

Pertemuan VII

Selamat datang pada MK Praktikum Komputasi Big Data pertemuan ke-7. Melanjutkan kegiatan praktikum sebelumnya, pada pertemuan ini Anda akan melanjutkan tahapan pengolahan data yaitu data training dan model generation.

Nama : Agrieva Xananda Pramuditha

Kelas : 2IA18

NPM: 50423070

Import Dataset

```
# Import library
import pandas as pd
import numpy as np

# Import dataset
data = pd.read_csv('https://gitlab.com/andreass.bayu/file-directory/-/raw/main/adult.csv')

# Lihat 5 data teratas
data.head(5)
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States

Langkah berikutnya: [Buat kode dengan data](#) [Lihat plot yang direkomendasikan](#) [New interactive sheet](#)

Review Dataset

```
# Lihat data deskripsi dari tiap kolom dengan menggunakan fungsi describe
data.describe()
```

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

```
# Lihat tipe data dari tiap kolom dengan menggunakan fungsi dtypes
data.dtypes
```



0

<b>age</b>	int64
<b>workclass</b>	object
<b>fnlwgt</b>	int64
<b>education</b>	object
<b>educational-num</b>	int64
<b>marital-status</b>	object
<b>occupation</b>	object
<b>relationship</b>	object
<b>race</b>	object
<b>gender</b>	object
<b>capital-gain</b>	int64
<b>capital-loss</b>	int64
<b>hours-per-week</b>	int64
<b>native-country</b>	object
<b>income</b>	object
<b>dtype:</b>	object

```
# Melihat jumlah atribut dan dimensi data (baris dan kolom) dengan menggunakan fungsi shape
data.shape
```



```
(48842, 15)
```

```
# Menghitung dan melihat jumlah data per label kelas
for col in data.columns:
    if data[col].dtype == "object":
        print('Attribute name:',col)
        print('-----')
        print(data[col].value_counts())
        print('-----')
```



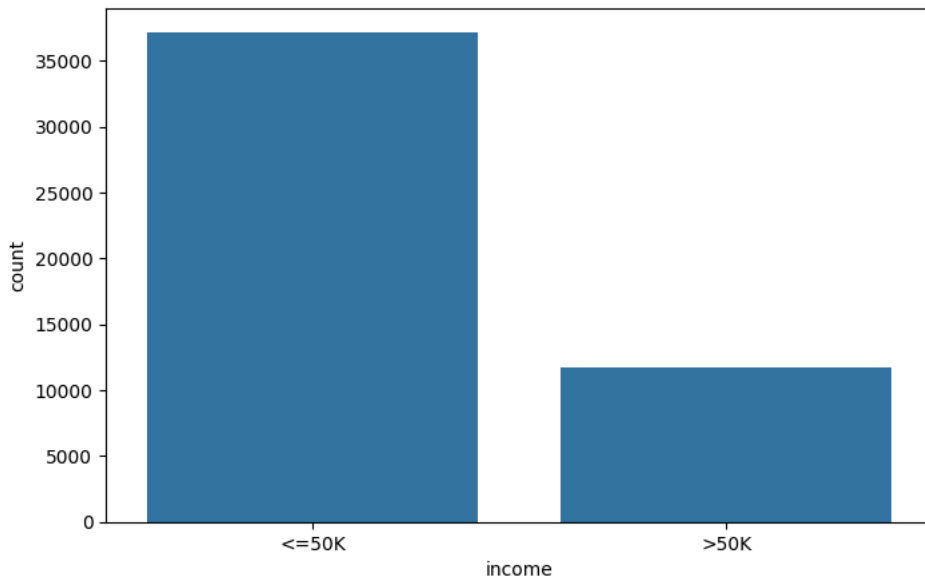
```
Name: count, dtype: int64
-----
Attribute name: income
-----
income
<=50K    37155
>50K     11687
Name: count, dtype: int64
-----
```

```
# Import library seaborn untuk visualisasi
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Plot figure untuk menentukan distribusi kelas
plt.figure(figsize=(8,5))

# Menghitung baris setiap kelas
sns.countplot(x="income", data=data)
```

```
<Axes: xlabel='income', ylabel='count'>
```



Diketahui bahwa dataset memiliki distribusi kelas yang tidak seimbang (imbalanced) sehingga secara teknis akan digunakan teknik untuk menangani data yang tidak seimbang.

## Dataset preparation

```
# Buat salinan dataframe
df = data.copy(deep = True)

# Mengubah nilai "?" nilai ke bentuk Na / NaN untuk diproses lebih lanjut
for col in data.columns:
    df[[col]] = data[[col]].replace('?', np.NaN)
```

```
# Melakukan seleksi kolom fitur/feature columns dari dataset
null_data = df.iloc[:, :-1]
```

```
# Temukan nilai null untuk semua atribut dan jumlahkan total nilai null
null_data.isnull().sum()
```



	0
age	0
workclass	2799
fnlwtg	0
education	0
educational-num	0
marital-status	0
occupation	2809
relationship	0
race	0
gender	0
capital-gain	0
capital-loss	0
hours-per-week	0
native-country	857

dtype: int64

```
# Hapus/drop semua baris yang memiliki nilai null
df = df.dropna()

# Pilih kolom fitur/feature columns dari dataset
null_data = df.iloc[:, :-1]

# Cek ulang nilai null
null_data.isnull().sum()
```



	0
age	0
workclass	0
fnlwtg	0
education	0
educational-num	0
marital-status	0
occupation	0
relationship	0
race	0
gender	0
capital-gain	0
capital-loss	0
hours-per-week	0
native-country	0

dtype: int64

**StandardScaler** adalah class dari sklearn untuk melakukan normalisasi data agar data yang digunakan tidak memiliki penyimpangan yang besar.

```
# Import library standard scaler
from sklearn.preprocessing import StandardScaler

# Buat dataframe dengan tipe data int64
colname= []
for col in df.columns:
    if df[col].dtype == "int64":
        colname.append(col)

# Buat salinan dataset untuk keperluan persiapan data (data preparation)
df_copy = df.copy(deep = True)
df_fe = df_copy()

# Buat dataframe untuk fitur kategoris (categorical features)
df_fe.drop('income',axis='columns', inplace=True)
df_fe.drop(colname,axis='columns', inplace=True)

# Buat dataframe untuk kelas target (target class)
```

```
df_cl = df.copy()
df_cl.drop(df_copy.iloc[:, :-1], axis='columns', inplace=True)

# Membuat objek scaler (scaler object)
std_scaler = StandardScaler()
std_scaler

# Normalisasikan atribut numerik dan tetapkan ke dalam dataframe baru
df_norm = pd.DataFrame(std_scaler.fit_transform(df_copy[colname]), columns=colname)

# Import library Ordinal Encoder dari package library sklearn.preprocessing
from sklearn.preprocessing import OrdinalEncoder
ord_enc = OrdinalEncoder()

# Encode fitur kategoris/categorical features menjadi fitur numerik (numerical features)
for col in df_fe.columns[:]:
    if df_fe[col].dtype == "object":
        df_fe[col] = ord_enc.fit_transform(df_fe[[col]])

# Encode label kategorikal/categorical label menjadi label biner (binary label)
df_cl["income"] = np.where(df_cl["income"].str.contains("> 50K"), 0, 1)
```

```
# Masukkan kolom id ke dataset yang berbeda
df_norm.insert(0, 'id', range(0, 0 + len(df_norm)))
df_fe.insert(0, 'id', range(0, 0 + len(df_fe)))
df_cl.insert(0, 'id', range(0, 0 + len(df_cl)))

# Lihat shapes dataset yang telah di proses
print(df_norm.shape)
print(df_fe.shape)
print(df_cl.shape)
```

```
(45222, 7)
(45222, 9)
(45222, 2)
```

```
# Gabungkan semua dataset
df_feature = pd.merge(df_norm, df_fe, on=["id"])
df_final = pd.merge(df_feature, df_cl, on=["id"])

# Drop kolom id dari gabungan dataset
df_final.drop('id', axis='columns', inplace=True)

# Lihat 5 data teratas dari gabungan dataset
df_final.head(5)
```



	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-per-week	workclass	education	marital-status	occupation	relationship	race	gender
0	-1.024983	0.350889	-1.221559	-0.146733	-0.21878	-0.078120	2.0	1.0	4.0	6.0	3.0	2.0	1.0
1	-0.041455	-0.945878	-0.438122	-0.146733	-0.21878	0.754701	2.0	11.0	2.0	4.0	0.0	4.0	1.0
2	-0.798015	1.393592	0.737034	-0.146733	-0.21878	-0.078120	1.0	7.0	2.0	10.0	0.0	4.0	1.0
3	0.412481	-0.278420	-0.046403	0.877467	-0.21878	-0.078120	2.0	15.0	2.0	6.0	0.0	2.0	1.0
4	-0.344079	0.084802	-1.613277	-0.146733	-0.21878	-0.910942	2.0	0.0	4.0	7.0	1.0	4.0	1.0

Langkah berikutnya:

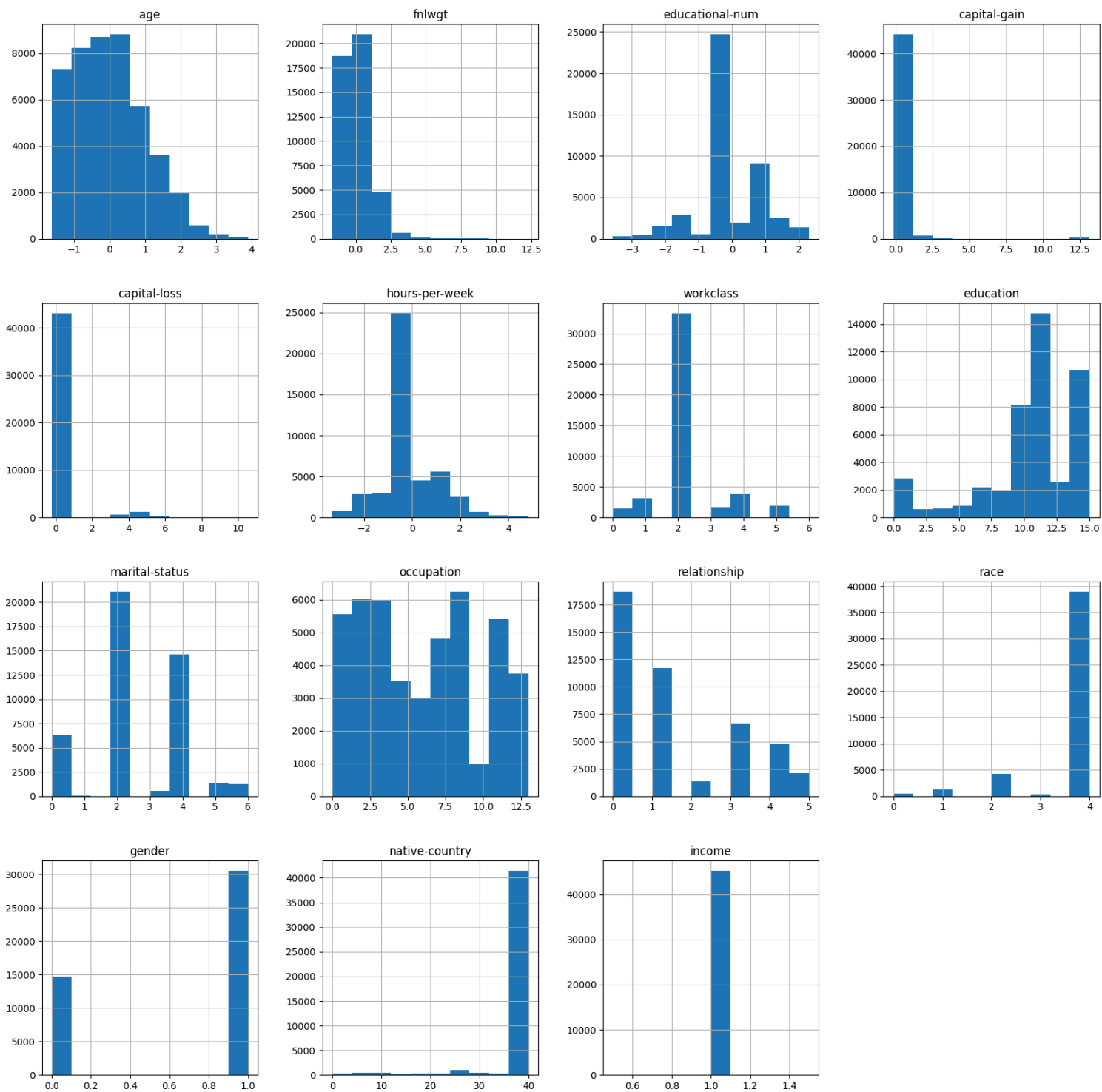
[Buat kode dengan df\\_final](#)

[Lihat plot yang direkomendasikan](#)

[New interactive sheet](#)

## Visualization

```
p = df_final.hist(figsize = (20,20))
```




.....

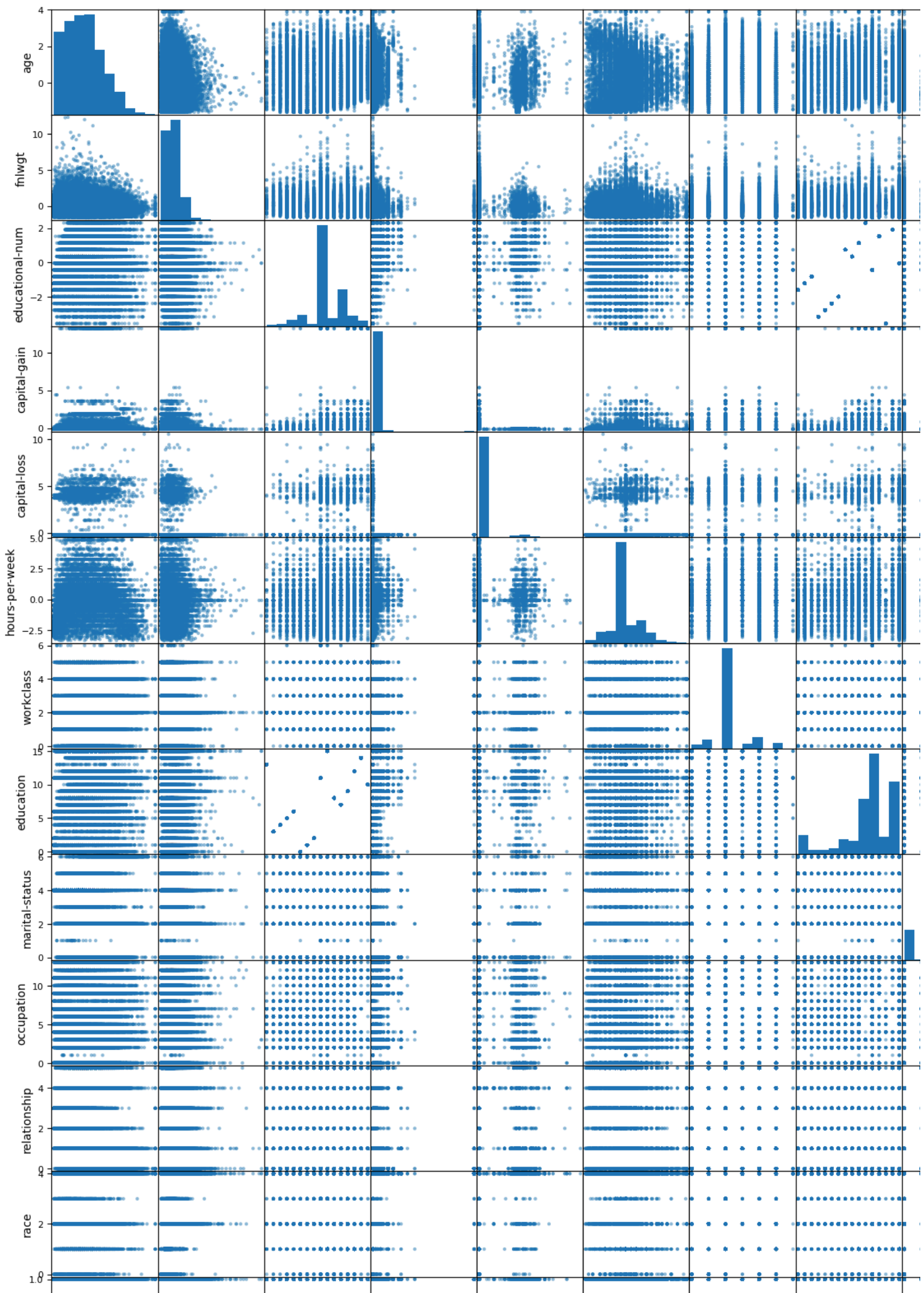
Gambar ini menampilkan histogram untuk masing-masing variabel dalam dataset. Histogram ini memperlihatkan distribusi frekuensi data pada s  
Misalnya, distribusi usia (age) menunjukkan mayoritas data berada di usia muda hingga paruh baya. Variabel seperti capital-gain dan capita  
terkonsentrasi di bagian rendah, menunjukkan banyak individu yang tidak memiliki keuntungan atau kerugian modal yang signifikan. Histogram  
distribusi jam kerja dengan puncak di kisaran tengah. Beberapa variabel kategorikal seperti workclass, marital-status, dan occupation juga  
yang lebih dominan tampak lebih tinggi. Histogram ini berguna untuk memahami pola distribusi awal dari setiap variabel, mendeteksi ketidak

yang tebin dominan tampak tebin tinggi. Histogram ini berguna untuk memahami pola distribusi awal dari setiap variabel, mendeteksi ketiadaan kecenderungan atau nilai yang paling umum.

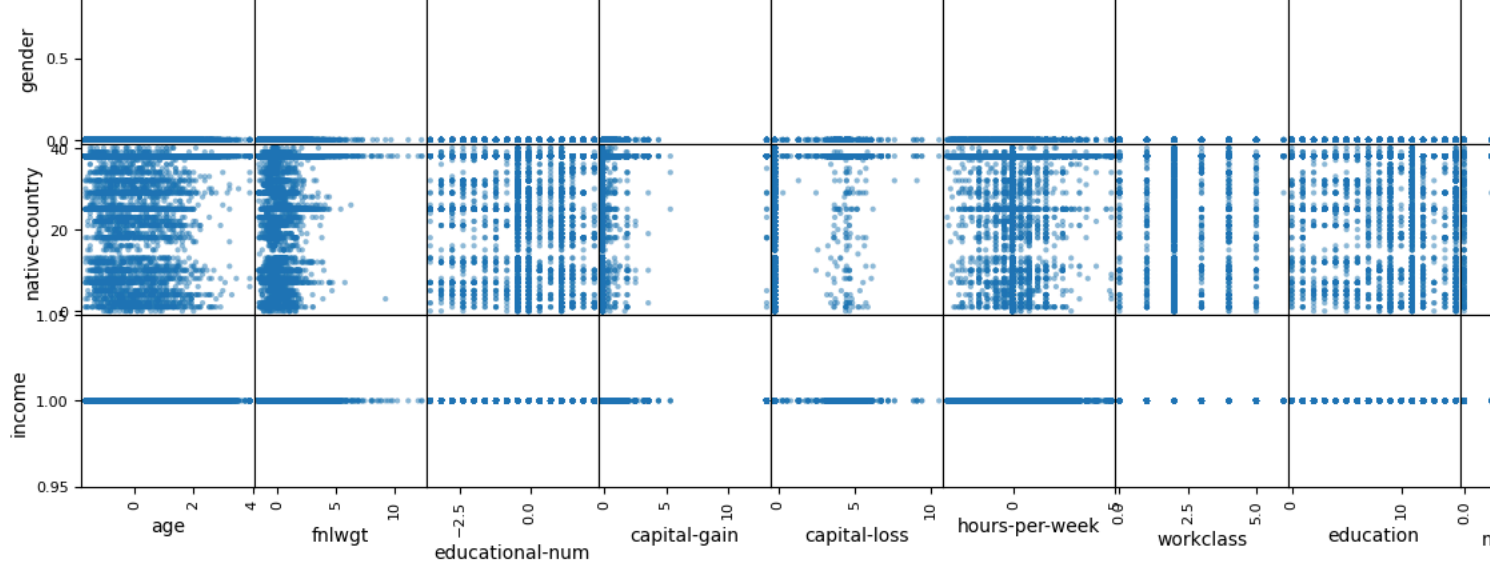
**Scatter matrix plot adalah** plot yang digunakan untuk membuat sekumpulan scatter plot dari beberapa pasang variabel. Hal ini sangat bermanfaat terutama ketika ingin menganalisis bagaimana bentuk hubungan antar variabel. Plot ini sangat bermanfaat untuk digunakan untuk data yang ukurannya tidak terlalu besar. Untuk menggunakan scatter matrix kita harus memanggil fungsi *scatter\_matrix* dari *pandas.plotting*

```
from pandas.plotting import scatter_matrix  
  
p=scatter_matrix(df_final,figsize=(25, 25))
```

 /usr/local/lib/python3.10/dist-packages/pandas/plotting/\_matplotlib/misc.py:100: UserWarning: Attempting to set identical low and high  
ax.set\_xlim(boundaries\_list[j])  
/usr/local/lib/python3.10/dist-packages/pandas/plotting/\_matplotlib/misc.py:101: UserWarning: Attempting to set identical low and high  
ax.set\_ylim(boundaries\_list[i])  
/usr/local/lib/python3.10/dist-packages/pandas/plotting/\_matplotlib/misc.py:91: UserWarning: Attempting to set identical low and high  
ax.set\_xlim(boundaries\_list[i])







```

"""
Gambar ini merupakan matriks scatter plot yang menampilkan hubungan sepasang variabel dalam dataset. Setiap sel pada matriks ini menunjuk
variabel yang berbeda melalui scatter plot. Diagonal matriks memperlihatkan histogram dari masing-masing variabel, memberikan gambaran di
Scatter plot di luar diagonal membantu mengidentifikasi pola hubungan atau korelasi antara dua variabel. Misalnya, pola titik yang memben
menunjukkan korelasi linear antara dua variabel, sedangkan pola acak menunjukkan hubungan yang lemah atau tidak ada hubungan sama sekali.
awal data karena membantu mengenali hubungan antar fitur dan mengidentifikasi variabel yang mungkin berhubungan atau berdampak satu sama
"""

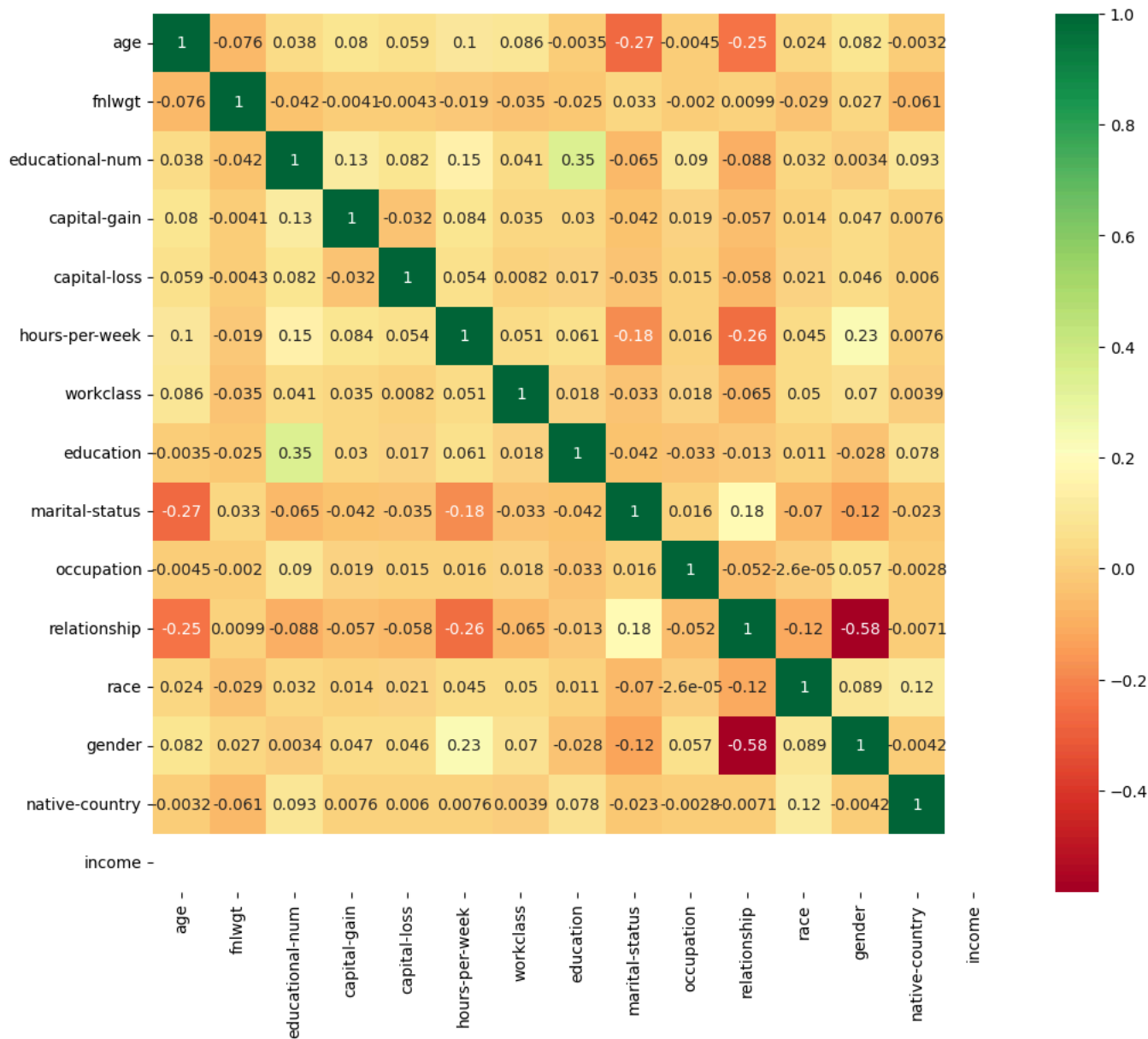
```

```

# Buat visualisasi korelasi data dengan heatmap
import seaborn as sns
import matplotlib.pyplot as plt

# plot heatmap
plt.figure(figsize=(12,10))
p=sns.heatmap(df_final.corr(), annot=True,cmap ='RdYlGn')

```



"""

Heatmap ini menunjukkan hubungan korelasi antar variabel dalam dataset, di mana warna hijau menunjukkan korelasi positif, warna merah menandakan korelasi negatif, dan warna kuning menunjukkan korelasi yang lemah atau tidak signifikan. Beberapa korelasi yang menonjol termasuk hubungan positif antara yang mengindikasikan bahwa tingkat pendidikan berhubungan dengan jumlah tahun pendidikan yang dijalani. Selain itu, terdapat korelasi negatif antara gender (-0.58), yang menunjukkan adanya pola hubungan tertentu antara jenis kelamin dan status hubungan. Sebagian besar variabel lain menunjukkan hubungan linear yang rendah. Korelasi dengan variabel target income juga terlihat lemah.

"""

## Proses Modelling dengan KNN

```
import numpy as np
from sklearn.model_selection import StratifiedKFold

# Corrected target extraction
X = df_final.iloc[:, :-1].to_numpy()
y = df_final.iloc[:, -1].to_numpy()

# Check unique values and class distribution in y
unique, counts = np.unique(y, return_counts=True)
print("Unique values in y:", unique)
print("Counts of each class in y:", counts)

# Define Stratified KFold to handle class imbalance
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Apply Stratified KFold cross-validation
for train_index, test_index in skf.split(X, y):
    print("TRAIN:", train_index, "TEST:", test_index)
    x_train, x_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```



```
Unique values in y: [1]
Counts of each class in y: [45222]
TRAIN: [ 0 1 2 ... 45218 45219 45221] TEST: [ 7 13 34 ... 45209 45216 45220]
```

```

TRAIN: [ 0 1 2 ... 45218 45219 45220] TEST: [ 4 18 20 ... 45215 45217 45221]
TRAIN: [ 2 3 4 ... 45219 45220 45221] TEST: [ 0 1 10 ... 45197 45202 45213]
TRAIN: [ 0 1 2 ... 45219 45220 45221] TEST: [ 11 12 16 ... 45191 45199 45201]
TRAIN: [ 0 1 4 ... 45217 45220 45221] TEST: [ 2 3 5 ... 45214 45218 45219]

```

```

print('----- x axis test -----')
print(x_test)
print('----- x axis train -----')
print(x_train)
print('----- y axis test -----')
print(y_test)
print('----- y axis train -----')
print(y_train)

print('*****')

```

```

----- x axis test -----
[[-0.79801494  1.39359159  0.73703421 ...  4.          1.
 38.          ]
 [ 0.4124809  -0.27841995 -0.046403   ...  2.          1.
 38.          ]
 [ 1.84994471 -0.8056638   1.91219002 ...  4.          1.
 38.          ]
 ...
 [-0.49539098 -0.69668789  1.52047141 ...  1.          1.
 35.          ]
 [ 0.10985694 -0.3347349  -0.43812161 ...  4.          1.
 38.          ]
 [ 1.47166476 -0.35805983 -0.43812161 ...  4.          0.
 38.          ]]
----- x axis train -----
[[-1.02498291  0.35088942 -1.22155881 ...  2.          1.
 38.          ]
 [-0.04145504 -0.94587846 -0.43812161 ...  4.          1.
 38.          ]
 [-0.344079   0.08480152 -1.61327742 ...  4.          1.
 38.          ]
 ...
 [-0.87367093  0.6396112   0.73703421 ...  4.          0.
 38.          ]
 [-1.25195088  0.11127873 -0.43812161 ...  4.          1.
 38.          ]
 [ 1.01772882  0.92951628 -0.43812161 ...  4.          0.
 38.          ]]
----- y axis test -----
[1 1 1 ... 1 1 1]
----- y axis train -----
[1 1 1 ... 1 1 1]
*****

```

```

"""
Kode ini menggunakan Stratified K-Fold cross-validation untuk membagi dataset menjadi lima bagian (fold) dengan mempertahankan proporsi k
Namun, hasil yang ditampilkan menunjukkan bahwa y hanya memiliki satu kelas (1) dengan total 45,222 data, yang berarti dataset ini tidak
Akibatnya, setiap fold dari y_train dan y_test hanya berisi nilai 1, sehingga tidak memungkinkan model untuk belajar atau membedakan anta
Ketiadaan variasi kelas ini membuat algoritma klasifikasi seperti KNN menjadi tidak berguna karena memerlukan minimal dua kelas untuk mem
mendapatkan hasil yang bermakna, perlu dipastikan bahwa dataset memiliki beberapa kelas atau mempertimbangkan pendekatan lain, seperti de
satu kelas ini tetap digunakan.
"""

```

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

```