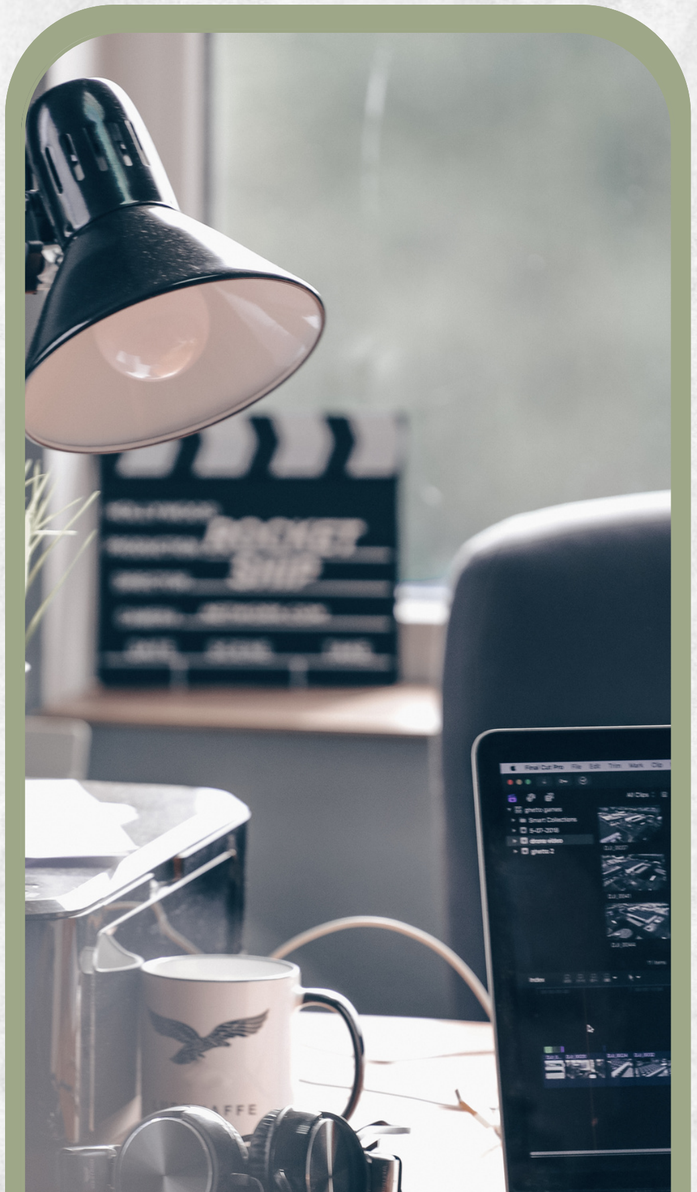
SPESIFIKASI MOTOR

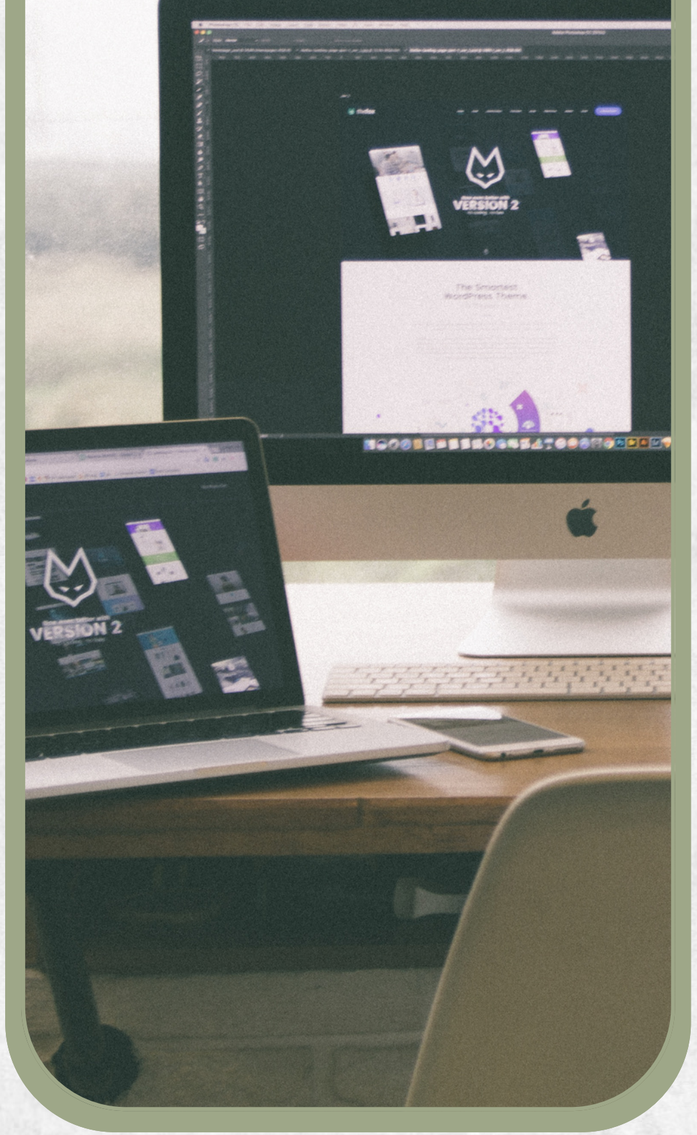
PROGRESS 1

KELOMPOK KAS-DAD

- Aulia Fikri Al Khawariz - 2106701160
- Kathleen Daniella Wijaya - 2106637366
- Stelline Claudia -2106700933

TABLE OF CONTENT





Introduction

Terdapat dataset Spesifikasi Motor, dimana dataset tersebut memiliki informasi mengenai merek motor, tahun produksi, rating, kategori, hingga spesifikasi detail seperti power, kapasitas mesin, dan lain-lain. Dari seluruh informasi dan data yang terdapat pada dataset tersebut, kami akan melakukan Data Preprocessing, Clustering, Classification, dan Regression demi membuat dua model untuk mengklasifikasi kategori dan memprediksi nilai kapasitas mesin suatu motor.

Dataset Summary

Dataset Description

Dataset ini berisi informasi tentang berbagai atribut dan spesifikasi dari motor-motor yang diproduksi oleh berbagai merek. Dataset ini terdiri dari nama brand dan model, tahun produksi, rating, power, kapasitas bahan bakar, hingga kategori motor.

Dataset Information

29966 Jumlah Baris

26 Kolom

10 Numerikal

16 Kategorikal

Data Type Information

```
RangeIndex: 6199 entries, 0 to 6198
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        6199 non-null   int64
 1   Model                     6199 non-null   object
 2   Tahun                     6199 non-null   int64
 3   Rating                    6199 non-null   float64
 4   Power                     6199 non-null   float64
 5   Kapasitas Bahan Bakar     6199 non-null   float64
 6   Jarak Sumbu Roda          6199 non-null   float64
 7   Ketinggian Tempat Duduk   6199 non-null   float64
 8   Bore                      6199 non-null   float64
 9   Stroke                    6198 non-null   float64
 10  Jenis Mesin               6198 non-null   object
 11  Jenis Bahan Bakar         6198 non-null   object
 12  Jenis Sistem Pendingin    6198 non-null   object
 13  Gearbox                   6198 non-null   object
 14  Jenis Transmisi           6198 non-null   object
 15  Suspensi Depan            6198 non-null   object
 16  Suspensi Belakang         6198 non-null   object
 17  Kompresi                  6198 non-null   object
 18  Ban Depan                 6198 non-null   object
 19  Ban Belakang              6198 non-null   object
 20  Rem Depan                 6198 non-null   object
 21  Rem Belakang              6198 non-null   object
 22  Warna                     6198 non-null   object
 23  Starter                   6198 non-null   object
 24  Kapasitas Mesin           6198 non-null   float64
 25  Kategori                  6198 non-null   object
dtypes: float64(8), int64(2), object(16)
memory usage: 1.2+ MB
```

Dataset Overview

	Id	Model	Tahun	Rating	Power	Kapasitas Bahan Bakar	Jarak Sumbu Roda	Ketinggian Tempat Duduk	Bore	Stroke	Jenis Mesin	Jenis Bahan Bakar	Jenis Sistem Pendingin
0	345	Adiva Birdie	2012	2.2	9.4	5.0	1400.0	800.0	54.0	66.0	Single cylinder, four-stroke	Carburettor	Air
1	35956	Yamaha SZR 660	1997	3.7	48.0	17.0	1400.0	690.0	54.0	66.0	Single cylinder, four-stroke	Carburettor	Liquid
2	23521	Modenas Kriss 100	2011	3.0	27.0	4.3	1400.0	800.0	50.0	49.5	Single cylinder, four-stroke	Carburettor. Keihin PB18X1	Air
3	32412	Triumph Speed Twin	1956	3.5	27.0	17.0	1400.0	800.0	54.0	66.0	Twin, four-stroke	Carburettor. Amal	Air
4	31061	Sym Mio 100 (Disk Type)	2007	2.8	27.0	4.8	1220.0	800.0	54.0	66.0	Single cylinder, four-stroke	Carburettor	Air

	Gearbox	Jenis Transmisi	Suspensi Depan	Suspensi Belakang	Kompresi	Ban Depan	Ban Belakang	Rem Depan	Rem Belakang	Warna	Starter	Kapasitas Mesin	Kategori
0	6-speed	Chain (final drive)	Telescopic fork	Monoshock	9.5:1	3.50-10	3.50-10	Expanding brake (drum brake)	Expanding brake (drum brake)	Red/white	Electric	124.6	Classic
1	5-speed	Chain (final drive)	Telescopic fork	Monoshock	9.5:1	120/70-ZR17	180/55-ZR17	Single disc	Single disc	Black	Electric	660.0	Classic
2	4-speed	Belt (final drive)	Telescopic	Trailing swing arm	9.0:1	120/70-ZR17	180/55-ZR17	Expanding brake (drum brake)	Expanding brake (drum brake)	Blue, black, red	Kick	97.2	Classic
3	6-speed	Chain (final drive)	Telescopic	Rigid-optional sprung hub	9.5:1	120/70-ZR17	180/55-ZR17	Expanding brake (drum brake)	Expanding brake (drum brake)	Black	Electric	498.0	Classic
4	6-speed	Chain (final drive)	Telescopic fork	Monoshock	9.5:1	120/70-ZR17	180/55-ZR17	Single disc	Expanding brake (drum brake)	Black, Green/White, Orange/White, Pink/White, ...	Electric & kick	100.0	Classic

Column Description

- **Model** : Nama brand dan model dari motor
- **Tahun** : Tahun motor diproduksi
- **Rating** : Rating motor
- **Power** : Output power maksimum dalam Horsepower (HP)
- **Kapasitas Bahan Bakar** : Kapasitas Bahan Bakar Maksimum dalam Liter (Liter)
- **Jarak Sumbu Roda** : Jarak antara ban depan dan ban belakang dalam milimiter (mm)
- **Ketinggian Tempat Duduk** : Ketinggian tempat duduk dari bawah tempat duduk hingga dasar dalam milimeter (mm)
- **Bore** : Diameter silinder piston dalam milimeter (mm)
- **Stroke** : Jarak silinder piston bergerak dalam milimeter (mm)
- **Jenis Mesin** : Jenis mesin yang digunakan oleh motor
- **Jenis Bahan Bakar** : Jenis bahan bakar yang digunakan oleh motor
- **Jenis Sistem Pendingin** : Jenis sistem pendingin yang digunakan oleh motor
- **Gearbox** : Banyaknya gear dalam gearbox.

Column Description

- **Jenis Transmisi** : Jenis transmisi yang digunakan oleh motor
- **Suspensi Depan** : Jenis dan konfigurasi suspensi bagian depan
- **Suspensi Belakang** : Jenis dan konfigurasi suspensi bagian belakang
- **Kompresi** : Rasio kompresi yang digunakan oleh motor
- **Ban Depan** : Jenis dan ukuran yang digunakan oleh ban depan
- **Ban Belakang** : Jenis dan ukuran yang digunakan oleh ban belakang
- **Rem Depan** : Jenis rem depan yang digunakan
- **Rem Belakang** : Jenis rem belakang yang digunakan
- **Warna** : Warna yang digunakan pada motor
- **Starter** : Jenis starter yang digunakan untuk menyalakan motor
- **Kapasitas Mesin** : Ukuran kapasitas mesin dalam centimeter cubic (cc)
- **Kategori** : Kategori motor

Task

Tunjukkan brand yang memiliki motor rating terbagus?

```
df['Brand'] = df['Model'].apply(lambda x: x.split()[0])
rata_rata_per_brand = df.groupby('Brand')['Rating'].mean()
brand_rating_terbagus = rata_rata_per_brand.sort_values(ascending=False).index[0]
print(f"Brand yang memiliki motor dengan rata-rata rating tertinggi adalah: {brand_rating_terbagus}")
```

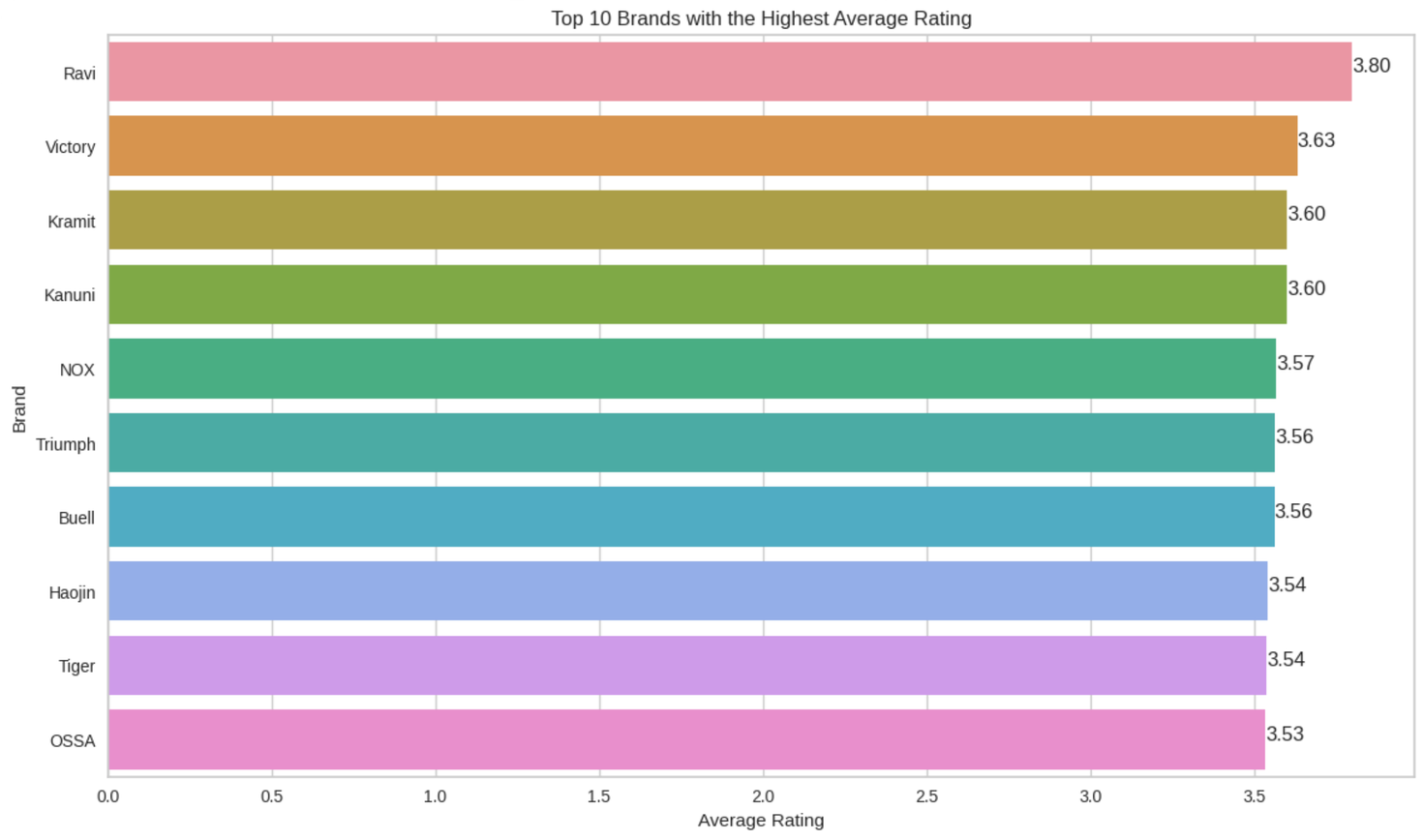
```
Brand yang memiliki motor dengan rata-rata rating tertinggi adalah: Ravi
```

Tunjukkan brand yang memiliki motor rating terbagus?

```
brand_rating = pd.DataFrame({'Brand': rata_rata_per_brand.index, 'Average Rating': rata_rata_per_brand.values})
brand_rating = brand_rating.sort_values(by='Average Rating', ascending=False)
top_10 = brand_rating.head(10)
plt.figure(figsize=(14, 8))
sns.barplot(x='Average Rating', y='Brand', data= top_10.head(10), label='Average Rating')
for index, value in enumerate(top_10['Average Rating']):
    plt.text(value, index, f'{value:.2f}')

plt.title('Top 10 Brands with the Highest Average Rating')
plt.xlabel('Average Rating')
plt.ylabel('Brand')
plt.show()
```

Tunjukkan brand yang memiliki motor rating terbagus?

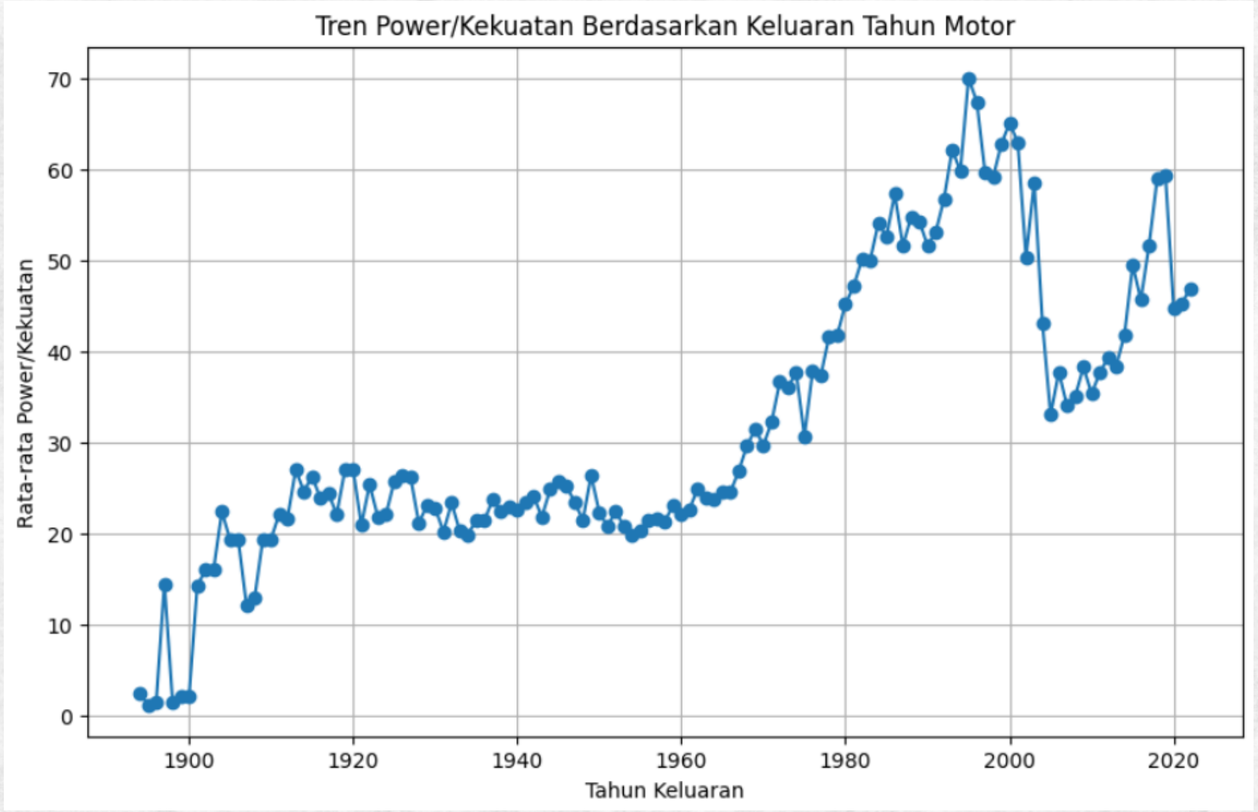


Tunjukkan tren power/kekuatan berdasarkan keluaran tahun motor?

```
tren_power = df.groupby("Tahun")["Power"].mean()

plt.figure(figsize=(10, 6))
plt.plot(tren_power.index, tren_power.values, marker='o', linestyle='-')
plt.title('Tren Power/Kekuatan Berdasarkan Keluaran Tahun Motor')
plt.xlabel('Tahun Keluaran')
plt.ylabel('Rata-rata Power/Kekuatan')
plt.grid(True)
plt.show()
```

Tunjukkan tren power/kekuatan berdasarkan keluaran tahun motor?



Kategori motor apa yang paling banyak diproduksi di setiap tahunnya?

```
kategori_paling_banyak_diproduksi = df.groupby(['Tahun', 'Kategori']).size()
kategori_paling_banyak_diproduksi = kategori_paling_banyak_diproduksi.loc[kategori_paling_banyak_diproduksi.groupby('Tahun').idxmax()]
print(kategori_paling_banyak_diproduksi.reset_index()[['Tahun', 'Kategori']].head(20))
```

```
    Tahun Kategori
0    1894  Classic
1    1895  Classic
2    1896  Classic
3    1897  Classic
4    1898  Classic
5    1899  Classic
6    1900  Classic
7    1901  Classic
8    1902  Classic
9    1903  Classic
10   1904  Classic
11   1905    Sport
12   1906    Sport
13   1907    Sport
14   1908    Sport
15   1909    Sport
16   1910    Sport
17   1911    Sport
18   1912  Classic
19   1913  Classic
```

Kategori motor apa yang paling banyak diproduksi di setiap tahunnya?

	Tahun	Kategori
20	1914	Sport
21	1915	Classic
22	1916	Classic
23	1917	Classic
24	1918	Classic
25	1919	Sport
26	1920	Sport
27	1921	Sport
28	1922	Sport
29	1923	Sport
30	1924	Sport
31	1925	Sport
32	1926	Sport
33	1927	Sport
34	1928	Sport
35	1929	Classic
36	1930	Classic
37	1931	Classic
38	1932	Classic
39	1933	Classic
40	1934	Classic
41	1935	Classic
42	1936	Classic
43	1937	Classic
44	1938	Classic
45	1939	Classic
46	1940	Classic
47	1941	Classic
48	1942	Classic
49	1943	Classic

	Tahun	Kategori
50	1944	Classic
51	1945	Classic
52	1946	Classic
53	1947	Classic
54	1948	Classic
55	1949	Classic
56	1950	Classic
57	1951	Classic
58	1952	Classic
59	1953	Classic
60	1954	Classic
61	1955	Classic
62	1956	Classic
63	1957	Classic
64	1958	Classic
65	1959	Classic
66	1960	Classic
67	1961	Classic
68	1962	Classic
69	1963	Classic
70	1964	Classic
71	1965	Classic
72	1966	Classic
73	1967	Classic
74	1968	Classic
75	1969	Classic
76	1970	Classic
77	1971	Classic
78	1972	Classic
79	1973	Classic

	Tahun	Kategori
80	1974	Classic
81	1975	Classic
82	1976	Classic
83	1977	Classic
84	1978	Classic
85	1979	Classic
86	1980	Classic
87	1981	Classic
88	1982	Classic
89	1983	Classic
90	1984	Sport
91	1985	Sport
92	1986	Sport
93	1987	Sport
94	1988	Sport
95	1989	Sport
96	1990	Sport
97	1991	Sport
98	1992	Sport
99	1993	Sport
100	1994	Sport
101	1995	Sport
102	1996	Sport
103	1997	Classic
104	1998	Classic
105	1999	Sport
106	2000	Classic
107	2001	Classic
108	2002	Offroad
109	2003	Classic

	Tahun	Kategori
110	2004	Offroad
111	2005	Offroad
112	2006	Classic
113	2007	Classic
114	2008	Classic
115	2009	Classic
116	2010	Classic
117	2011	Classic
118	2012	Classic
119	2013	Classic
120	2014	Classic
121	2015	Classic
122	2016	Classic
123	2017	Classic
124	2018	Classic
125	2019	Classic
126	2020	Classic
127	2021	Classic
128	2022	Classic

Apa model motor scooter dengan rating terbaik yang diproduksi oleh lima brand teratas yang memproduksi motor paling banyak?

```
lima_brand_teratas = df['Model'].apply(lambda x: x.split()[0]).value_counts().nlargest(5).index
filtered_df = df[df['Model'].apply(lambda x: x.split()[0] in lima_brand_teratas)]
scooter_dengan_rating_terbaik = filtered_df[filtered_df['Model'].str.contains('Scooter')].groupby('Model')['Rating'].max()
print(scooter_dengan_rating_terbaik.reset_index())
```

```
                            Model  Rating
0  Harley-Davidson AH Topper-Scooter     3.5
1                 Honda BeAT Scooter     3.9
```

Apa jenis sistem pendingin yang paling umum digunakan untuk masing-masing kategori motor?

```
jenis_sistem_pendingin = df.groupby('Kategori')['Jenis Sistem Pendingin'].agg(pd.Series.mode)

print("Jenis sistem pendingin yang paling umum digunakan untuk masing-masing kategori motor:")
print(jenis_sistem_pendingin)
```

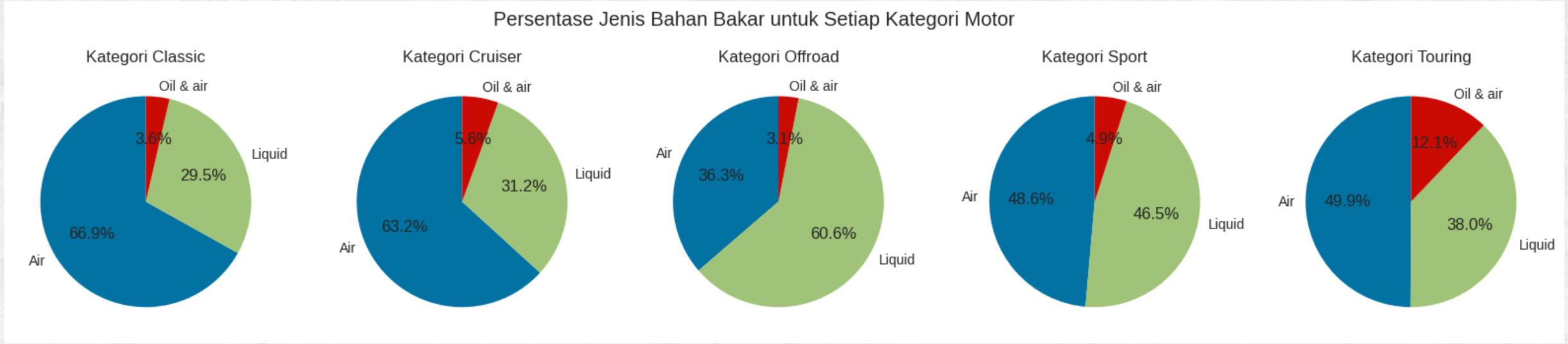
```
Jenis sistem pendingin yang paling umum digunakan untuk masing-masing kategori motor:
Kategori
Classic       Air
Cruiser       Air
Offroad    Liquid
Sport         Air
Touring       Air
Name: Jenis Sistem Pendingin, dtype: object
```

Apa jenis sistem pendingin yang paling umum digunakan untuk masing-masing kategori motor?

```
data = df.groupby(['Kategori', 'Jenis Sistem Pendingin']).size().unstack().reset_index()
persentase_jenis_sistem_pendingin = data.set_index('Kategori').div(data.set_index('Kategori').sum(axis=1), axis=0) * 100
persentase_jenis_sistem_pendingin = persentase_jenis_sistem_pendingin.reset_index()
jenis_sistem_pendingin = data.columns[1:]
fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(20, 4), sharey=True)
for i, (kategori, data) in enumerate(persentase_jenis_sistem_pendingin.groupby('Kategori')):
    ax = axes[i]
    data = data.dropna(subset=jenis_sistem_pendingin)
    if not data.empty:
        ax.pie(data[jenis_sistem_pendingin].iloc[0], labels=jenis_sistem_pendingin, autopct='%1.1f%%', startangle=90)
        ax.set_title(f'Kategori {kategori}')

plt.suptitle('Persentase Jenis Bahan Bakar untuk Setiap Kategori Motor')
plt.show()
```

Apa jenis sistem pendingin yang paling umum digunakan untuk masing-masing kategori motor?



Tunjukkan rata-rata kapasitas mesin untuk setiap jenis mesin dan urutkan dari yang terbesar hingga terkecil

```
rata_rata_kapasitas = df.groupby('Jenis Mesin')['Kapasitas Mesin'].mean().reset_index()
urutan_rata_rata = rata_rata_kapasitas.sort_values(by='Kapasitas Mesin', ascending=False)

print(urutan_rata_rata)
```

```
                        Jenis Mesin  Kapasitas Mesin
28                   V8, four-stroke      5952.877612
16   Six cylinder boxer, four-stroke      1773.735849
27                   V6, four-stroke      1329.272727
8          In-line six, four-stroke      1321.624324
22                   V2, four-stroke      1125.752853
3   Four cylinder boxer, four-stroke      1080.121212
25                   V4, four-stroke      1054.483090
10        In-line three, four-stroke       958.931006
21    Two cylinder boxer, two-stroke       934.677419
17              Square four cylinder       931.000000
6          In-line four, four-stroke       893.081770
20   Two cylinder boxer, four-stroke       803.694888
```

Tunjukkan rata-rata kapasitas mesin untuk setiap jenis mesin dan urutkan dari yang terbesar hingga terkecil

0	Diesel	763.379310
1	Dual disk Wankel	690.250000
18	Twin, four-stroke	610.977758
23	V2, two-stroke	570.007143
4	Four cylinder boxer, two-stroke	550.000000
26	V4, two-stroke	512.566667
11	In-line three, two-stroke	494.717391
7	In-line four, two-stroke	481.038462
24	V3, two-stroke	387.000000
15	Single disk Wankel	361.666667
19	Twin, two-stroke	338.217465
13	Single cylinder, four-stroke	256.149639
12	Radial	196.428571
2	Electric	126.257534
5	Gas turbine	125.000000
14	Single cylinder, two-stroke	123.395398
9	In-line six, two-stroke	59.333333

Pre Processing

Pengecekan Nilai Null

```python
# mengecek nilai null
def check_null(df):
    col_na = df.isnull().sum().sort_values(ascending=True)
    percent = col_na / len(df)
    missing_data = pd.concat([col_na, percent], axis=1, keys=['Total', 'Percent'])

    if (missing_data[missing_data['Total'] > 0].shape[0] == 0):
        print("Tidak ditemukan missing value pada dataset")
    else:
        print(missing_data[missing_data['Total'] > 0])
```

```
[76] check_null(motor)
```

```
Tidak ditemukan missing value pada dataset
```

Pengecekan Nilai Duplikat

```
print("Jumlah duplikasi data : " + str(motor.duplicated().sum()))
```

```
Jumlah duplikasi data : 0
```

Pengecekan Nilai Outlier

```python
def check_outlier(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)

    # Menghitung RUB dan RLB.
    IQR = Q3 - Q1
    lower_limit = Q1 - 1.5*IQR
    upper_limit = Q3 + 1.5*IQR

    # Menampilkan banyaknya outlier pada atribut.
    outliers = (df < lower_limit) | (df > upper_limit)
    print ("Outlier pada tiap atribut:")
    print(outliers.sum())

    return outliers
```

```
Outlier pada tiap atribut:
Ban Belakang                  0
Ban Depan                     0
Bore                          6
Gearbox                       0
Id                            0
Jarak Sumbu Roda           4170
Jenis Bahan Bakar             0
Jenis Mesin                   0
Jenis Sistem Pendingin        0
Jenis Transmisi               0
Kapasitas Bahan Bakar       213
Kapasitas Mesin             626
Kategori                      0
Ketinggian Tempat Duduk    8748
Kompresi                      0
Model                         0
Power                      3340
Rating                     9659
Rem Belakang                  0
Rem Depan                     0
Starter                       0
Stroke                     2611
Suspensi Belakang             0
Suspensi Depan                0
Tahun                      3382
Warna                         0
```

Percobaan Penanganan Nilai Outlier

```python
def cek_outlier(df, col):
  Q1 = df[col].quantile(0.25)
  Q3 = df[col].quantile(0.75)
  IQR = Q3 - Q1

  lower = Q1 - 1.5*IQR
  upper = Q3 + 1.5*IQR

  return IQR, lower, upper
```

```python
def handle_outlier_mean(df, col):
  mean =  df.loc[:, col].mean()
  df[col] = np.where(df[col] > cek_outlier(df, col)[2], mean,
                     np.where(df[col] < cek_outlier(df, col)[1], mean, df[col]))
```

```python
def handle_outlier_median(df, col):
  median = df.loc[:, col].median()
  df[col] = np.where(df[col] > cek_outlier(df, col)[2], median,
                     np.where(df[col] < cek_outlier(df, col)[1], median, df[col]))
```

Melakukan encoding pada data

```
columns_obj = ['Model', 'Jenis Mesin','Jenis Bahan Bakar','Jenis Sistem Pendingin','Gearbox','Jenis Transmisi','Suspensi Depan',
               'Suspensi Belakang','Kompresi','Ban Depan','Ban Belakang','Rem Depan','Rem Belakang','Warna','Starter', 'Kategori']
from sklearn.preprocessing import LabelEncoder
columns_to_encode = [column for column in motor.columns if column in columns_obj ]
encoder = LabelEncoder()
for col in columns_to_encode :
  motor[col] = encoder.fit_transform(motor[col])

motor
```

	Id	Model	Tahun	Rating	Power	Kapasitas Bahan Bakar	Jarak Sumbu Roda	Ketinggian Tempat Duduk	Bore	Stroke	...	Suspensi Belakang	Kompresi	Ban Depan	Ban Belakang	Rem Depan	Rem Belakang	Warna	Starter	Kapasitas Mesin	Kategori
0	345	165	2012	2.2	9.4	5.00	1400.0	800.0	54.0	66.0	...	1739	105	757	1016	857	27	5930	0	124.6	0
1	35956	15075	1997	3.7	48.0	17.00	1400.0	690.0	54.0	66.0	...	1739	105	317	625	940	148	263	0	660.0	0
2	23521	9691	2011	3.0	27.0	4.30	1400.0	800.0	50.0	49.5	...	3472	100	317	625	857	27	1630	2	97.2	0
3	32412	13561	1956	3.5	27.0	17.00	1400.0	800.0	54.0	66.0	...	2516	105	317	625	857	27	263	0	498.0	0
4	31061	12886	2007	2.8	27.0	4.80	1220.0	800.0	54.0	66.0	...	1739	105	317	625	940	27	582	1	100.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29961	27123	11365	2016	3.5	27.0	12.00	1350.0	800.0	52.0	58.6	...	1706	105	58	292	940	148	5676	0	125.0	3
29962	36003	15095	2016	3.5	27.0	12.11	1570.0	765.0	85.0	83.0	...	842	100	93	473	1755	1296	2550	0	950.5	1
29963	27744	11658	2020	3.5	24.1	16.00	2010.0	800.0	54.0	66.0	...	1739	105	714	102	940	148	1400	0	249.0	3
29964	8380	3623	2020	3.5	14.8	6.30	1425.0	980.0	58.0	47.0	...	2803	21	58	258	1344	742	7639	0	124.2	2
29965	12348	5265	2010	3.8	123.4	17.00	1445.0	825.0	75.0	56.5	...	1798	12	317	625	89	287	4846	0	998.0	3

29966 rows × 26 columns

Melakukan encoding pada data

```
RangeIndex: 6199 entries, 0 to 6198
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        6199 non-null   int64
 1   Model                     6199 non-null   object
 2   Tahun                     6199 non-null   int64
 3   Rating                    6199 non-null   float64
 4   Power                     6199 non-null   float64
 5   Kapasitas Bahan Bakar     6199 non-null   float64
 6   Jarak Sumbu Roda          6199 non-null   float64
 7   Ketinggian Tempat Duduk   6199 non-null   float64
 8   Bore                      6199 non-null   float64
 9   Stroke                    6198 non-null   float64
 10  Jenis Mesin               6198 non-null   object
 11  Jenis Bahan Bakar         6198 non-null   object
 12  Jenis Sistem Pendingin    6198 non-null   object
 13  Gearbox                   6198 non-null   object
 14  Jenis Transmisi           6198 non-null   object
 15  Suspensi Depan            6198 non-null   object
 16  Suspensi Belakang         6198 non-null   object
 17  Kompresi                  6198 non-null   object
 18  Ban Depan                 6198 non-null   object
 19  Ban Belakang              6198 non-null   object
 20  Rem Depan                 6198 non-null   object
 21  Rem Belakang              6198 non-null   object
 22  Warna                     6198 non-null   object
 23  Starter                   6198 non-null   object
 24  Kapasitas Mesin           6198 non-null   float64
 25  Kategori                  6198 non-null   object
dtypes: float64(8), int64(2), object(16)
memory usage: 1.2+ MB
```

```
Data columns (total 26 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        29966 non-null  int64
 1   Model                     29966 non-null  int64
 2   Tahun                     29966 non-null  int64
 3   Rating                    29966 non-null  float64
 4   Power                     29966 non-null  float64
 5   Kapasitas Bahan Bakar     29966 non-null  float64
 6   Jarak Sumbu Roda          29966 non-null  float64
 7   Ketinggian Tempat Duduk   29966 non-null  float64
 8   Bore                      29966 non-null  float64
 9   Stroke                    29966 non-null  float64
 10  Jenis Mesin               29966 non-null  int64
 11  Jenis Bahan Bakar         29966 non-null  int64
 12  Jenis Sistem Pendingin    29966 non-null  int64
 13  Gearbox                   29966 non-null  int64
 14  Jenis Transmisi           29966 non-null  int64
 15  Suspensi Depan            29966 non-null  int64
 16  Suspensi Belakang         29966 non-null  int64
 17  Kompresi                  29966 non-null  int64
 18  Ban Depan                 29966 non-null  int64
 19  Ban Belakang              29966 non-null  int64
 20  Rem Depan                 29966 non-null  int64
 21  Rem Belakang              29966 non-null  int64
 22  Warna                     29966 non-null  int64
 23  Starter                   29966 non-null  int64
 24  Kapasitas Mesin           29966 non-null  float64
 25  Kategori                  29966 non-null  int64
dtypes: float64(8), int64(18)
```

Data Modelling

Classification

Model untuk mengklasifikasi
Kategori suatu motor

Classification

Model untuk mengklasifikasi Kategori suatu motor

Untuk sementara data yang di train adalah data yang bersifat numerikal, data yang kategorikal belum diolah. Kemudian penanganan data outlier belum di tangani pada pembuatan model ini

```
drop_col = [key for key, val in outliers.items() if val == 0]
drop_col
```

```
['Ban Belakang',
 'Ban Depan',
 'Gearbox',
 'Id',
 'Jenis Bahan Bakar',
 'Jenis Mesin',
 'Jenis Sistem Pendingin',
 'Jenis Transmisi',
 'Kategori',
 'Kompresi',
 'Model',
 'Rem Belakang',
 'Rem Depan',
 'Starter',
 'Suspensi Belakang',
 'Suspensi Depan',
 'Warna']
```

Classification dengan Random Forest

```
# Grid Search untuk Random Forest Classifier
rf_classifier = RandomForestClassifier(max_depth=30, random_state=42)
rf_classifier.fit(x_train_motor, y_train_motor)
```

RandomForestClassifier

RandomForestClassifier(max_depth=30, random_state=42)

```
predicted = rf_classifier.predict(x_test_motor)
classification_metrics(predicted, y_test_motor)
```

```
Accuracy: 0.87620954287620960
F1 Score: 0.8390710564817067
Recall Score: 0.8235919918453529
Precision Score: 0.8576587137918261
```

Classification dengan Naive Bayes

Sebelum membuat model, kami membagi data menjadi data yang bersifat diskrit (kategorikal) dan data yang bersifat kontinu (numerik).

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB, MultinomialNB, CategoricalNB
from sklearn.model_selection import train_test_split, KFold, cross_val_score
discrete_cols   = ['Model', 'Jenis Mesin','Jenis Bahan Bakar','Jenis Sistem Pendingin','Gearbox','Jenis Transmisi',
                   'Suspensi Depan', 'Suspensi Belakang','Kompresi','Ban Depan','Ban Belakang','Rem Depan','Rem Belakang','Warna','Starter']
continuous_cols = [col for col in motor.columns if col not in discrete_cols and col != 'Kategori']

X_train_continuous = x_train_motor[continuous_cols]
X_train_discrete   = x_train_motor[discrete_cols]

X_test_continuous = x_test_motor[continuous_cols]
X_test_discrete   = x_test_motor[discrete_cols]
```

Classification dengan Naive Bayes

Kemudian, kami juga melakukan standarisasi pada data dan membangun model untuk data yang bersifat diskrit dan kontinu

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_continuous)
X_test_scaled = scaler.transform(X_test_continuous)
```

```
model_gnb = GaussianNB() # For numeric
model_gnb.fit(X_train_scaled, y_train_motor)
```

```
▾ GaussianNB
GaussianNB()
```

```
model_mnb = CategoricalNB() # For categorical
model_mnb.fit(X_train_discrete, y_train_motor)
```

```
▾ CategoricalNB
CategoricalNB()
```

Classification dengan Naive Bayes

```
categorical_posteriors = model_mnb.predict_proba(X_test_discrete)
numerical_posteriors   = model_gnb.predict_proba(X_test_scaled)

combined_probabilities = categorical_posteriors * numerical_posteriors

# Get the class with the highest combined probability as the final prediction
final_predictions = np.argmax(combined_probabilities, axis=1)
final_predictions
```

```
array([2, 3, 0, ..., 1, 3, 3])
```

Hasil Evaluasi berdasarkan classification report

	precision	recall	f1-score	support
0	0.74	0.81	0.78	2392
1	0.81	0.72	0.76	666
2	0.90	0.83	0.86	1560
3	0.62	0.65	0.63	1125
4	0.68	0.43	0.53	251
accuracy			0.76	5994
macro avg	0.75	0.69	0.71	5994
weighted avg	0.76	0.76	0.76	5994

Confusion Matrix

prediction actual	0	1	2	3	4
0	1949	30	63	333	17
1	141	477	17	19	12
2	193	4	1292	57	14
3	308	23	61	726	7
4	44	56	4	39	108

Butuh informasi lebih lengkap? silakan simak di bawah ini :
Accuracy Average: 0.7594260927594261
F1 Macro Average: 0.7113717493615223
F1 Micro Average: 0.759426092759426
Precision Macro Average: 0.7498342597476227
Precision Micro Average: 0.7594260927594261
Recall Macro Average: 0.6869665786641017
Recall Micro Average: 0.7594260927594261

Classification dengan KNN

Sebelum membuat model, kami melakukan Hyperparameter tuning

```
knn_kfold = KFold(n_splits=7, shuffle=True, random_state=42)
```

```
tunned_params = [{'n_neighbors': [1,2,3,4,5,6,7,8],'metric': ['minkowski', 'euclidean', 'manhattan', 'jaccard']}]
model_knn = KNeighborsClassifier()

print("Tuning hyper-parameters model KNN")
best_knn = GridSearchCV(
    model_knn,
    tunned_params,
    scoring="accuracy",
    cv=knn_kfold,
    verbose = 3
)

best_knn.fit(X_train_scaled, y_train_motor)
```

Classification dengan KNN

```
print("Best parameters set found on development set:")
print(best_knn.best_params_)
```

```
Best parameters set found on development set:
{'metric': 'manhattan', 'n_neighbors': 3}
```

```
best_knn_model = KNeighborsClassifier(**best_knn.best_params_)
```

```
best_knn_model.fit(X_train_scaled, y_train_motor)
```

KNeighborsClassifier

```
KNeighborsClassifier(metric='manhattan', n_neighbors=3)
```

```
Detailed classification report:
Hasil Evaluasi berdasarkan classification report

              precision    recall  f1-score   support

           0       0.79      0.87      0.83      2392
           1       0.84      0.79      0.81       666
           2       0.91      0.86      0.88      1560
           3       0.78      0.71      0.74      1125
           4       0.63      0.51      0.56       251

    accuracy                           0.81      5994
   macro avg       0.79      0.75      0.77      5994
weighted avg       0.81      0.81      0.81      5994


Confusion Matrix
```

prediction actual	0	1	2	3	4
0	2088	30	82	174	18
1	78	526	11	15	36
2	183	9	1336	27	5
3	248	19	38	802	18
4	54	44	9	15	129

```
Butuh informasi lebih lengkap? silakan simak di bawah ini :
Accuracy Average: 0.8143143143143143
F1 Macro Average: 0.7657998325752141
F1 Micro Average: 0.8143143143143143
Precision Macro Average: 0.7865898098515066
Precision Micro Average: 0.8143143143143143
Recall Macro Average: 0.7491885714386319
Recall Micro Average: 0.8143143143143143
```

Regression

Model untuk memprediksi nilai kapasitas mesin dari suatu motor.

Regression dengan Random Forest

```
rf_regression = RandomForestRegressor(max_depth=30, random_state=42)
rf_regression.fit(X_train, y_train)
```

RandomForestRegressor

RandomForestRegressor(max_depth=30, random_state=42)

```
predicted = rf_regression.predict(X_test)
regression_metrics(predicted, y_test)
```

```
MAE: 18.315703274114792
MSE: 5429.146954258717
RMSE: 73.68274529534521
R_squared: 0.982774097109436
```

Regression dengan Linear Regression

```
linreg = LinearRegression()
linreg.fit(X_train, y_train)

y_pred = linreg.predict(X_test)
y_pred
```

```
array([1441.07943499,  209.81698472,  569.60204938, ..., 1549.11394441,
        155.54418555,  840.92613016])
```

```
 metrics(y_test, y_pred)
```

```
MAE: 173.33106607182737
MSE: 89747.17780696566
RMSE: 299.57833334032296
R_squared: 0.715245105238443
```

Regression dengan Lasso Regression

```
lasso_alpha_list = [1, 2, 3, 4, 5]
lasso_model_list = []
for alpha in lasso_alpha_list:
    lasso = Lasso(alpha=alpha)
    lasso.fit(X_train, y_train)
    lasso_model_list.append(lasso)
```

```
counter = 1
for model in lasso_model_list:
    y_pred = model.predict(X_test)
    print(f'Model Lasso regression ke {counter}')
    metrics(y_test, y_pred)
    print()
    counter += 1
```

```
Model Lasso regression ke 1
MAE: 173.68404330347707
MSE: 85613.47405884329
RMSE: 292.59780255299813
R_squared: 0.7283607530452609

Model Lasso regression ke 2
MAE: 177.36308944367605
MSE: 85732.50940667483
RMSE: 292.8011431102598
R_squared: 0.7279830709969455

Model Lasso regression ke 3
MAE: 184.4083930861308
MSE: 90526.5008628868
RMSE: 300.8762218303181
R_squared: 0.7127724252032959

Model Lasso regression ke 4
MAE: 190.25956116923714
MSE: 96438.61481136453
RMSE: 310.5456726656556
R_squared: 0.694014137462615

Model Lasso regression ke 5
MAE: 193.96110023120235
MSE: 101312.0436996802
RMSE: 318.295528871645
R_squared: 0.6785514481153789
```

Regression dengan Ridge Regression

```
ridge_alpha_list = [10, 12, 14, 16, 18]
ridge_model_list = []
for alpha in ridge_alpha_list:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train, y_train)
    ridge_model_list.append(ridge)
```

```
counter = 1
for model in ridge_model_list:
    y_pred = model.predict(X_test)
    print(f'Model Ridge regression ke {counter}')
    metrics(y_test, y_pred)
    print()
    counter += 1
```

```
Model Ridge regression ke 1
MAE: 175.3471583479281
MSE: 83915.29707705024
RMSE: 289.6813716431387
R_squared: 0.7337488245094921

Model Ridge regression ke 2
MAE: 175.81135002123486
MSE: 83885.15315275527
RMSE: 289.6293375208307
R_squared: 0.7338444668483384

Model Ridge regression ke 3
MAE: 176.25585590728735
MSE: 83970.17055563224
RMSE: 289.7760696738643
R_squared: 0.7335747188495634

Model Ridge regression ke 4
MAE: 176.68565542580276
MSE: 84134.65581574249
RMSE: 290.05974525215055
R_squared: 0.7330528307626392

Model Ridge regression ke 5
MAE: 177.10934778290988
MSE: 84355.23117356948
RMSE: 290.43972037854854
R_squared: 0.7323529768581507
```

Clustering

Melakukan clustering motor berdasarkan karakteristik performanya.

Menghandle outliers dengan metode capping

```
motor = motor.apply(lambda x: x.clip(lower=lower_limit[x.name], upper=upper_limit[x.name]))
outliers = dict(check_outlier(motor).sum())
```

```
Outlier setiap atribut:

Id                          0
Model                       0
Tahun                    3382
Rating                   9659
Power                    3340
Kapasitas Bahan Bakar     213
Jarak Sumbu Roda         4170
Ketinggian Tempat Duduk  8748
Bore                        6
Stroke                   2611
Jenis Mesin               923
Jenis Bahan Bakar           0
Jenis Sistem Pendingin      0
Gearbox                  5540
Jenis Transmisi          9010
Suspensi Depan              6
Suspensi Belakang         131
Kompresi                    0
Ban Depan                   0
Ban Belakang                0
Rem Depan               10331
Rem Belakang             4982
Warna                       0
Starter                  7355
Kapasitas Mesin           626
Kategori                    0
dtype: int64
```

```
Outlier pada tiap atribut:
Id                          0
Model                       0
Tahun                       0
Rating                      0
Power                       0
Kapasitas Bahan Bakar       0
Jarak Sumbu Roda            0
Ketinggian Tempat Duduk     0
Bore                        0
Stroke                      0
Jenis Mesin                 0
Jenis Bahan Bakar           0
Jenis Sistem Pendingin      0
Gearbox                     0
Jenis Transmisi             0
Suspensi Depan              0
Suspensi Belakang           0
Kompresi                    0
Ban Depan                   0
Ban Belakang                0
Rem Depan                   0
Rem Belakang                0
Warna                       0
Starter                     0
Kapasitas Mesin             0
Kategori                    0
dtype: int64
```

Melakukan standarisasi

```
scaler = MinMaxScaler()
cols = [column for column in motor ]
motor[cols] = scaler.fit_transform(motor[cols])
motor
```

	Id	Model	Tahun	Rating	Power	Kapasitas Bahan Bakar	Jarak Sumbu Roda	Ketinggian Tempat Duduk	Bore	Stroke	...	Suspensi Belakang	Kompresi	Ban Depan	Ban Belakang	Rem Depan	Rem Belakang	Warna	Starter	Kapasitas Mesin	Kategori
	0.008942	0.010374	0.770115	0.000	0.093142	0.157895	0.440789	0.575	0.375000	0.584350	...	0.404147	0.963303	0.695772	0.790661	0.386494	0.057816	0.751013	0.0	0.055877	0.00
	0.934673	0.947815	0.425287	1.000	0.488229	0.578947	0.440789	0.000	0.375000	0.584350	...	0.404147	0.963303	0.291360	0.486381	0.625000	0.316916	0.033308	0.0	0.356241	0.00
	0.611417	0.609305	0.747126	0.000	0.273286	0.133333	0.440789	0.575	0.337963	0.248984	...	0.824421	0.917431	0.291360	0.486381	0.386494	0.057816	0.206434	0.0	0.040505	0.00
	0.842544	0.852625	0.000000	0.625	0.273286	0.578947	0.440789	0.575	0.375000	0.584350	...	0.592579	0.963303	0.291360	0.486381	0.386494	0.057816	0.033308	0.0	0.265358	0.00
	0.807424	0.810185	0.655172	0.000	0.273286	0.150877	0.000000	0.575	0.375000	0.584350	...	0.404147	0.963303	0.291360	0.486381	0.625000	0.057816	0.073708	0.0	0.042076	0.00
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61	0.705054	0.714555	0.862069	0.625	0.273286	0.403509	0.309211	0.575	0.356481	0.433943	...	0.396144	0.963303	0.053309	0.227237	0.625000	0.316916	0.718845	0.0	0.056101	0.75
62	0.935895	0.949073	0.862069	0.625	0.273286	0.407368	0.888158	0.225	0.662037	0.929878	...	0.186613	0.917431	0.085478	0.368093	1.000000	1.000000	0.322948	0.0	0.519215	0.25
63	0.721197	0.732977	0.954023	0.625	0.243603	0.543860	1.000000	0.575	0.375000	0.584350	...	0.404147	0.963303	0.656250	0.079377	0.625000	0.316916	0.177305	0.0	0.125666	0.75
64	0.217817	0.227790	0.954023	0.625	0.148414	0.203509	0.506579	1.000	0.412037	0.198171	...	0.662180	0.192661	0.053309	0.200778	1.000000	1.000000	0.967452	0.0	0.055652	0.50
65	0.320968	0.331028	0.724138	1.000	1.000000	0.578947	0.559211	0.825	0.569444	0.391260	...	0.418455	0.110092	0.291360	0.486381	0.000000	0.614561	0.613728	0.0	0.545863	0.75

5 rows × 26 columns

Melakukan pemilihan kolom yang relevan dengan performa motor

```
X = motor[['Power', 'Bore','Stroke', 'Gearbox','Kompresi','Kapasitas Mesin']]
X
```

	Power	Bore	Stroke	Gearbox	Kompresi	Kapasitas Mesin
0	0.093142	0.375000	0.584350	0.625	0.963303	0.055877
1	0.488229	0.375000	0.584350	0.375	0.963303	0.356241
2	0.273286	0.337963	0.248984	0.125	0.917431	0.040505
3	0.273286	0.375000	0.584350	0.625	0.963303	0.265358
4	0.273286	0.375000	0.584350	0.625	0.963303	0.042076
...	...	...	...	...	...	...
29961	0.273286	0.356481	0.433943	0.375	0.963303	0.056101
29962	0.273286	0.662037	0.929878	0.375	0.917431	0.519215
29963	0.243603	0.375000	0.584350	0.625	0.963303	0.125666
29964	0.148414	0.412037	0.198171	0.625	0.192661	0.055652
29965	1.000000	0.569444	0.391260	0.625	0.110092	0.545863

29966 rows × 6 columns

Melakukan pemilihan kolom yang relevan dengan performa motor

- Power

Power adalah kekuatan atau daya maksimum yang dapat dihasilkan oleh mesin, sehingga semakin tinggi power berarti performa motor semakin baik

- Bore

Bore adalah diameter silinder piston dalam mesin. Bore yang lebih besar berarti dapat meningkatkan kapasitas pembakaran, sehingga memungkinkan aliran udara yang lebih besar, yang mana akan meningkatkan performa motor

- Stroke

Stroke adalah pergerakan piston dalam satu arah di dalam silinder mesin. Stroke yang panjang dapat memberikan torsi yang lebih besar dan memiliki efisiensi bahan bakar yang lebih baik sehingga akan meningkatkan performa motor

- Gearbox

Gearbox adalah komponen yang berperan dalam pemindahan daya mesin dari satu bagian ke bagian lain. Banyaknya gear dapat mempengaruhi efisiensi bahan bakar, performa akselerasi, dan kemampuan motor dalam menangani berbagai kondisi berkendara

- Kompresi

Bore adalah banyaknya udara yang dapat dipadatkan dalam ruang bakar mesin. Rasio kompresi yang lebih tinggi dapat meningkatkan efisiensi pembakaran dan memberikan daya yang lebih tinggi sehingga dapat meningkatkan performa motor

- Kapasitas mesin

Kapasitas mesin adalah ukuran kapasitas mesin, sehingga kapasitas yang lebih tinggi akan memberikan lebih banyak daya yang mana akan meningkatkan performa motor

Melakukan pengecekan jumlah cluster untuk kemungkinan pembentukan cluster yang lebih baik

```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer
num_of_cluster = [2, 3, 4, 5, 6, 7, 8]

fig, ax = plt.subplots(4, 2, figsize=(20,10))
for k in num_of_cluster:
    # Create KMeans instance for different number of clusters
    clusterer = KMeans(n_clusters = k, n_init=10)

    # Draw silhouette diagram
    q, mod = divmod(k, 2)
    visualizer = SilhouetteVisualizer(clusterer, colors = 'yellowbrick', ax = ax[q-1][mod])
    visualizer.fit(X)

    # Compute silhoutte score
    # This gives a perspective into the density and separation of the formed clusters
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    print(
        "For n_clusters =",
        k,
        "The average silhouette_coefficient is :",
        silhouette_avg,
    )
```

```
For n_clusters = 2 The average silhouette_coefficient is : 0.3653621068268298
For n_clusters = 3 The average silhouette_coefficient is : 0.3326393996806258
For n_clusters = 4 The average silhouette_coefficient is : 0.3608844306610392
For n_clusters = 5 The average silhouette_coefficient is : 0.31600107693925367
For n_clusters = 6 The average silhouette_coefficient is : 0.32365193582190693
For n_clusters = 7 The average silhouette_coefficient is : 0.3079111359883477
For n_clusters = 8 The average silhouette_coefficient is : 0.2990728770473573
```

Menentukan cluster

```
kmeans = KMeans(n_clusters=2)
cluster_assignment = kmeans.fit_predict(X)
data_with_clusters = pd.DataFrame(X.copy(), columns=('Power', 'Bore','Stroke', 'Gearbox','Kompresi','Kapasitas Mesin'))
data_with_clusters['Clusters'] = cluster_assignment
data_with_clusters
```

	Power	Bore	Stroke	Gearbox	Kompresi	Kapasitas Mesin	Clusters
0	0.093142	0.375000	0.584350	0.625	0.963303	0.055877	0
1	0.488229	0.375000	0.584350	0.375	0.963303	0.356241	0
2	0.273286	0.337963	0.248984	0.125	0.917431	0.040505	0
3	0.273286	0.375000	0.584350	0.625	0.963303	0.265358	0
4	0.273286	0.375000	0.584350	0.625	0.963303	0.042076	0
...	...	...	...	...	...	...	...
29961	0.273286	0.356481	0.433943	0.375	0.963303	0.056101	0
29962	0.273286	0.662037	0.929878	0.375	0.917431	0.519215	0
29963	0.243603	0.375000	0.584350	0.625	0.963303	0.125666	0
29964	0.148414	0.412037	0.198171	0.625	0.192661	0.055652	1
29965	1.000000	0.569444	0.391260	0.625	0.110092	0.545863	1

29966 rows × 7 columns

Memahami karakteristik cluster

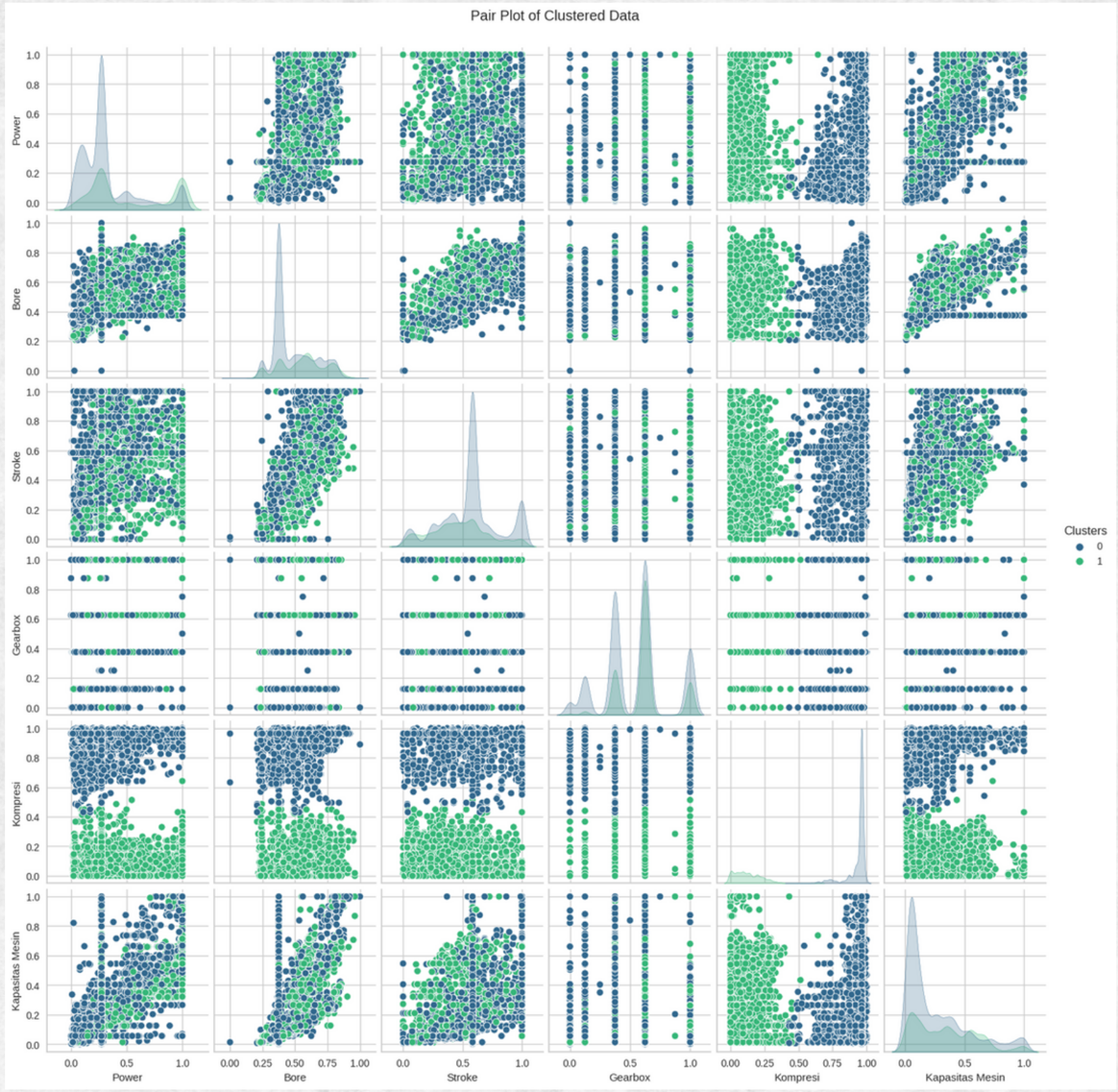
```
cluster_stats = data_with_clusters.groupby('Clusters').agg(['mean', 'median', 'std'])
cluster_stats
```

	Power			Bore			Stroke			Gearbox			Kompresi			Kapasitas Mesin		
	mean	median	std	mean	median	std	mean	median	std	mean	median	std	mean	median	std	mean	median	std
Clusters																		
0	0.323342	0.273286	0.256831	0.477392	0.375000	0.158117	0.569465	0.584350	0.243514	0.541169	0.625	0.269903	0.93083	0.963303	0.077230	0.257836	0.126171	0.267415
1	0.519616	0.273286	0.349216	0.558506	0.577778	0.174157	0.472121	0.462398	0.246638	0.618863	0.625	0.200104	0.12664	0.110092	0.098172	0.331574	0.322020	0.254822

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(data_with_clusters, hue='Clusters', diag_kind='kde', palette='viridis')
plt.suptitle('Pair Plot of Clustered Data', y=1.02)
plt.show()
```

Memahami karakteristik cluster



Hasil analisis cluster

```
cluster_stats = data_with_clusters.groupby('Clusters').agg(['mean', 'median', 'std'])
cluster_stats
```

	Power			Bore			Stroke			Gearbox			Kompresi			Kapasitas Mesin		
	mean	median	std	mean	median	std	mean	median	std	mean	median	std	mean	median	std	mean	median	std
Clusters																		
0	0.323342	0.273286	0.256831	0.477392	0.375000	0.158117	0.569465	0.584350	0.243514	0.541169	0.625	0.269903	0.93083	0.963303	0.077230	0.257836	0.126171	0.267415
1	0.519616	0.273286	0.349216	0.558506	0.577778	0.174157	0.472121	0.462398	0.246638	0.618863	0.625	0.200104	0.12664	0.110092	0.098172	0.331574	0.322020	0.254822

Berdasarkan gambar diatas, dapat dilihat bahwa:

Rata-rata Power cluster 1 > cluster 0

Rata-rata Bore cluster 1 > cluster 0

Rata-rata Stroke cluster 1 < cluster 0

Rata-rata Gearbox cluster 1 > cluster 0

Rata-rata Kompresi cluster 1 < cluster 0

Rata-rata Kapasitas mesin cluster 1 > cluster 0

Hasil analisis cluster

Cluster 0

Cluster ini memiliki power, bore, gearbox, dan kapasitas mesin yang lebih rendah, sedangkan stroke dan kompresi yang lebih tinggi, sehingga motor pada cluster ini kemungkinan memiliki performa yang lebih rendah. Cluster ini memiliki kemungkinan merupakan motor biasa yang digunakan sehari-hari yang tidak memerlukan performa tinggi, seperti motor touring, classic, dan cruiser

Cluster 1

Cluster ini memiliki power, bore, gearbox, dan kapasitas mesin yang lebih tinggi, sedangkan stroke dan kompresi yang lebih rendah, sehingga motor pada cluster ini kemungkinan memiliki performa yang lebih tinggi. Cluster ini memiliki kemungkinan merupakan motor yang memiliki kebutuhan daya yang lebih tinggi seperti motor sport atau offroad

THANKYOU