Tugas 2: Praktikum Machine Learning (Tugas/Praktikum Mandiri)

Shapiere Januar Rafiansyah - 0110224191

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: shapierejanuarr@gmail.com

1. Praktikum 2

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

```
Mounted at /content/gdrive
```

Kode from google.colab import drive dan drive.mount('/content/gdrive') dipakai buat nyambungin Colab ke Google Drive. Tujuannya biar dataset yang ada di Drive bisa langsung diakses tanpa upload manual. Outputnya biasanya minta izin akses, dan kalau berhasil Drive bakal muncul di folder /content/gdrive.

```
path = "/content/gdrive/MyDrive/praktikum/praktikum02"
```

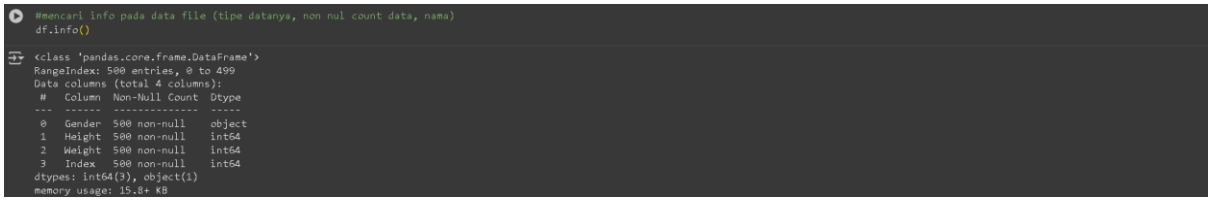
Kode path = "/content/gdrive/MyDrive/praktikum/praktikum02" dipakai buat nyimpen alamat folder tempat dataset berada di Google Drive. Tujuannya biar lebih praktis, jadi kalau mau manggil file cukup nambahin nama filenya aja tanpa nulis path panjang berulang kali.

```
# membaca file csv menggunakan pandas
import pandas as pd

df = pd.read_csv(path + '/data/500_Person_Gender_Height_Weight_Index.csv')
df
```

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

Bagian ini ngimpor library pandas lalu membaca file CSV 500_Person_Gender_Height_Weight_Index.csv yang ada di folder data. File dibaca pakai pd.read_csv() dan disimpan ke variabel df. Dengan nulis df, otomatis isi dataset ditampilkan. Dari output kelihatan ada 500 baris dan 4 kolom (Gender, Height, Weight, Index), yang berisi data jenis kelamin, tinggi badan, berat badan, dan indeks tubuh.



```
#mencari info pada data file (tipe datanya, non nul count data, nama)
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64
 2   Weight  500 non-null    int64
 3   Index   500 non-null    int64
dtypes: int64(3), object(1)
memory usage: 15.8+ KB
```

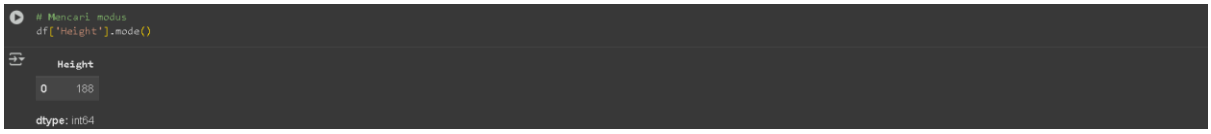
Perintah df.info() dipakai buat ngeliat informasi umum dari dataset. Dari output kelihatan ada 500 baris dan 4 kolom, semua kolom terisi penuh (nggak ada data kosong). Tipe datanya terdiri dari 3 kolom numerik (Height, Weight, Index) dan 1 kolom kategori (Gender). Info ini penting supaya kita tahu kondisi awal dataset sebelum dianalisis lebih lanjut

```
# Menghitung mean semua kolom numerik
df['Height'].mean()

np.float64(169.944)
```

Kode df['Height'].mean() dipakai buat ngitung nilai rata-rata dari kolom Height. Outputnya 169.944, artinya rata-rata tinggi badan dalam dataset ini sekitar 170 cm.

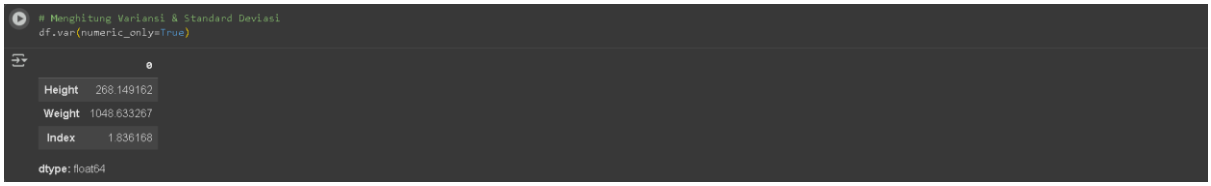
```
# Menghitung median semua kolom numerik
df['Height'].median()

170.5
```

Kode df['Height'].median() dipakai buat ngitung nilai median dari kolom *Height*. Hasilnya 170.5, yang berarti setengah dari data tinggi badan ada di bawah 170.5 cm dan setengahnya lagi di atas angka tersebut.



```
# Mencari modus
df['Height'].mode()

   Height
0     188

dtype: int64
```

Kode df['Height'].mode() dipakai buat nyari nilai modus dari kolom *Height*. Outputnya 188, artinya tinggi badan yang paling sering muncul di dataset ini adalah 188 cm. Modus ini nunjukin nilai yang paling dominan dalam data.

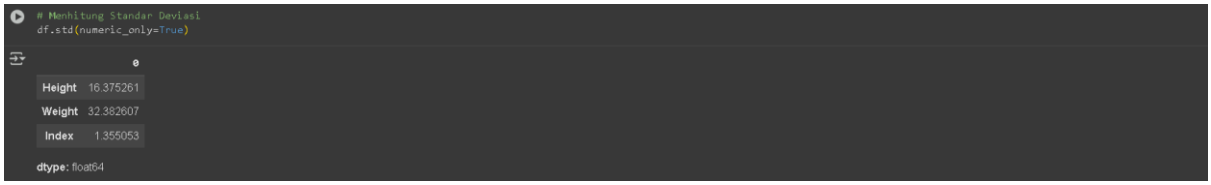


```
# Menghitung Variansi & Standard Deviasi
df.var(numeric_only=True)

                 0
Height   268.149162
Weight  1048.633267
Index      1.836168

dtype: float64
```

Kode df.var(numeric_only=True) dipakai buat ngitung variansi dari tiap kolom numerik. Output nunjukin:

- Variansi *Height* = 268.14
- Variansi *Weight* = 1048.63
- Variansi *Index* = 1.83

Variansi ini ngukur seberapa jauh data nyebar dari rata-ratanya. Nilai variansi yang lebih besar (contoh di *Weight*) berarti data lebih menyebar, sedangkan variansi kecil (contoh di *Index*) berarti datanya lebih rapat.



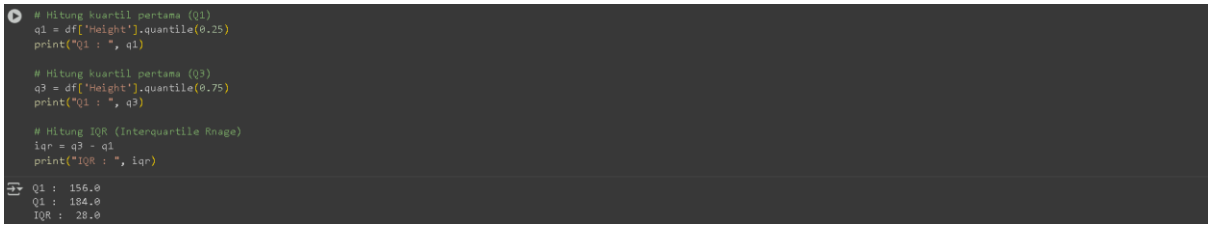
```
# Menhitung Standar Deviasi
df.std(numeric_only=True)
```

```
                0
Height  16.375261
Weight  32.382607
Index    1.355053
dtype: float64
```

ode df.std(numeric_only=True) dipakai buat ngitung standar deviasi dari tiap kolom numerik. Output nunjukin:

- *Height* = 16.37
- *Weight* = 32.38
- *Index* = 1.36

Standar deviasi ini nunjukin seberapa jauh data menyebar dari rata-ratanya dalam satuan aslinya



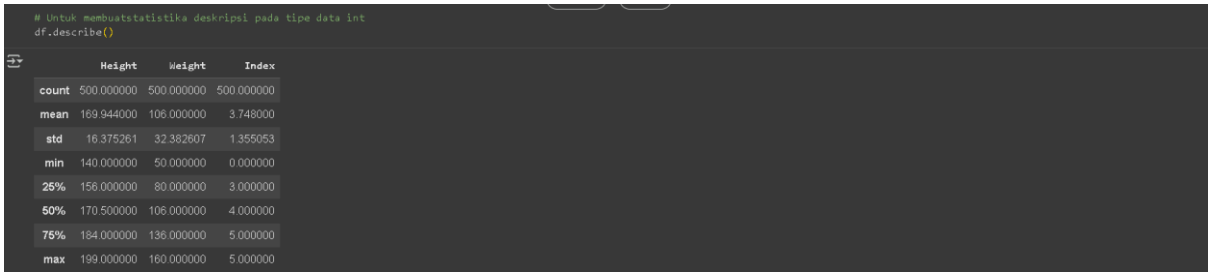
```
# Hitung kuartil pertama (Q1)
q1 = df['Height'].quantile(0.25)
print("Q1 : ", q1)

# Hitung kuartil pertama (Q3)
q3 = df['Height'].quantile(0.75)
print("Q1 : ", q3)

# Hitung IQR (Interquartile Rnage)
iqr = q3 - q1
print("IQR : ", iqr)
```

```
Q1 :  156.0
Q1 :  184.0
IQR :  28.0
```

Kode ini ngitung Q1, Q3, dan IQR pada kolom Height. Hasilnya:

- Q1 = 156
- Q3 = 184
- IQR = 28

Artinya setengah data utama tinggi badan ada di rentang 156–184 cm, dan IQR bisa dipakai buat deteksi outlier.



```
# Untuk membuatstatistika deskripsi pada tipe data int
df.describe()
```

	Height	Weight	Index
count	500.000000	500.000000	500.000000
mean	169.944000	106.000000	3.748000
std	16.375261	32.382607	1.355053
min	140.000000	50.000000	0.000000
25%	156.000000	80.000000	3.000000
50%	170.500000	106.000000	4.000000
75%	184.000000	136.000000	5.000000
max	199.000000	160.000000	5.000000

Perintah df.describe() dipakai buat bikin statistik deskriptif pada kolom numerik. Outputnya nunjukin ukuran dasar seperti:

- Mean: rata-rata tinggi 169.94 cm, berat 106 kg.
- Std: standar deviasi tinggi 16.37 cm → data lumayan nyebar.
- Min & Max: tinggi 140–199 cm, berat 50–160 kg.
- Quartile: median tinggi 170.5 cm, berat 106 kg, sesuai dengan distribusi data utama.

Ringkasnya, fungsi ini ngasih gambaran umum sebaran data tanpa harus hitung manual.

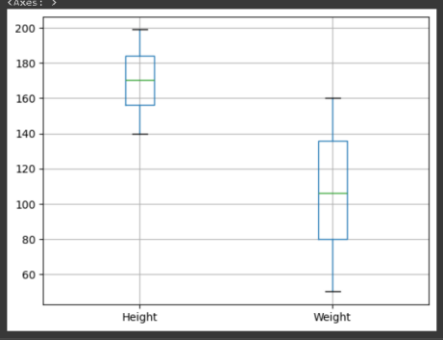
```python
# Menghitung matriks korelasi untuk semua kolom numerik
correlation_matrix = df.corr(numeric_only=True)

# Menampilkan matriks korelasi
print("Matriks Korelasi.")
print(correlation_matrix)
```

```
Matriks Korelasi.
          Height    Weight     Index
Height  1.000000  0.000446 -0.422223
Weight  0.000446  1.000000  0.804569
Index  -0.422223  0.804569  1.000000
```

Kode ini menghitung matriks korelasi antar kolom numerik. Hasilnya, Height dan Index punya korelasi negatif sedang, Weight dan Index korelasi positif kuat, sedangkan Height dan Weight hampir tidak berkorelasi.

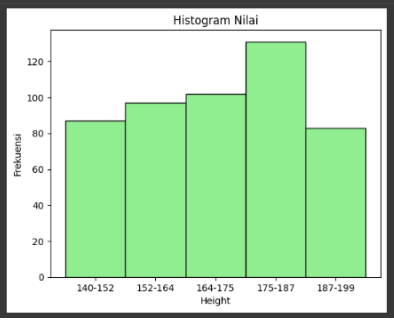
```python
import pandas as pd
import numpy as np

df.boxplot(column=['Height', 'Weight'])
```

```
<Axes: >
```



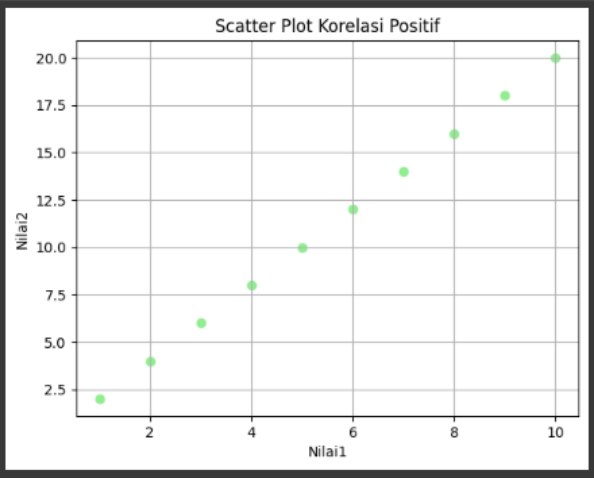
Kode ini bikin boxplot untuk kolom Height dan Weight. Dari outputnya kelihatan kalau tinggi badan (Height) mayoritas ada di sekitar 160–180 cm, sedangkan berat badan (Weight) lebih bervariasi, dari 50 sampai 160 kg. Boxplot ini juga nunjukin sebaran data dan potensi outlier.

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Ambil data Height
data_height = df["Height"]

# Buat histogram
n, bins, patches = plt.hist(data_height, bins=5, color='lightgreen', edgecolor='black')

# Tambahkan label
plt.title('Histogram Nilai')
plt.xlabel('Height')
plt.ylabel('Frekuensi')

# Tampilkan rentang frekuensi di sumbu x
bin_centers = 0.5 * (bins[:-1] + bins[1:])
plt.xticks(bin_centers, ['{:.0f}-{:.0f}'.format(bins[i], bins[i+1]) for i in range(len(bins)-1)])

# Tampilkan histogram
plt.show()
```



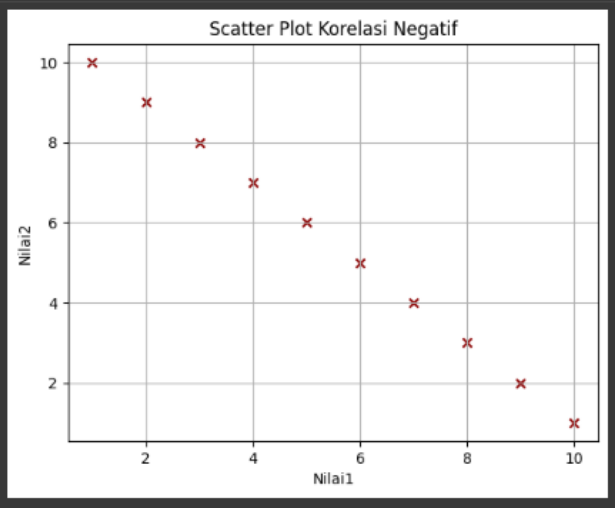
Kode ini dipakai buat bikin histogram Height dengan 5 interval. Dari outputnya kelihatan kalau tinggi badan paling banyak ada di rentang 175–187 cm dengan frekuensi tertinggi (lebih dari 120 orang). Sementara itu, rentang terendah ada di 187–199 cm dan 140–152 cm. Histogram ini ngasih gambaran distribusi data tinggi badan secara visual.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Buat DataFrame contoh
data = {
    'Nilai1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Nilai2': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
}

df2 = pd.DataFrame(data)

# Buat scatter plot
plt.scatter(df2['Nilai1'], df2['Nilai2'], color='lightgreen', marker='o')

# Tambahkan label
plt.title('Scatter Plot Korelasi Positif')
plt.xlabel('Nilai1')
plt.ylabel('Nilai2')

# Tambahkan grid
plt.grid(True)

# Tampilkan plot
plt.show()
```



Kode ini bikin scatter plot antara Nilai1 dan Nilai2. Dari grafiknya jelas kelihatan kalau dua variabel ini punya korelasi positif sempurna: setiap kenaikan Nilai1 selalu diikuti kenaikan Nilai2 secara teratur. Visualisasi ini nunjukin pola hubungan linear yang kuat, cocok buat ngegambarin data yang saling berbanding lurus.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Buat DataFrame contoh
data = {
    'Nilai1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Nilai2': [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
}

df3 = pd.DataFrame(data)

# Buat scatter plot
plt.scatter(df3['Nilai1'], df3['Nilai2'], color='darkred', marker='x')

# Tambahkan label
plt.title('Scatter Plot Korelasi Negatif')
plt.xlabel('Nilai1')
plt.ylabel('Nilai2')

# Tambahkan grid
plt.grid(True)

# Tampilkan plot
plt.show()
```



Scatter plot ini nunjukin korelasi negatif antara Nilai1 dan Nilai2. Polanya jelas terlihat, makin tinggi Nilai1, nilai Nilai2 justru turun secara teratur. Grafik ini ngasih gambaran hubungan yang saling berlawanan arah, cocok buat ngejelasin variabel yang saling berbanding terbalik.

1.1 Kesimpulan

Dari praktikum ini bisa dilihat kalau analisis statistik deskriptif cukup efektif buat ngerti karakteristik dataset. Tinggi badan punya rata-rata sekitar 170 cm dengan variasi yang lumayan besar, sementara berat badan terbukti punya korelasi kuat dengan indeks kesehatan. Sebaliknya, tinggi badan malah berkorelasi negatif dengan indeks. Visualisasi kayak boxplot, histogram, dan scatter plot juga bantu banget buat nangkep pola distribusi dan hubungan antar variabel secara lebih jelas.

2. Tugas Praktikum Mandiri

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

```
Mounted at /content/gdrive
```

Kode from google.colab import drive dan drive.mount('/content/gdrive') dipakai buat nyambungin Colab ke Google Drive. Tujuannya biar dataset yang ada di Drive bisa langsung diakses tanpa upload manual. Outputnya biasanya minta izin akses, dan kalau berhasil Drive bakal muncul di folder /content/gdrive.

```
path = "/content/gdrive/MyDrive/praktikum/praktikum02"
```

Kode path = "/content/gdrive/MyDrive/praktikum/praktikum02" dipakai buat nyimpen alamat folder tempat dataset berada di Google Drive. Tujuannya biar lebih praktis, jadi kalau mau manggil file cukup nambahin nama filenya aja tanpa nulis path panjang berulang kali.

```
# membaca file csv menggunakan pandas
import pandas as pd

df = pd.read_csv(path + '/data/day.csv')
df
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
726	727	2012-12-27	1	1	12	0	4	1	2	0.254167	0.226642	0.652917	0.350133	247	1867	2114
727	728	2012-12-28	1	1	12	0	5	1	2	0.253333	0.255046	0.590000	0.155471	644	2451	3095
728	729	2012-12-29	1	1	12	0	6	0	2	0.253333	0.242400	0.752917	0.124383	159	1182	1341
729	730	2012-12-30	1	1	12	0	0	0	1	0.255833	0.231700	0.483333	0.350754	364	1432	1796
730	731	2012-12-31	1	1	12	0	1	1	2	0.215833	0.223487	0.577500	0.154846	439	2290	2729

731 rows × 16 columns

Kode ini dipakai buat membaca dataset day.csv yang ada di folder data menggunakan library pandas. Output yang muncul adalah tampilan tabel lengkap berisi 731 baris dan 16 kolom. Setiap baris merepresentasikan data harian penyewaan sepeda dengan atribut seperti tanggal (dteday), musim (season), suhu (temp), kelembapan (hum), hingga jumlah penyewa (casual, registered, dan cnt).

Intinya, langkah ini jadi pondasi awal sebelum masuk ke analisis lebih lanjut karena kita perlu pastiin dataset berhasil dimuat dan strukturnya terbaca dengan benar.

```python
# bagi data
# 80% training, 20% testing
train_size = int(0.8 * len(df))
train = df[:train_size]
test = df[train_size:]

# 10% validation dari training
val_size = int(0.1 * len(train))
val = train[:val_size]
train = train[val_size:]
```

Kode ini dipakai buat membagi dataset jadi tiga bagian:

- Training (80%): data utama untuk membangun model.
- Testing (20%): data untuk menguji performa model.
- Validation (10% dari training): data tambahan buat ngecek dan menyesuaikan model sebelum diuji di testing.

Output dari kode ini adalah tiga subset data (train, val, test) yang siap dipakai sesuai fungsinya.

```python
# jumlah data
print("Jumlah data training:", len(train))
print("Jumlah data validation:", len(val))
print("Jumlah data testing:", len(test))
```

```
Jumlah data training: 526
Jumlah data validation: 58
Jumlah data testing: 147
```

Dari hasil pembagian, dataset terdistribusi jadi 526 data untuk training, 58 data untuk validation, dan 147 data untuk testing. Jumlah ini sudah sesuai dengan proporsi yang ditentukan (80% : 10% : 20%), jadi pembagiannya bisa dianggap valid untuk keperluan analisis maupun pembuatan model.

```python
# 5 data pertama training
print("Training data:")
print(train.head())

# 5 data pertama validation
print("\nValidation data:")
print(val.head())

# 5 data pertama testing
print("\nTesting data:")
print(test.head())
```

```
    weathersit      temp     atemp       hum  windspeed  casual  registered  \
58           2  0.407273  0.400118  0.876364   0.289686      81        1365
59           1  0.266667  0.263879  0.535000   0.216425     137        1714
60           1  0.335000  0.320071  0.449583   0.307833     231        1903
61           1  0.198333  0.200133  0.318333   0.225754     123        1562
62           2  0.261667  0.255679  0.610417   0.203346     214        1730

     cnt
58  1446
59  1851
60  2134
61  1685
62  1944

Validation data:
   instant      dteday  season  yr  mnth  holiday  weekday  workingday  \
0        1  2011-01-01       1   0     1        0        6           0
1        2  2011-01-02       1   0     1        0        0           0
2        3  2011-01-03       1   0     1        0        1           1
3        4  2011-01-04       1   0     1        0        2           1
4        5  2011-01-05       1   0     1        0        3           1

   weathersit      temp     atemp       hum  windspeed  casual  registered  \
0           2  0.344167  0.363625  0.805833   0.160446     331         654
1           2  0.363478  0.353739  0.696087   0.248539     131         670
2           1  0.196364  0.189405  0.437273   0.248309     120        1229
3           1  0.200000  0.212122  0.590435   0.160296     108        1454
4           1  0.226957  0.229270  0.436957   0.186900      82        1518

    cnt
0   985
1   801
2  1349
3  1562
4  1600

Testing data:
     instant      dteday  season  yr  mnth  holiday  weekday  workingday  \
584      585  2012-08-07       3   1     8        0        2           1
585      586  2012-08-08       3   1     8        0        3           1
586      587  2012-08-09       3   1     8        0        4           1
587      588  2012-08-10       3   1     8        0        5           1
588      589  2012-08-11       3   1     8        0        6           0

     weathersit      temp     atemp       hum  windspeed  casual  registered  \
584           2  0.735833  0.697621  0.703750   0.116908    1278        5995
585           2  0.750000  0.707717  0.672917   0.110700    1263        6271
586           1  0.755833  0.699508  0.620417   0.156100    1196        6090
587           2  0.715833  0.667942  0.715833   0.238813    1065        4721
588           2  0.692500  0.638267  0.732917   0.206479    2247        4052

      cnt
584  7273
585  7534
586  7286
587  5786
588  6299
```

Dari output yang ditampilkan, terlihat bahwa pembagian data berhasil dilakukan sesuai proporsi. Training set menampilkan 5 baris pertama setelah validasi, berisi data mulai akhir Februari 2011. Validation set diambil dari awal dataset (Januari 2011) sebanyak 10% dari training, sedangkan Testing set diambil dari bagian akhir dataset (Agustus 2012).

Hal ini nunjukin kalau dataset udah terpisah dengan jelas: training dipakai buat belajar pola, validation buat evaluasi model sementara, dan testing buat menguji performa di data baru.

2.1 Kesimpulan

Pada tugas mandiri ini, dataset berhasil dibagi menjadi tiga bagian: training, validation, dan testing sesuai proporsi yang ditentukan. Hasilnya nunjukin bahwa data terbagi merata dan tetap konsisten dari segi format maupun isi. Pembagian ini penting karena memungkinkan model machine learning dilatih, dievaluasi, dan diuji secara terpisah, sehingga performanya bisa lebih objektif dan tidak bias terhadap data tertentu.

3. GitHub

https://github.com/Shapiere/Praktikum-MachineLearning