.2 documentation.” *Pandas*, 22 November 2022, https://pandas.pydata.org/docs/. Diakses pada 8 Januari 2023.

Ried, Chris. “Demystifying R-Squared and Adjusted R-Squared | by KSV Muralidhar.” *Towards Data Science*, 27 Agustus 2021,

https://towardsdatascience.com/demystifying-r-squared-and-adjusted-r-squared-52903c006a60. Diakses pada 8 Januari 2023.

Rocca, Joseph. "Ensemble methods: bagging, boosting and stacking | by Joseph Rocca." *Towards Data Science*, 22 April 2019, https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205. Diakses pada 8 Januari 2023.

scikit-learn. "Machine Learning in Python." *scikit-learn: machine learning in Python — scikit-learn 1.2.0 documentation*, https://scikit-learn.org/stable/. Diakses pada 8 Januari 2023.

seaborn. "seaborn: statistical data visualization." *seaborn: statistical data visualization — seaborn 0.12.2 documentation*, https://seaborn.pydata.org/. Diakses pada 8 Januari 2023.

Tineges, Rian. "Kenali 4 Jenis Data Statistik Berikut Yuk." *DQLab*, 25 Agustus 2021, https://dqlab.id/kenali-4-jenis-data-statistik-berikut-yuk. Diakses pada 8 Januari 2023.

Vadapalli, Pavan. "6 Types of Regression Models in Machine Learning You Should Know About." *upGrad*, 4 Oktober 2022, https://www.upgrad.com/blog/types-of-regression-models-in-machine-learning/. Diakses pada 8 Januari 2023.