



Nama : Irfan Kurniawan Suthiono

Kelas : IF-7 NoBP : 22101152630330

PORTFOLIO MACHINE LEARNING

Kasus :

**Prediksi Pendapatan Tahunan Seseorang
Melebihi \$50k/Tahun Berdasarkan Data
Sensus**

UNIT 1 : MENGUMPULKAN DATA

1.1 SUMBER DATA

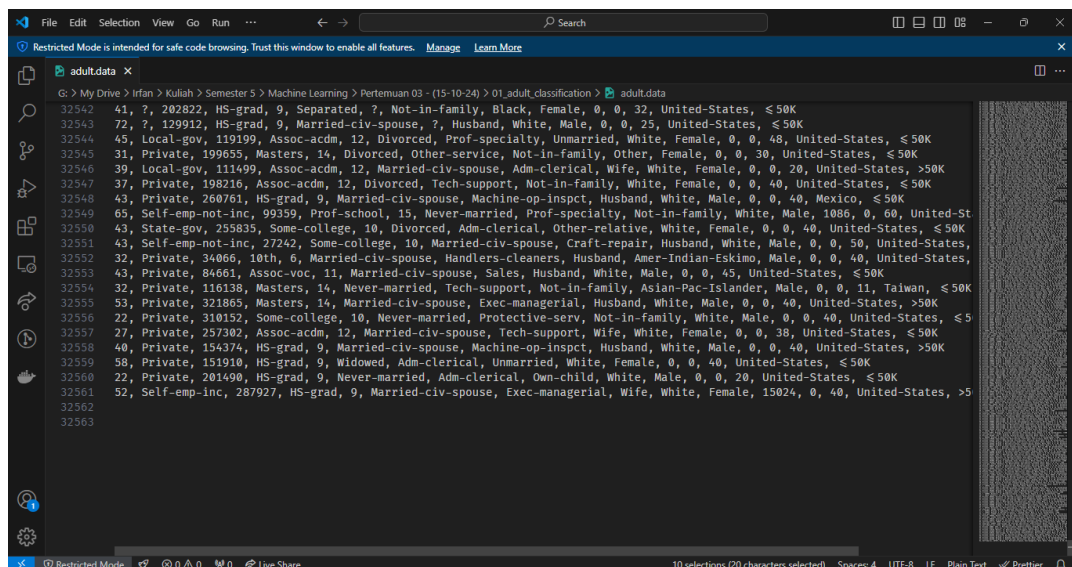
Data yang digunakan dalam proyek ini diambil dari

<https://archive.ics.uci.edu/dataset/2/adult>

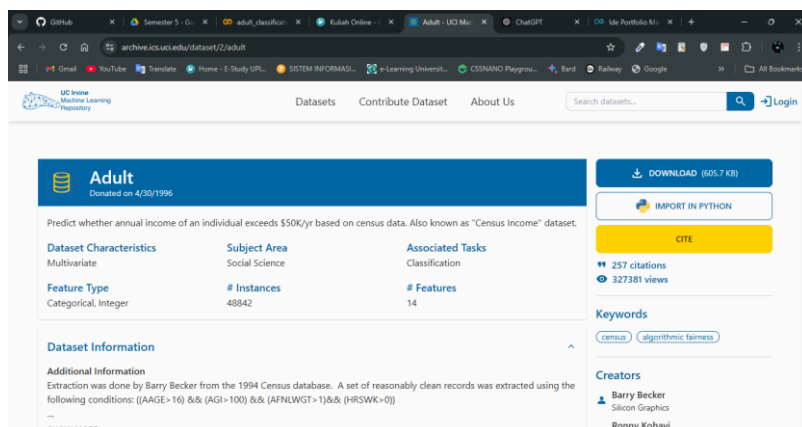
1.2 DESKRIPSI DATASET

Sekumpulan data yang diekstrak dengan kondisi ((AAGE>16) & (AGI>100) && (AFNLWGT>1)&& (HRSWK>0)), karakteristik dari dataset yang digunakan adalah Multivariate (categorical, Integer, Biner).

Jumlah data yang terdapat di dalam dataset : 32.561 data tetapi dalam website tertera 48842



1.3 IDENTIFIKASI SUMBER DATA



Adult
Donated on 4/30/1996

Predict whether annual income of an individual exceeds \$50K/yr based on census data. Also known as "Census Income" dataset.

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Social Science	Classification
Feature Type	# Instances	# Features
Categorical, Integer	48842	14

Dataset Information

Additional Information
Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Keywords
census, algorithmic fairness

Creators
Barry Becker
Silicon Graphics
Ronny Kohavi

Download Options
DOWNLOAD (605.7 KB)
IMPORT IN PYTHON
CITE

Statistics
257 citations
327381 views

Data ini diambil dari repository yang tersedia di **UCI Machine Learning** yang di ekstrak oleh **Barry Becker** dari **database Sensus 1994**.

1.4 STRUKTUR DATA PADA DATASET

1. Jumlah Instance :
 - a. Yang tertera pada website : 48.842
 - b. Yang sebenarnya pada dataset : 32.561
2. Jumlah Atribut : 14
3. Target Variabel : Yang menjadi target variabel pada proyek ini adalah variabel **Income** dimana akan memprediksi pendapatan seseorang melebihi \$50k/tahun

1.5 DAFTAR ATRIBUT DAN TIPE DATANYA :

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
age	Feature	Integer	Age	N/A		no
workclass	Feature	Categorical	Income	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.		yes
fnlwgt	Feature	Integer				no
education	Feature	Categorical	Education Level	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.		no
education-num	Feature	Integer	Education Level			no
marital-status	Feature	Categorical	Other	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.		no
occupation	Feature	Categorical	Other	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.		yes
relationship	Feature	Categorical	Other	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.		no
race	Feature	Categorical	Race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.		no
sex	Feature	Binary	Sex	Female, Male.		no
capital-gain	Feature	Integer				no
capital-loss	Feature	Integer				no
hours-per-week	Feature	Integer				no

native-country	Feature	Categorical	Other	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad&Tobago, Peru, Hong, Holand-Netherlands.	yes
income	Target	Binary	Income	> 50K, <= 50K.	no

Penjelasan :

a. Age

Memiliki role **Feature** dan tipe data dari age adalah **Integer** dan data ini tidak memiliki *missing values*

b. Workclass

Memiliki role **Feature** dan tipe data dari workclass adalah **Categorical** yang kemungkinan berisi value : Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. Dan variabel ini memiliki *missing values*

c. Fnlwgt

Memiliki role Feature dan tipe datanya adalah Integer, data ini tidak memiliki *missing values*

d. Education

Memiliki role **Feature** dan tipe datanya adalah **Categorical**, yang kemungkinan berisi : Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. Data ini tidak memiliki *missing values*

e. Education-num

Memiliki role **Feature** dan tipe datanya adalah **Integer**, data ini tidak memiliki *missing values*

f. Marital-status

Memiliki role **Feature** dan tipe datanya adalah **Categorical**, yang kemungkinan berisi : Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. Data ini tidak memiliki *missing values*

g. Occupation

Memiliki role **Feature** dengan tipe datanya **Categorical**, yang kemungkinan berisi : Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces. Data ini memiliki *missing values*

h. Relationship

Memiliki role **Feature** dan tipe datanya **Categorical**, yang kemungkinan berisi : Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. Data ini tidak memiliki *missing values*

i. Race

Memiliki role **Feature** dan tipe datanya **Categorical**, yang kemungkinan berisi : White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. Data ini tidak memiliki *missing values*

j. Sex

Memiliki role **Feature** dan tipe datanya **Binary**, yang dimana Male dan Female. Data ini tidak memiliki *missing values*

k. Capital-gain

Memiliki role **Feature** dan tipe datanya adalah **Integer**, data ini tidak memiliki *missing values*

l. Capital-loss

Memiliki role **Feature** dan tipe datanya adalah **Integer**, data ini tidak memiliki *missing values*

m. Hours-per-week

Memiliki role **Feature** dan tipe datanya adalah **Integer**, data ini tidak memiliki *missing values*

n. Native-country

Memiliki role **Feature** dan tipe datanya adalah **Categorical** yang kemungkinan berisi : United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua,

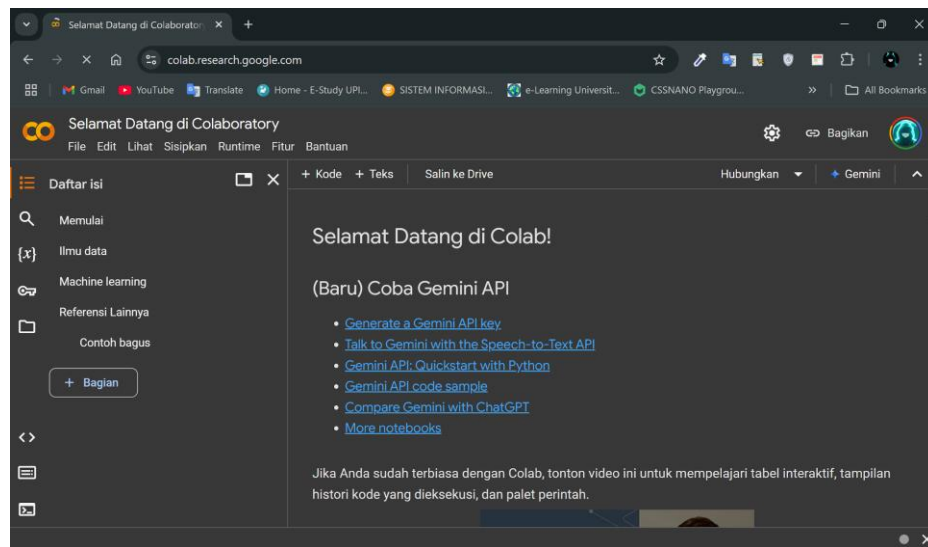
Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad&Tobago, Peru, Hong, Holand-Netherlands. Data ini memiliki *missing values*

o. Income

Variabel ini merupakan variabel yang menjadi target untuk proyek kali ini, memiliki tipe data **Binary**, berisi >50k dan <=50k

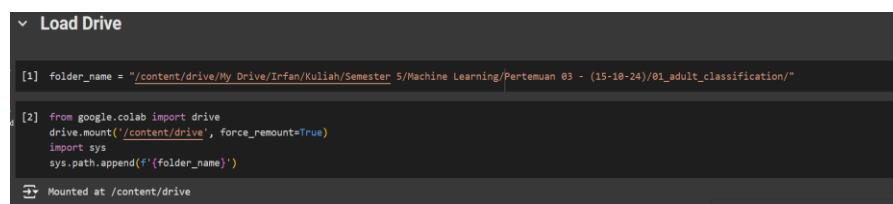
UNIT 2 : MENELAAH DATA

2.1 Platform yang digunakan



Menggunakan google colaboratory sebagai sarana untuk pemrosesan data

2.2 Membaca Lokasi Data Pada Drive



Pada tahap awal ini, karena menggunakan google colaboratory, maka perlu membaca data dari drive terlebih dahulu. Dengan cara mengimport drive dari google.colab kemudian mount lokasi file (variabel folder_name) yang ada pada drive.

2.3 Pembacaan Dataset

2.3.1 Import Library Yang Dibutuhkan

```
Load Library

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.set_option('display.max_columns', None)
```

Sebelum pengolahan data, harus import seluruh library yang dibutuhkan seperti pandas untuk mempermudah pembacaan dan pengolahan data, numpy untuk melakukan perhitungan yang rumit berbentuk array atau objek, pyplot dari library matplotlib berfungsi sebagai visualisasi data berbentuk grafik dan seaborn untuk membentuk visualisasi bar plot.

2.3.2 Membaca dan Memberikan Nama Kolom Pada Dataset

```
Read Data

column_names=['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']
df = pd.read_csv(f'{folder_name}/adult.data', sep=',', names=column_names, skipinitialspace=True, na_values="?")
df.head()
```

Karena dataset ini tidak memiliki nama kolom, jadi sebelum menampilkan data, sebaiknya memberikan nama kolom agar visualisasi data menjadi lebih baik, setelah dibuatkan variabel array `column_names`, maka selanjutnya akan membaca dataset dengan atribut separator sebagai “,” dan atribut `names` berisikan `column_names`, dan `na_values` (simbol ? pertanda dari dataset bahwa data dengan value tersebut adalah missing values).

Tampilan Data :

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Menampilkan 5 baris pertama pada dataset

2.3.3 Informasi Dataset

2.3.3.1 Missing Values Pada Dataset

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  -
 0   age                 32561 non-null  int64
 1   workclass           30725 non-null  object
 2   fnlwgt              32561 non-null  int64
 3   education           32561 non-null  object
 4   education-num       32561 non-null  int64
 5   marital-status      32561 non-null  object
 6   occupation          30718 non-null  object
 7   relationship        32561 non-null  object
 8   race                32561 non-null  object
 9   sex                 32561 non-null  object
10   capital-gain        32561 non-null  int64
11   capital-loss        32561 non-null  int64
12   hours-per-week      32561 non-null  int64
13   native-country      31978 non-null  object
14   income              32561 non-null  object
dtypes: int64(6), object(9)

missing_values = df.isnull().sum()
print("jumlah missing values : ")
print(missing_values)

jumlah missing values :
age                0
workclass          1836
fnlwgt              0
education           0
education-num       0
marital-status      0
occupation          1843
relationship         0
race                0
sex                 0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country      583
income              0
dtype: int64

missing_values_percentage = (missing_values / len(df)) * 100
print("Missing values percentage: ")
print(missing_values_percentage)

Missing values percentage:
age                0.000000
workclass          5.638647
fnlwgt              0.000000
education           0.000000
education-num       0.000000
marital-status      0.000000
occupation          5.660146
relationship         0.000000
race                0.000000
sex                 0.000000
capital-gain        0.000000
capital-loss        0.000000
hours-per-week      0.000000
native-country      1.790486
income              0.000000
dtype: float64
```

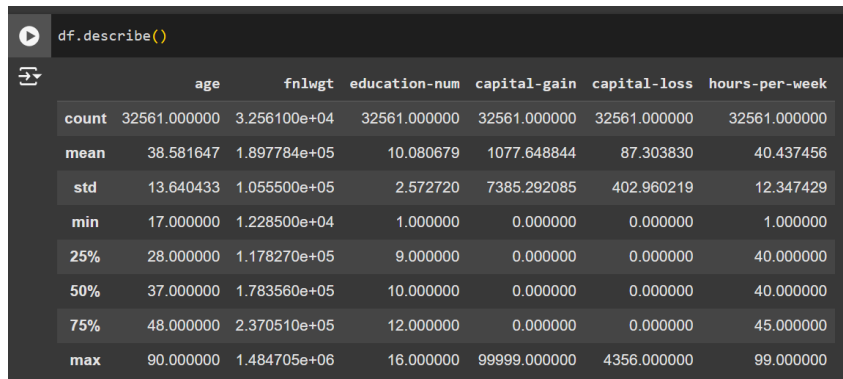
- Age tidak memiliki baris yang kosong karena berisi **32.561** dari **32.561** baris, memiliki tipe data **integer**, total *missing values*nya: **0 baris** dan presentase *missing values*nya: **0%**
- Workclass memiliki baris yang kosong karena hanya berisi **30.725 baris** dari **32.561 baris**, memiliki tipe data **object**, total *missing values*nya: **1.836 baris** dan presentase *missing values*nya **5,64%**
- Fnlwgt tidak memiliki baris yang kosong karena berisi **32.561 baris** dari **32.561 baris**, memiliki tipe data **integer**, total *missing values*nya: **0 baris** dan presentase *missing values*nya: **0%**
- Education tidak memiliki baris yang kosong karena berisi **32.561 baris** dari **32.561 baris**, memiliki tipe data **object**, total *missing values*nya: **0 baris** dan presentase *missing values*nya: **0%**

- Education-num tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **integer**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Marital-status tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **object**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Occupation memiliki baris yang kosong karena hanya berisi **30.718 baris dari 32.561 baris**, total *missing valuenya*: **1.843 baris**, dan presentase *missing valuesnya*: **5,66%**
- Relationship tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **object**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Race tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **object**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Sex tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **object**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Capital-gain tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **integer**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Capital-loss tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **integer**, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**
- Hours-per-week tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data

integer, total *missing valuesnya*: **0 baris** dan presentase *missing valuesnya*: **0%**

- Native-country memiliki baris yang kosong karena hanya berisi **31.978 baris dari 32.561 baris**, memiliki tipe data **object** total *missing valuesnya*: **583 baris** dan presentase *missing valuesnya*: **1,79%**
- Income tidak memiliki baris yang kosong karena berisi **32.561 baris dari 32.561 baris**, memiliki tipe data **integer**, total *missing valuesnya*: **0**, dan presentase *missing valuesnya*: **0%**

2.3.3.2 Analisa Statistik Deskriptif



```
df.describe()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Penjelasan :

- Age :
 - Total Data : 32561.000000
 - Rata-rata : 38.581647
 - Standar Deviasi : 13.640433
 - Minimal : 17.000000
 - Q1 : 28.000000
 - Q2 : 37.000000
 - Q3 : 48.000000
 - Maksimal : 90.000000
- Fnlwgt:
 - Total Data : 3.256100e+04
 - Rata-rata : 1.897784e+05
 - Standar Deviasi : 1.055500e+05
 - Minimal : 1.228500e+04

- Q1 : 1.178270e+05
- Q2 : 1.783560e+05
- Q3 : 2.370510e+05
- Maksimal : 1.484705e+06

- Education-num:

- Total Data : 32561.000000
- Rata-rata : 10.080679
- Standar Deviasi : 2.572720
- Minimal : 1.000000
- Q1 : 9.000000
- Q2 : 10.000000
- Q3 : 12.000000
- Maksimal : 16.000000

- Capital-gain:

- Total Data : 32561.000000
- Rata-rata : 1077.648844
- Standar Deviasi : 7385.292085
- Minimal : 0.000000
- Q1 : 0.000000
- Q2 : 0.000000
- Q3 : 0.000000
- Maksimal : 99999.000000

- Capital-loss:

- Total Data : 32561.000000
- Rata-rata : 87.303830
- Standar Deviasi : 402.960219
- Minimal : 0.000000
- Q1 : 0.000000
- Q2 : 0.000000
- Q3 : 0.000000
- Maksimal : 4356.000000

- Hours-per-week:
 - Total Data : 32561.000000
 - Rata-rata : 40.437456
 - Standar Deviasi : 12.347429
 - Minimal : 1.000000
 - Q1 : 40.000000
 - Q2 : 40.000000
 - Q3 : 45.000000
 - Maksimal : 99.000000

2.3.3.3 Analisa Statistik Kategorikal

```
df.describe(include=['object'])
```

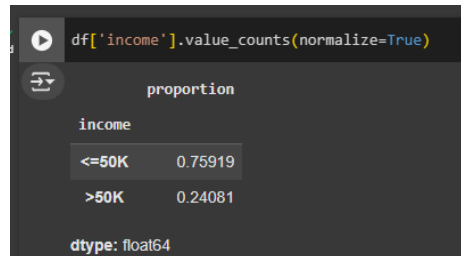
	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
count	30725	32561	32561	30718	32561	32561	32561	31978	32561
unique	8	16	7	14	6	5	2	41	2
top	Private	HS-grad	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States	<=50K
freq	22696	10501	14976	4140	13193	27816	21790	29170	24720

Penjelasan :

- Workclass :
 - Total Data : 30.725
 - Varian Data : 8
 - Data yang sering muncul : Private
 - Banyak data yang sering muncul : 22.696
- Education :
 - Total Data : 32.561
 - Varian Data : 16
 - Data yang sering muncul : HS-grad
 - Banyak data yang sering muncul : 10.501
- Marital-status :
 - Total Data : 30.725
 - Varian Data : 7
 - Data yang sering muncul : Married-civ-spouse
 - Banyak data yang sering muncul : 14.976

- Occupation :
 - Total Data : 30.718
 - Varian Data : 14
 - Data yang sering muncul : Prof-specialty
 - Banyak data yang sering muncul : 4.140
- Relationship :
 - Total Data : 32.561
 - Varian Data : 6
 - Data yang sering muncul : Husband
 - Banyak data yang sering muncul : 13.193
- Race :
 - Total Data : 32.561
 - Varian Data : 5
 - Data yang sering muncul : White
 - Banyak data yang sering muncul : 13.193
- Sex :
 - Total Data : 32.561
 - Varian Data : 2
 - Data yang sering muncul : Male
 - Banyak data yang sering muncul : 21.790
- Native-country :
 - Total Data : 31.978
 - Varian Data : 41
 - Data yang sering muncul : United-States
 - Banyak data yang sering muncul : 29.170
- Income :
 - Total Data : 32.561
 - Varian Data : 2
 - Data yang sering muncul : <=50K
 - Banyak data yang sering muncul : 24.720

2.3.3.4 Distribusi Target Variabel



```
df['income'].value_counts(normalize=True)
```

income	proportion
<=50K	0.75919
>50K	0.24081

dtype: float64

Penjelasan :

75.919% dari total data memiliki income $\leq 50k$

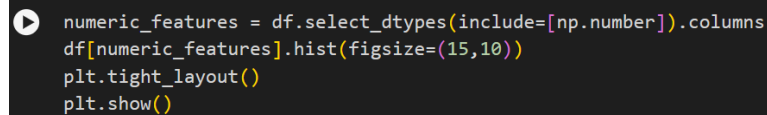
24.081% dari total data memiliki income $> 50k$

terjadinya class imbalance (ketidakseimbangan kelas) karena berbanding 3:1

2.4 Visualisasi Grafik

2.4.1 Variabel Numerik Menggunakan Histogram

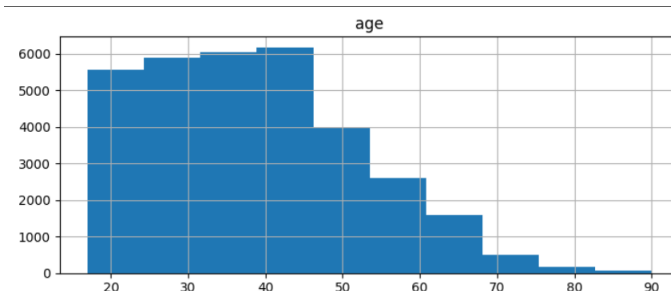
Menampilkan histogram dengan cara dibawah ini :



```
numeric_features = df.select_dtypes(include=[np.number]).columns
df[numeric_features].hist(figsize=(15,10))
plt.tight_layout()
plt.show()
```

Membuat variabel `numeric_features` untuk filter kolom data dari seluruh tipe data sehingga hanya mengambil kolom data dengan type number numpy, kemudian membuat grafik histogram berdasarkan kolom data yang difilter, setelah itu `plt.tight_layout()` berfungsi untuk merapikan grafik histogram dan mengatur jarak antar histogram, terakhir `plt.show()` untuk menampilkan grafik histogramnya.

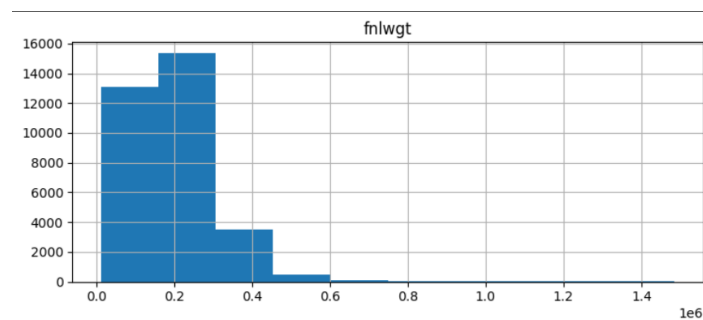
2.4.1.1 Visualisasi Variabel Age Dengan Grafik Histogram



Penjelasan :

Pada grafik diatas, terlihat jelas bahwa dataset ini memiliki data paling banyak jumlah datanya adalah yang berumur 38-48 dan yang paling sedikit jumlah datanya adalah yang berumur 84-90.

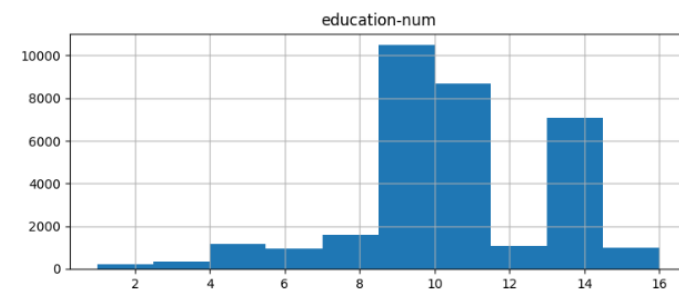
2.4.1.2 Visualisasi Variabel Fnlwgt Menggunakan Grafik Histogram



Penjelasan :

Pada grafik diatas, terlihat bahwa dataset pada kolom fnlwgt terbanyaknya ada diantara 0,1 - 0,3. Dan yang paling sedikit dari keseluruhan data adalah 0,6 - 0,8.

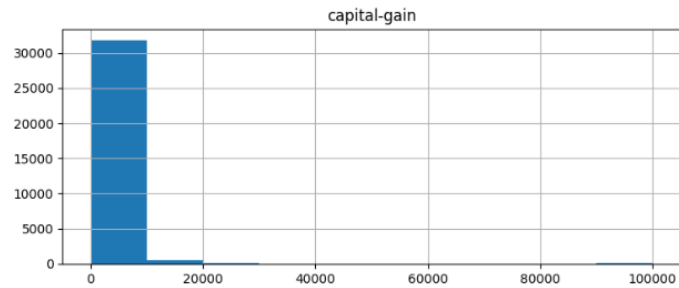
2.4.1.3 Visualisasi Education-num Menggunakan Grafik Histogram



Penjelasan :

Menurut grafik diatas, dapat diberikan kesimpulan yaitu level edukasi 9 merupakan yang terbanyak, dan level edukasi 1 yang sedikit.

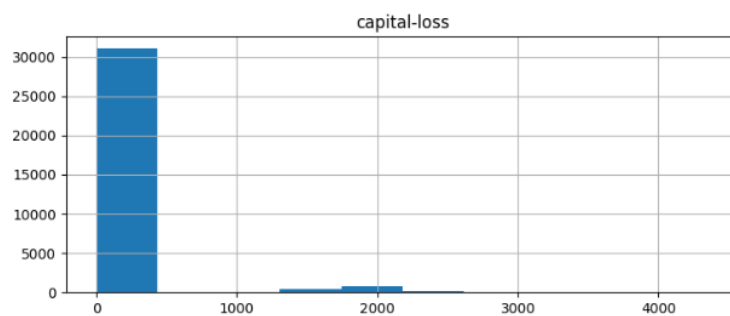
2.4.1.4 Visualisasi Capital-gain Menggunakan Grafik Histogram



Penjelasan :

Pada grafik histogram capital-gain, dapat dilihat bahwa keuntungan yang didapat terbanyak berkisar di antara \$0 - \$10K, yang terkecil berkisar diantara \$10K - \$20K.

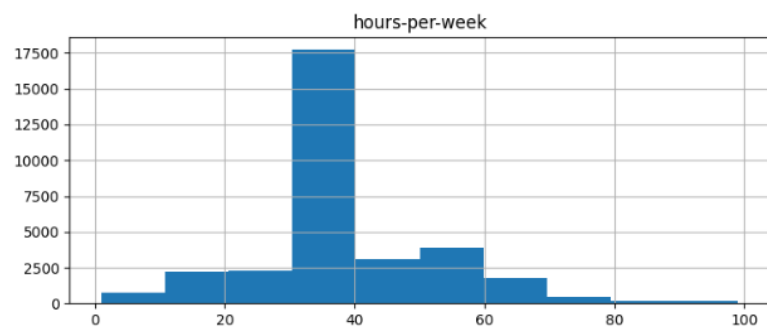
2.4.1.5 Visualisasi Capital-loss Menggunakan Grafik Histogram



Penjelasan :

Menurut grafik histogram variabel capital loss, terlihat bahwa kerugian terbanyak berkisaran antara \$0 - \$500 dan kerugian terkecil berkisar diantara \$1.5K - \$2.5K.

2.4.1.6 Visualisasi Hours-per-week Menggunakan Grafik Histogram



Penjelasan :

Berdasarkan grafik yang tertera bahwa hours-per-week terbanyak berada diantara 30 hingga 40 jam, dan jam kerja terkecil per minggu diantara 80 hingga 100 jam.

2.4.2 Variabel Kategorikal Menggunakan Grafik Barplot

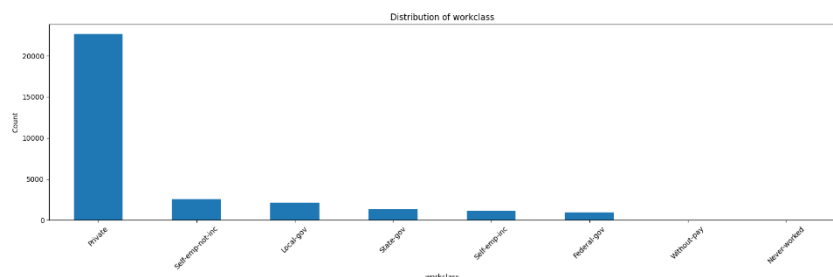
Menampilkan grafik barplot dengan cara dibawah ini :

```
categorical_features=df.select_dtypes(include=['object']).columns
for feature in categorical_features:
    plt.figure(figsize=(20,5))
    df[feature].value_counts().plot(kind='bar')
    plt.title(f'Distribution of {feature}')
    plt.ylabel('Count')
    plt.xlabel(feature)
    plt.xticks(rotation=45)
    plt.show()
```

Penjelasan :

Membuat variabel categorical_features untuk filter semua kolom hanay bertipe object, kemudian melakukan perulangan pada categorical_features untuk mencetak barplot yang berukuran panjang 20 dan lebar 5, kemudian menghitung valuenya dan membentuk grafik barnya. Diberi judulnya adalah distribusi untuk setiap feature yang di looping, dan memberikan variabel feature sebagai x labelnya tetapi karena kemungkinan besar untuk penamaan x label terlalu panjang makanya akan dirotasi 45°, dan plt.show() untuk menampilkan barplotnya.

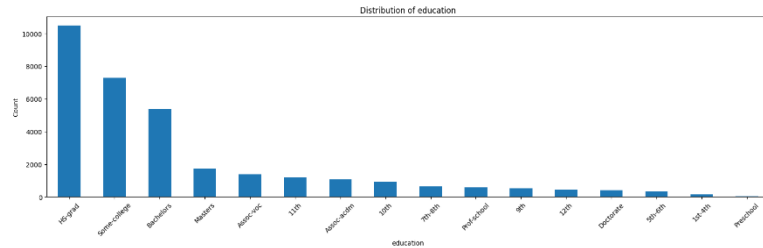
2.4.2.1 Visualisasi Grafik Barplot Untuk Variabel Workclass



Penjelasan :

Pada grafik diatas, dapat dilihat bahwa kategori private adalah yang terbanyak pada data, dan yang paling sedikit adalah federal-gov.

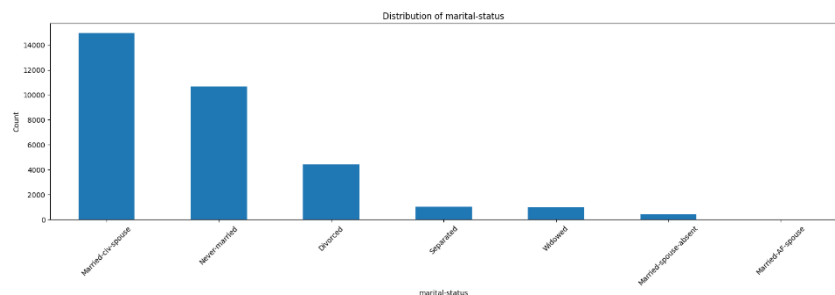
2.4.2.2 Visualisasi Grafik Barplot Untuk Variabel Education



Penjelasan :

Pada grafik diatas, dapat disimpulkan bahwa edukasi yang terbanyak adalah HS-grad dan yang paling sedikit adalah 1st – 4th.

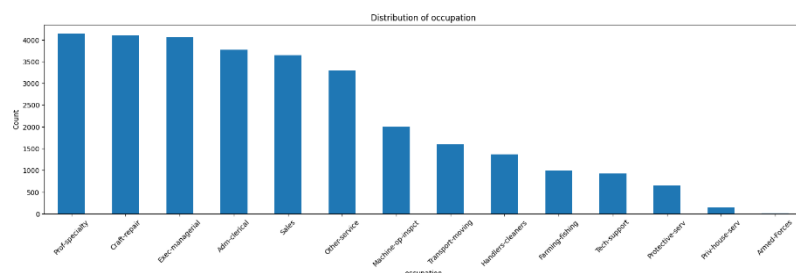
2.4.2.3 Visualisasi Grafik Barplot Untuk Variabel Marital-status



Penjelasan :

Dilihat dari grafik diatas, yang terbanyak adalah Married-civ-spouse, dan yang paling sedikit status perkawinannya adalah Married-spouse-absent.

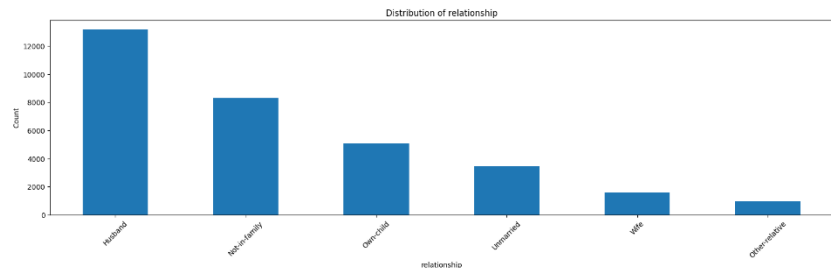
2.4.2.4 Visualisasi Grafik Barplot Untuk Variabel Occupation



Penjelasan:

Menurut grafik diatas, occupation terbanyak dari dataset adalah Prof-specialty dan yang paling sedikit dari dataset adalah occupationArmed-Forces.

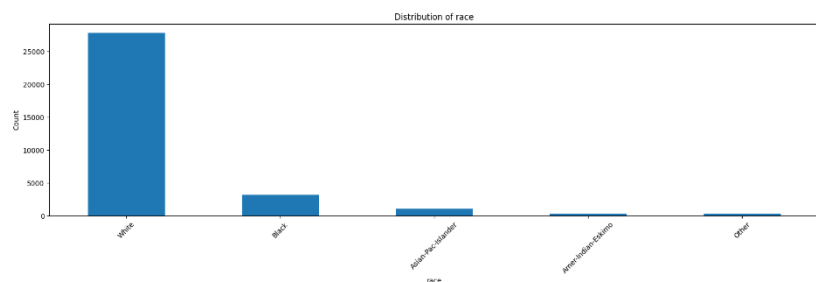
2.4.2.5 Visualisasi Grafik Barplot Untuk Variabel Relationship



Penjelasan :

Pada gambar grafik diatas tertera bahwa relationship terbanyak dari dataset adalah Husband dan yang paling sedikit adalah Other-relative.

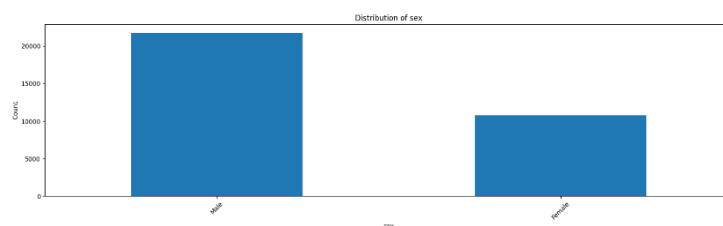
2.4.2.6 Visualisasi Grafik Barplot Untuk Variabel Race



Penjelasan :

Pada gambar grafik diatas, Race terbanyak dari dataset adalah White, dan yang paling sedikit adalah Other.

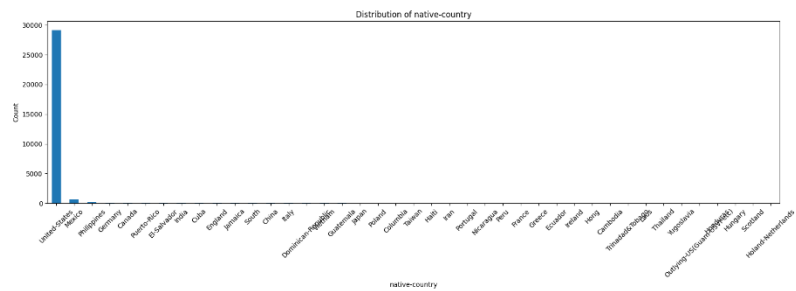
2.4.2.7 Visualisasi Grafik Barplot Untuk Variabel Sex



Penjelasan :

Grafik barplot diatas menunjukkan bahwa sex terbanyak dari dataset adalah Male daripada Female.

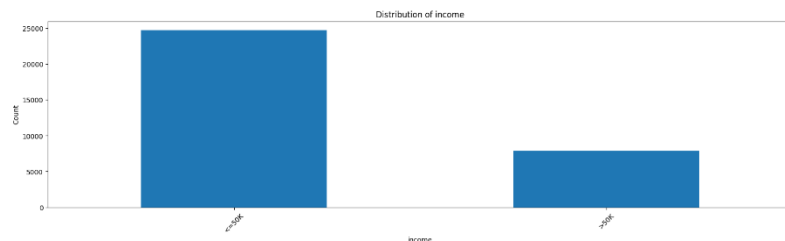
2.4.2.8 Visualisasi Grafik Barplot Untuk Variabel Native-country



Penjelasan :

Grafik barplot diatas menunjukkan bahwa dari dataset tersebut data terbanyak adalah dari negara United-States, dan yang paling sedikit adalah dari Holand-Netherlands

2.4.2.9 Visualisasi Grafik Barplot Untuk Variabel Income

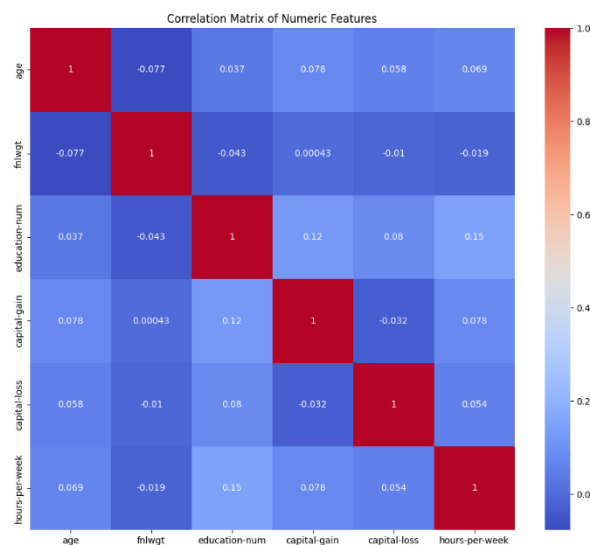


Penjelasan :

Banyaknya data dari dataset yang berpenghasilan \leq \$50K daripada yang memiliki penghasilan $>$ \$50K.

2.5 Analisa Korelasi

2.5.1 Analisa Korelasi Variabel Bertipe Numerik



Penjelasan :

Terlihat pada tabel korelasi tidak ada data yang saling berhubungan erat yang akan mempengaruhi nilai variabel lainnya, sehingga tiap hal memberikan informasi yang berbeda tanpa saling tumpang tindih terlalu banyak.

2.5.2 Analisa Korelasi Variabel Numerik Dengan Variabel Target

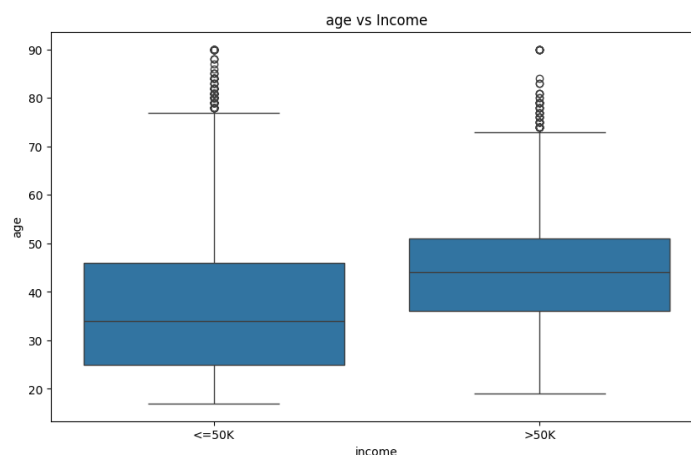
Untuk memunculkan korelasi dengan variabel target seperti dibawah ini

```
for feature in numeric_features:  
    plt.figure(figsize=(10,6))  
    sns.boxplot(x='income', y=feature, data=df)  
    plt.title(f'{feature} vs Income')  
    plt.show()
```

Penjelasan :

Disini variabel numeric_features akan dilooping dan variabel sementara akan disimpan pada variabel feature, kemudian di dalam looping yang akan menampilkan boxplot dimana x label adalah income dan y adalah variabel feature pada looping numeric_features, dan datanya ada pada variabel df. Kemudian berikan title setiap grafik boxplot, dan tampilkan grafiknya.

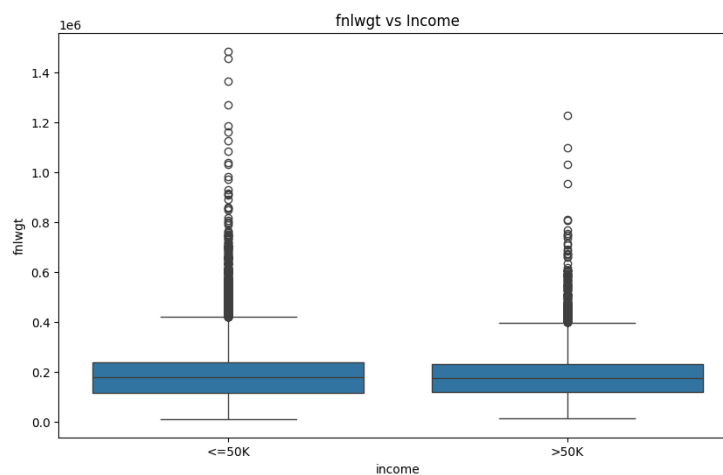
2.5.2.1 Analisa Hubungan Age Dengan Income



Penjelasan :

Diagram box plot ini menunjukkan perbedaan usia berdasarkan pendapatan. Median usia kelompok dengan pendapatan lebih dari 50K lebih tinggi daripada kelompok dengan pendapatan 50K atau kurang. Kotak biru di setiap kelompok menunjukkan rentang usia utama (50% tengah), di mana kelompok berpendapatan lebih tinggi memiliki usia yang lebih terkonsentrasi. Titik-titik di atas garis menunjukkan outlier, yaitu individu yang usianya jauh lebih tinggi daripada sebagian besar data. Diagram ini menggambarkan bahwa rata-rata usia orang dengan pendapatan lebih tinggi cenderung lebih tua dibandingkan kelompok dengan pendapatan lebih rendah.

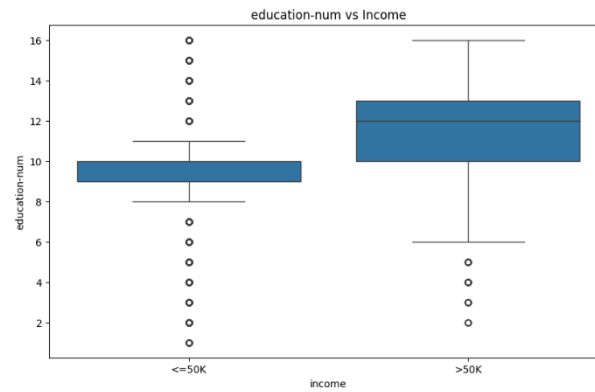
2.5.2.2 Analisa Hubungan Fnlwgt Dengan Income



Penjelasan :

Diagram box plot ini menunjukkan bahwa distribusi nilai fnlwgt (final weight) pada kedua kelompok pendapatan ($\leq 50K$ dan $>50K$) relatif serupa, dengan median fnlwgt di sekitar 200,000 untuk keduanya, rentang utama yang mirip, serta banyak pencilan (outliers) di bagian atas yang menunjukkan adanya individu dengan fnlwgt sangat tinggi.

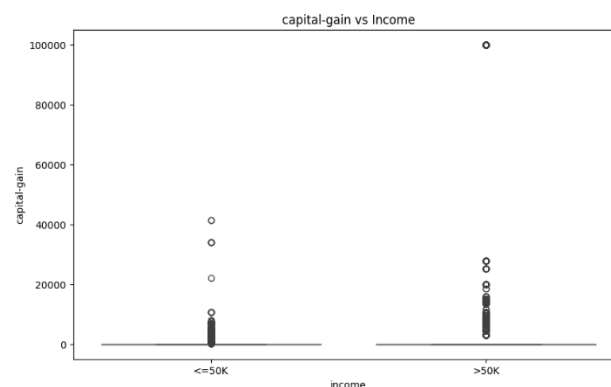
2.5.2.3 Analisa Hubungan Education-num Dengan Income



Penjelasan :

Diagram boxplot diatas ini menunjukkan bahwa distribusi education-num pada kedua kelompok yaitu yang memiliki pendapatan $\leq \$50K$ dan $>50K$ menunjukkan bahwa level edukasi orang-orang yang berpenghasilan $\leq \$50K$ memiliki nilai tengah pada level pendidikan 10 dan income yang $>50K$ rata-rata berada pada level edukasi 12. Titik-titik di atas garis menunjukkan outlier, yaitu individu yang level edukasinya lebih tinggi dan lebih kecil daripada sebagian besar data. Diagram ini menggambarkan bahwa rata-rata level edukasi orang dengan pendapatan lebih tinggi cenderung lebih tinggi dibandingkan kelompok dengan pendapatan lebih rendah.

2.5.2.4 Analisa Hubungan Capital-gain Dengan Income

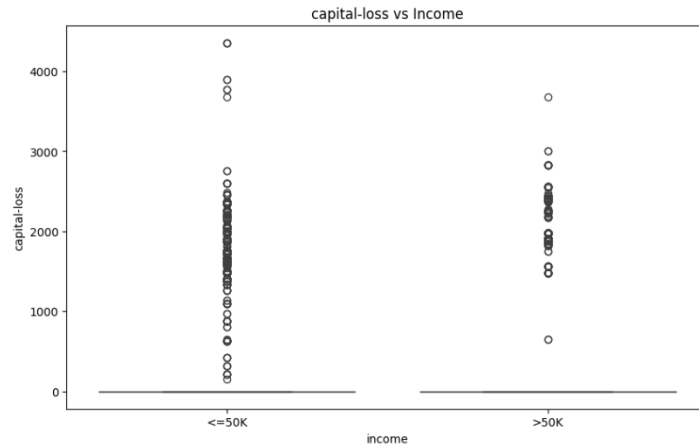


Penjelasan :

Diagram boxplot diatas menunjukkan distribusi Capital-gain dengan Income. Berdasarkan boxplot diatas data dengan pendapatan $\leq \$50K$ per tahun memiliki keuntungan yang kecil

dari data yang income >\$50K per tahun tetapi ada juga beberapa data yang memiliki keuntungan yang besar.

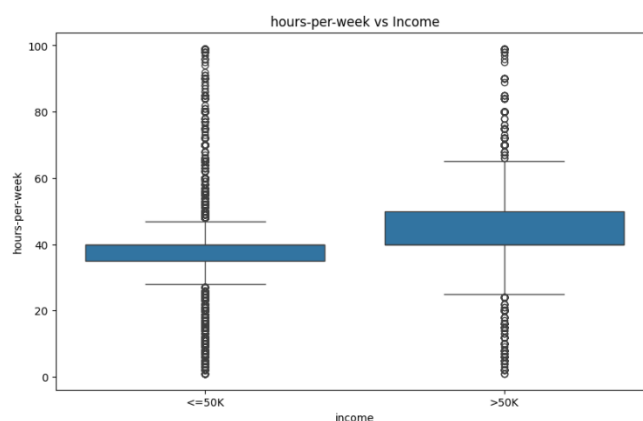
2.5.2.5 Analisa Hubungan Capital-loss Dengan Income



Penjelasan :

Diagram boxplot diatas menunjukkan distribusi Capital-loss dengan Income. Berdasarkan boxplot diatas data dengan pendapatan <=\$50K per tahun memiliki kerugian yang terbilang besar begitu juga dengan data yang income >\$50K per tahun. tetapi ada juga beberapa data yang memiliki kerugian yang lebih besar dan kerugian yang terlalu kecil.

2.5.2.6 Analisa Hubungan Hours-per-week Dengan Income



Penjelasan :

Berdasarkan data boxplot diatas ini, dapat disimpulkan bahwa rata-rata kerja per minggu untuk mendapatkan income <=\$50K adalah kisaran 35 hingga 40 jam per minggu, dari income

$\leq \$50K$ dapat disimpulkan ada juga *outlayer* bahwa hanya dengan kerja kurang dari 25 jam bisa mendapatkan income $\leq \$50K$ per tahun, banyak juga yang kerja lebih dari waktu batas wajar lebih dari 48 jam per minggu, pada income $> \$50K$ dan dapat dilihat bahwa rata-rata untuk mendapatkan income $> \$50K$ waktu bekerja setiap minggu adalah 40 hingga 50 jam per minggu, tetapi ada juga banyak *outlayer* yang hanya perlu bekerja kecil dari 18 jam per minggu dan ada juga yang bekerja lebih dari 63 jam per minggu.

2.5.3 Analisa Korelasi Variabel Kategorikal Dengan Variabel Target Dengan Stacked Bar Plot

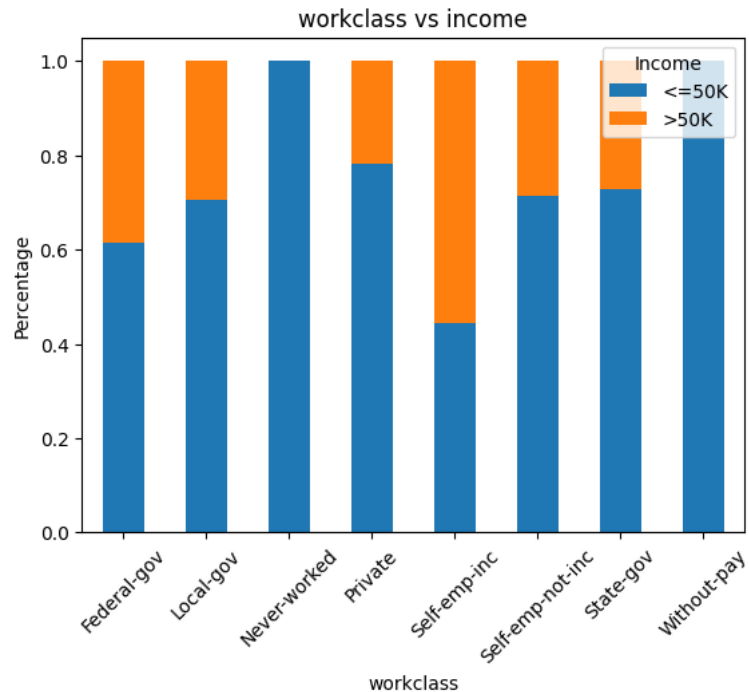
Untuk menampilkan stacked bar plot untuk setiap variabel dengan tipe data kategorikal dapat dengan cara dibawah ini :

```
for feature in categorical_features:
    if(feature != 'income'):
        plt.figure(figsize=(12,6))
        df_temp = df.groupby([feature, 'income']).size().unstack()
        df_temp_perc = df_temp.div(df_temp.sum(axis=1), axis=0)
        df_temp_perc.plot(kind='bar', stacked=True)
        plt.title(f'{feature} vs income')
        plt.xlabel(feature)
        plt.ylabel('Percentage')
        plt.legend(title='Income', loc='upper right')
        plt.xticks(rotation=45)
        plt.show()
```

Penjelasan :

Lakukan perulangan untuk setiap fitur kategori (seperti jenis kelamin, pekerjaan, dll.) dan menyimpannya sementara ke variabel **feature**, lalu memeriksa apakah fitur tersebut bukan **income** (karena **income** adalah target yang akan diprediksi). Jika bukan **income**, maka buat grafik batang bertumpuk untuk melihat bagaimana setiap kategori dalam fitur tersebut dibandingkan berdasarkan pendapatan (**income**). Grafik ini menampilkan persentase kelompok pendapatan ($\leq 50K$ dan $> 50K$) pada setiap kategori fitur, sehingga dapat memahami pengaruh fitur tersebut terhadap pendapatan.

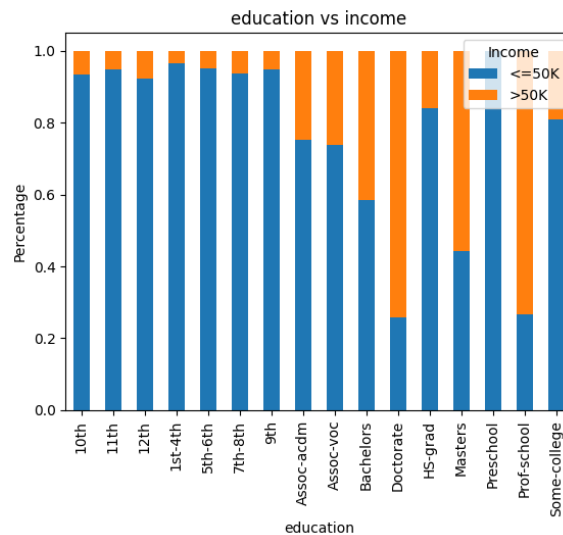
2.5.3.1 Analisa Stacked-bar-plot Variabel Workclass dengan Income



Penjelasan :

Dapat dilihat dari grafik diatas bahwa yang paling banyak pada income <=\$50K adalah Never-worked atau orang yang tidak bekerja, sedangkan income yang mencapai >\$50K adalah orang yang bekerja sebagai Self-emp-inc.

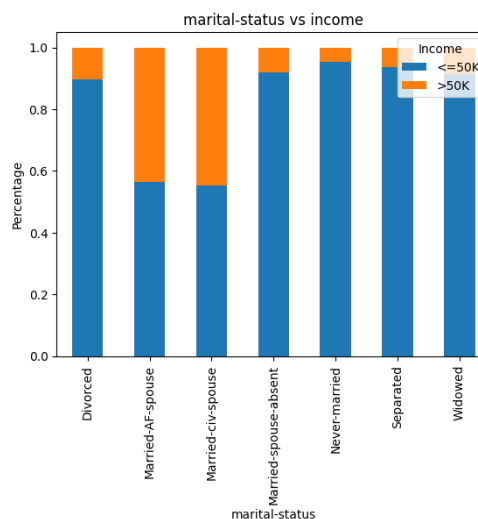
2.5.3.2 Analisa Stacked-bar-plot Variabel Education dengan Income



Penjelasan :

Pada grafik boxplot diatas terlihat bahwa lulusan edukasi pre-school rata-rata memiliki income terbesar pada $\leq \$50K$, dan income $\geq \$50K$ terbanyak adalah lulusan Doctorate.

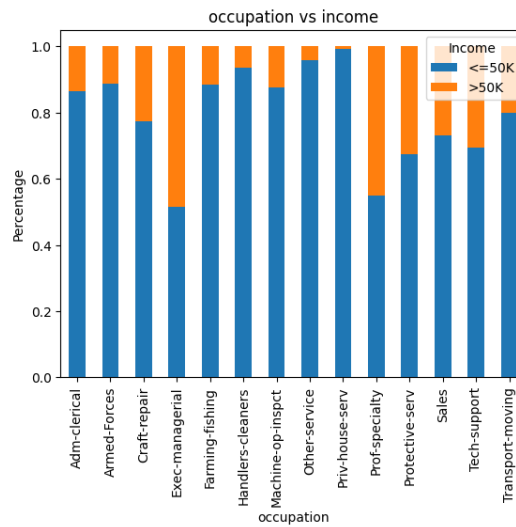
2.5.3.3 Analisa Stacked-bar-plot Variabel Marital-status dengan Income



Penjelasan :

Pada grafik diatas, terlihat bahwa data yang memiliki income $\leq \$50K$ terbanyak memiliki status pernikahan yaitu Never-married, dan untuk income $\geq \$50K$ yang terbanyak adalah dengan status pernikahan Married-civ-spouse.

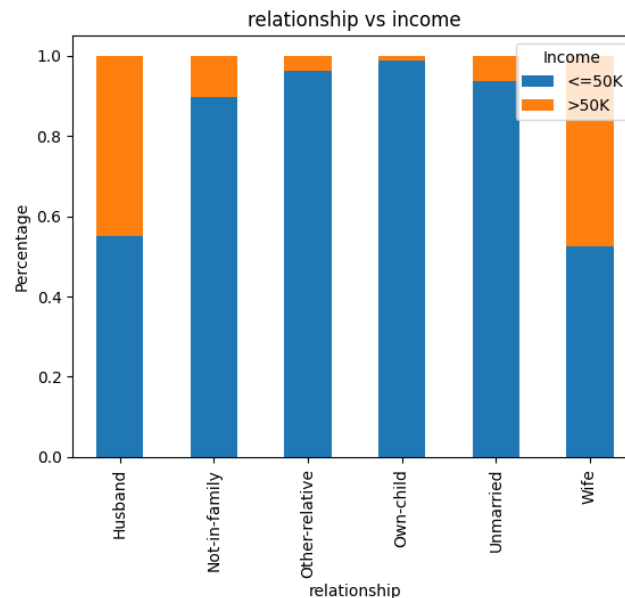
2.5.3.4 Analisa Stacked-bar-plot Variabel Occupation Dengan Income



Penjelasan :

Pada grafik diatas, terlihat bahwa income <=\$50K terbanyak adalah pekerjaan Priv-house-serv, sedangkan untuk income >\$50K terbanyak adalah pekerjaan Exec-managerial.

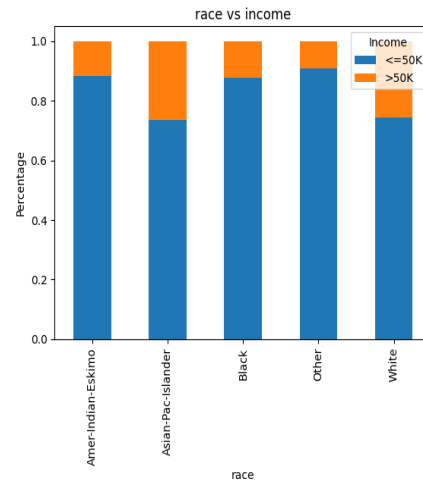
2.5.3.5 Analisa Stacked-bar-plot Variabel Relationship Dengan Income



Penjelasan :

Grafik diatas menjelaskan bahwa income $\leq \$50K$ terbanyak memiliki status relationship yaitu own-child, berbeda dengan income $> \$50K$ terbanyak memiliki status relationship Wife.

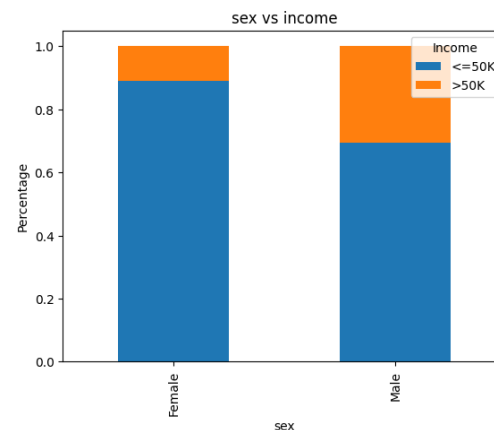
2.5.3.6 Analisa Stacked-bar-plot Variabel Race Dengan Income



Penjelasan :

Pada grafik diatas ini, terlihat bahwa orang yang memiliki income $\leq \$50K$ adalah ras Other, dan income $> \$50k$ terbanyak berasal dari ras Asian-Pac-islander.

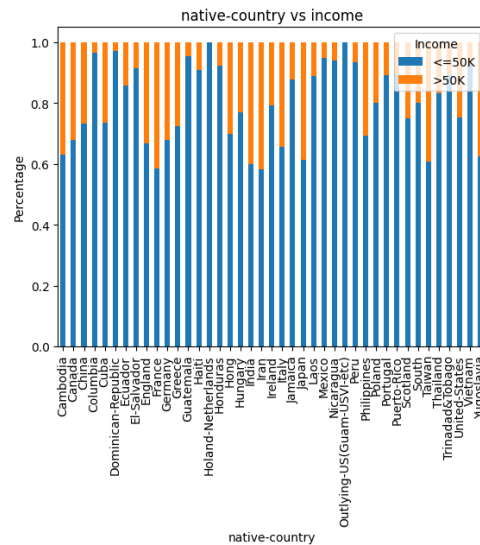
2.5.3.7 Analisa Stacked-bar-plot Variabel Sex dengan Income



Penjelasan :

Pada grafik diatas ini dapat dilihat bahwa yang memiliki income $\leq \$50K$ terbanyak adalah berjenis kelamin Female, sedangkan $> \$50K$ adalah Male.

2.5.3.8 Analisa Stacked-bar-plot Variabel Native-country dengan Income



Penjelasan :

Berdasarkan grafik diatas dapat disimpulkan bahwa negara yang memiliki income <=\$50K yang terbanyak adalah negara Holand-Netherlands dan negara Outlying-US, sedangkan untuk income >\$50K terbanyak adalah dari negara Iran dan France.

UNIT 3 : VALIDASI DATA

3.1. Missing Values

3.1.1. Memeriksa Missing Values

Untuk memeriksa *missing values*, dapat dilakukan dengan cara dibawah ini :

```
#Hitung jumlah missing values
missing_values= df.isnull().sum()

#Hitung persentase missing values
missing_percentage = 100* missing_values / len(df)

#Gabungkan informasi missing values
missing_table = pd.concat([missing_values, missing_percentage], axis=1, keys=['Total','Percent'])
print(missing_table)
```

Penjelasan :

Pertama, menghitung jumlah nilai yang hilang (*missing values*) untuk setiap kolom dalam dataset dan menyimpannya dalam variabel `missing_values`. Kemudian, hitung persentase nilai yang hilang untuk setiap kolom dengan

membandingkan jumlah nilai yang hilang dengan total baris di dataset, hasilnya disimpan di variabel `missing_percentage`. Setelah itu, gabungkan informasi jumlah total nilai yang hilang dan persentasenya ke dalam sebuah tabel baru yang disebut `missing_table`, sehingga bisa dilihat dengan jelas berapa banyak nilai yang hilang dan berapa persentasenya untuk setiap kolom.

OUTPUT :

	Total	Percent
age	0	0.000000
workclass	1836	5.638647
fnlwgt	0	0.000000
education	0	0.000000
education-num	0	0.000000
marital-status	0	0.000000
occupation	1843	5.660146
relationship	0	0.000000
race	0	0.000000
sex	0	0.000000
capital-gain	0	0.000000
capital-loss	0	0.000000
hours-per-week	0	0.000000
native-country	583	1.790486
income	0	0.000000

Penjelasan :

Variabel	Total Data Yang Hilang	Presentase Data Yang Hilang
Workclass	1.836	5,638647%
Occupation	1.843	5,660146%
Native-country	583	1,790486%

3.2 Memeriksa Duplikasi Data

Dengan menuliskan kode dibawah ini untuk menampilkan duplikasi data

```

duplicates = df.duplicated().sum()
print(f'Jumlah Baris Duplikat: {duplicates}')
if(duplicates > 0):
    print("\nContoh baris duplikat: ")
    print(df[df.duplicated(keep=False)].head())

```

Penjelasan :

Awalnya menghitung jumlah baris duplikat dalam dataset menggunakan fungsi **`df.duplicated()`** dan menyimpannya di variabel **`duplicates`**. Lalu, mencetak jumlah baris duplikat yang ditemukan. Jika ada baris duplikat (jumlahnya lebih dari 0), kemudian

menampilkan beberapa contoh baris duplikat tersebut. Untuk itu, gunakan **df[df.duplicated(keep=False)]** yang akan menampilkan semua baris yang duplikat tanpa mengabaikan salah satu barisnya, dan menampilkan beberapa baris pertama menggunakan **.head()**.

OUTPUT :

```
Jumlah Baris Duplikat: 24

Contoh baris duplikat:
  age workclass  fnlwtg  education  education-num  marital-status \
2303   90   Private   52386  Some-college          10  Never-married
3917   19   Private   251579  Some-college          10  Never-married
4325   25   Private   308144  Bachelors           13  Never-married
4767   21   Private   250051  Some-college          10  Never-married
4881   25   Private   308144  Bachelors           13  Never-married

  occupation  relationship  race  sex  capital-gain \
2303  Other-service  Not-in-family  Asian-Pac-Islander  Male      0
3917  Other-service    Own-child    White  Male      0
4325  Craft-repair  Not-in-family    White  Male      0
4767  Prof-specialty  Own-child    White  Female    0
4881  Craft-repair  Not-in-family    White  Male      0

  capital-loss  hours-per-week  native-country  income
2303          0              35  United-States  <=50K
3917          0              14  United-States  <=50K
4325          0              40    Mexico    <=50K
4767          0              10  United-States  <=50K
4881          0              40    Mexico    <=50K
```

Penjelasan :

Dari hasil pencarian duplikasi data diatas, kita mendapatkan bahwa ada **24** data yang terdeteksi duplikasi.

3.3 Validasi Tipe Data

Untuk melakukan validasi tipe data setiap kolom dengan cara dibawah ini:

```
#Tampilkan tipe data setiap kolom
print(df.dtypes)

#Periksa apakah ada nilai non-numerik dalam kolom numerik
numeric_columns = df.select_dtypes(include=[np.number]).columns
for col in numeric_columns:
    non_numeric = df[pd.to_numeric(df[col], errors='coerce').isna()]
    if(len(non_numeric) > 0):
        print(f'\nNilai non-numerik dalam kolom {col} :')
        print(non_numeric[col].unique())
```

Penjelasan :

Kita print terlebih dahulu tipe data untuk setiap kolom, kemudian kita akan lakukan pengecekan apakah ada nilai non-numerik di dalam kolom numerik pada kode perulangan tersebut.

OUTPUT :


```
age          int64
workclass    object
fnlwgt       int64
education    object
education-num int64
marital-status object
occupation   object
relationship object
race         object
sex          object
capital-gain  int64
capital-loss  int64
hours-per-week int64
native-country object
income       object
dtype: object
```

Penjelasan :

Dilihat pada output tipe data sudah sesuai dan tidak ada yang tercampur.

3.4 Validasi Nilai Range

Untuk melakukan validasi nilai range dari kolom education dan education-num dapat dilakukan dengan cara dibawah ini :

```
# periksa range nilai untuk kolom numerik
for col in numeric_columns:
    print(f'\nRange Nilai Untuk {col}: ')
    print(f'Min: {df[col].min()}, Max: {df[col].max()}')

# periksa kategori unik untuk kolom kategorikal
categorical_columns = df.select_dtypes(include=['object']).columns
for col in categorical_columns:
    print(f'\nKategori unik dalam {col} : ')
    print(df[col].unique())
```

OUTPUT :

```

Range Nilai Untuk age:
Min: 17, Max: 90

Range Nilai Untuk fnlwgt:
Min: 12285, Max: 1484705

Range Nilai Untuk education-num:
Min: 1, Max: 16

Range Nilai Untuk capital-gain:
Min: 0, Max: 99999

Range Nilai Untuk capital-loss:
Min: 0, Max: 4356

Range Nilai Untuk hours-per-week:
Min: 1, Max: 99

Kategori unik dalam workclass :
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov' nan
'Self-emp-inc' 'Without-pay' 'Never-worked']

Kategori unik dalam education :
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acads'
'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '18th'
'1st-4th' 'Preschool' '12th']

Kategori unik dalam marital-status :
['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
'Separated' 'Married-AF-spouse' 'Widowed']

Kategori unik dalam occupation :
['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
'Farming-fishing' 'Machine-op-inspct' 'Tech-support' nan
'Protective-serv' 'Armed-Forces' 'Priv-house-serv']

Kategori unik dalam relationship :
['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relative']

Kategori unik dalam race :
['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']

Kategori unik dalam sex :
['Male' 'Female']

Kategori unik dalam native-country :
['United-States' 'Cuba' 'Jamaica' 'India' nan 'Mexico' 'South'
'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinidad&Tobago' 'Greece'
'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']

Kategori unik dalam income :
['<=50K' '>50K']

```

Penjelasan :

- Kolom Tipe Data Numerik :

Variabel	Min	Max
Age	17	90
Fnlwgt	12285	1484705
Capital-gain	0	99999
Capital-loss	0	4356
Hours-per-week	1	99

- Kolom Tipe Data Kategorikal :

Variabel	Varian Data (<i>Unique</i>)
Workclass	['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov' nan 'Self-emp-inc' 'Without-pay' 'Never-worked']
Education	['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm' 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th' '1st-4th' 'Preschool' '12th']
Marital-status	['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent' 'Separated' 'Married-AF-spouse' 'Widowed']
Occupation	['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty' 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving' 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' nan 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
Relationship	['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relative']
Race	['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']
Sex	['Male' 'Female']
Native-country	['United-States' 'Cuba' 'Jamaica' 'India' nan 'Mexico' 'South' 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran' 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador']

	'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador' 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru' 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinidad&Tobago' 'Greece' 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand- Netherlands']
Income	['<=50K' '>50K']

3.5 Memeriksa Konsistensi Data

3.5.1. Konsistensi Data Education Dengan Education-num

Untuk Melakukan pemeriksaan konsistensi education dengan education-num :

```
#Contoh periksa konsistensi antara 'education' dan 'education_num'
education_mapping = df.groupby('education')['education-num'].mean().sort_values()
print("\nPemetaan rata-rata 'education-num' untuk setiap 'education' :")
print(education_mapping)
```

Penjelasan :

Memeriksa konsistensi antara kolom **'education'** dan **'education-num'** dengan membuat pemetaan rata-rata dari **'education-num'** untuk setiap kategori **'education'**. Pertama, kita mengelompokkan data berdasarkan kolom **'education'** dan menghitung nilai rata-rata **'education-num'** untuk setiap kelompok. Hasilnya diurutkan dari yang terkecil hingga terbesar menggunakan **sort_values()**, dan kemudian menampilkan pemetaan tersebut. Ini membantu agar dapat memeriksa apakah nilai numerik pada **'education-num'** secara konsisten mencerminkan tingkatan pendidikan yang ada di kolom **'education'**.

OUTPUT :

```

▶ Pemetaan rata-rata 'education-num' untuk setiap 'education' :
↳
education
Preschool      1.0
1st-4th        2.0
5th-6th        3.0
7th-8th        4.0
9th            5.0
10th           6.0
11th           7.0
12th           8.0
HS-grad        9.0
Some-college   10.0
Assoc-voc     11.0
Assoc-acdm    12.0
Bachelors     13.0
Masters       14.0
Prof-school   15.0
Doctorate     16.0
Name: education-num, dtype: float64

```

Penjelasan :

Bukti list yang ada pada gambar tersebut bahwa education-num secara konsisten mencerminkan tingkatan pendidikan yang ada pada kolom education.

3.5.2. Visualisasi Konsistensi Data Education Dengan Education-num

Untuk melakukan visualisasi konsistensi data adalah sebagai berikut :

```

▶ plt.figure(figsize=(10,6))
  education_mapping.plot(kind='bar')
  plt.title('numerisasi education-num untuk setiap kategori Education')
  plt.xlabel('Education')
  plt.ylabel('Rata-rata education-num')
  plt.xticks(rotation=45, ha='right')
  plt.tight_layout()
  plt.show()

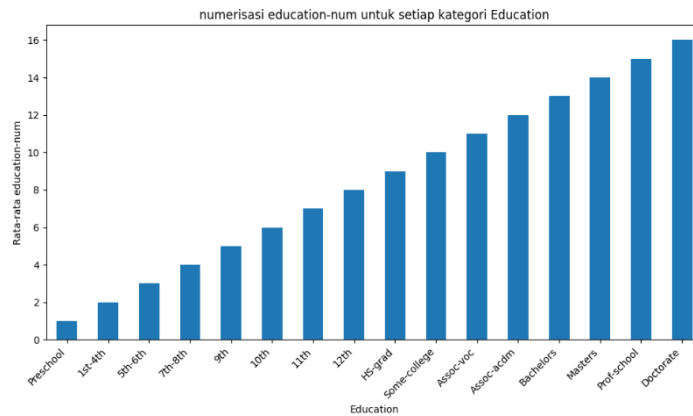
```

Penjelasan:

Awalnya membuat grafik batang (bar chart) untuk menampilkan pemetaan antara kategori 'education' dan rata-rata 'education-num'. Pertama, kita menentukan ukuran gambar dengan `plt.figure(figsize=(10,6))`. Lalu, memplot grafik batang menggunakan `education_mapping.plot(kind='bar')`. Judul grafik diatur menjadi "numerisasi education-num untuk setiap kategori Education" dan label sumbu X serta Y diatur masing-masing menjadi "Education" dan "Rata-rata education-num". Selain itu, kita memutar label sumbu X sebanyak 45 derajat agar lebih mudah dibaca, dengan penyesuaian posisi teksnya menggunakan `ha='right'`. Terakhir, kita

menggunakan `plt.tight_layout()` untuk memastikan elemen-elemen grafik tidak bertumpuk, lalu menampilkan grafik tersebut dengan `plt.show()`.

OUTPUT :



UNIT 4 : Mendefinisikan Data Object

4.1.Menentukan Objek Data

Memutuskan kriteria dan teknik pemilihan data.

4.1.1. Kuantitas dari dataset Adult Income adalah :

- Memiliki 14 *Features* (*variabel*)
- Terdiri dari 48.842 *Instance* (*record*)
- Ukuran data sebesar 3,79 MB

4.1.2. Kualitas dataset Adult Income yang digunakan adalah :

- Digunakan untuk Task : *Classification*
- *Feature Type* : *Categorical* dan *Integer*
- Memiliki *Missing Values*

4.1.3. Menentukan *Attributes* (*Columns*) dan *records* (*row*) data :

4.1.3.1. Menentukan *Attributes* (*Columns*)

- *age* : *Feature* : *Integer*
- *workclass* : *Feature* : *Categorical*
- *fnlwgt* : *Feature* : *Integer*
- *education* : *Feature* : *Categorical*
- *education-num* : *Feature* : *Integer*
- *marital-status* : *Feature* : *Integer*

- *occupation : Feature : Categorical*
- *relationship : Feature : Categorical*
- *race : Feature : Categorical*
- *sex : Feature : Binary*
- *capital-gain : Feature : Integer*
- *capital-loss : Feature : Integer*
- *hours-per-week : Feature : Integer*
- *native-country : Feature : Categorical*
- *income : Target : Binary*

4.1.3.2. Menentukan *records* (row)

Menampilkan baris dataset teratas :

age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K

- age: 39
- workclass: State-gov
- fnlwgt: 77516
- education: Bachelors
- education_num: 13
- marital_status: Never-married
- occupation: Adm-clerical
- relationship: Not-in-family
- race: White
- sex: Male
- capital_gain: 2174
- capital_loss: 0
- hours_per_week: 40
- native_country: United-States
- income: <=50K

UNIT 5 : Membersihkan Data

5.1. Menangani *Missing Values*

```
# Cek missing values
print("Missing values sebelum pembersihan:")
print(df.isnull().sum())

# Menangani missing values
for column in df.columns:
    if df[column].dtype == 'object':
        # Untuk kolom kategorikal, isi dengan modus
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        # Untuk kolom numerik, isi dengan median
        df[column].fillna(df[column].median(), inplace=True)

print("\nMissing values setelah pembersihan:")
print(df.isnull().sum())
```

```
Missing values sebelum pembersihan:
age          0
workclass    1836
fnlwgt       0
education    0
education_num 0
marital_status 0
occupation   1843
relationship 0
race         0
sex          0
capital_gain 0
capital_loss 0
hours_per_week 0
native_country 583
income       0
dtype: int64
```

```
Missing values setelah pembersihan:
age          0
workclass    0
fnlwgt       0
education    0
education_num 0
marital_status 0
occupation   0
relationship 0
race         0
sex          0
capital_gain 0
capital_loss 0
hours_per_week 0
native_country 0
income       0
dtype: int64
```

Pada pembersihan data, kita melakukan pengisian data, dimana jika data tersebut memiliki tipe numerik, maka kita akan mengisinya dengan nilai median, jika data bertipe kategorikal maka akan diisi dengan modus, sehingga total semua *missing values*nya menjadi 0

5.2. Menangani *Outlier*

```
def plot_boxplot(df, column):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot of {column}')
    plt.show()

# Contoh untuk kolom numerik
numeric_columns = df.select_dtypes(include=[np.number]).columns

for column in numeric_columns:
    plot_boxplot(df, column)

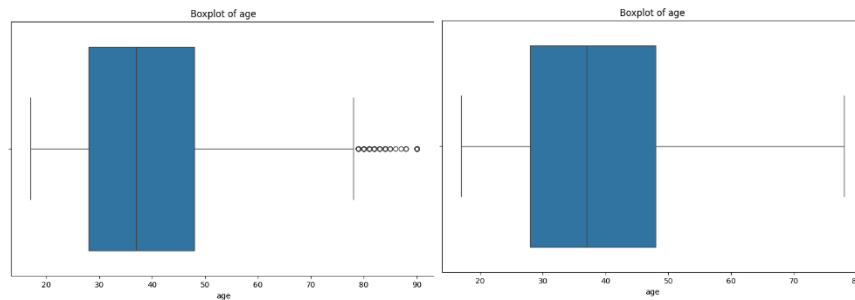
# Menangani outlier dengan IQR method
Q1 = df[column].quantile(0.25)
Q3 = df[column].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df[column] = np.where(df[column] > upper_bound, upper_bound,
                     np.where(df[column] < lower_bound, lower_bound, df[column]))

print(f"Outliers pada {column} sudah dihandle.")
plot_boxplot(df, column)
```


Kode diatas ebrfungsi untuk menangani outliers dan menampilkan boxplot grafik sebelum dan sesudah ditangani.

5.2.1. Menangani *outliers* pada *age*



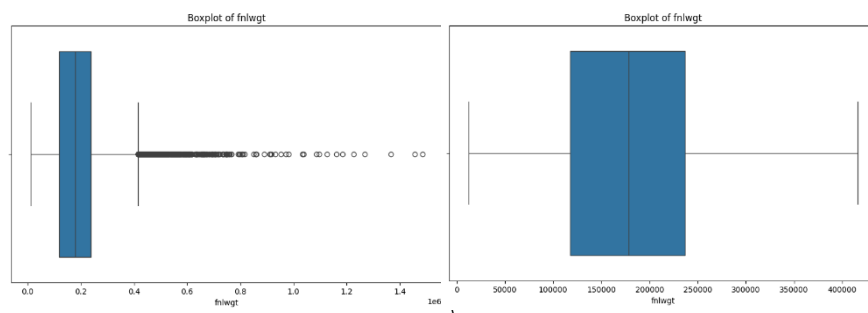
1. Sebelum handle *outliers*

2. Sesudah dihandle *outliers*

Kesimpulan :

Pada gambar pertama, terdapat **outlier** yang ditunjukkan oleh titik-titik di sebelah kanan garis whisker (usia di atas 80 tahun). Sedangkan pada gambar kedua, outlier telah dihilangkan atau disesuaikan, sehingga whisker meluas ke nilai maksimum yang lebih rendah, dan titik-titik outlier tidak lagi muncul. Hal ini menunjukkan bahwa data di gambar kedua lebih rapi dan tidak memiliki nilai ekstrem yang bisa mempengaruhi analisis.

5.2.2. Menangani *outliers* pada *fnlwtg*



1. Sebelum handle *outliers*

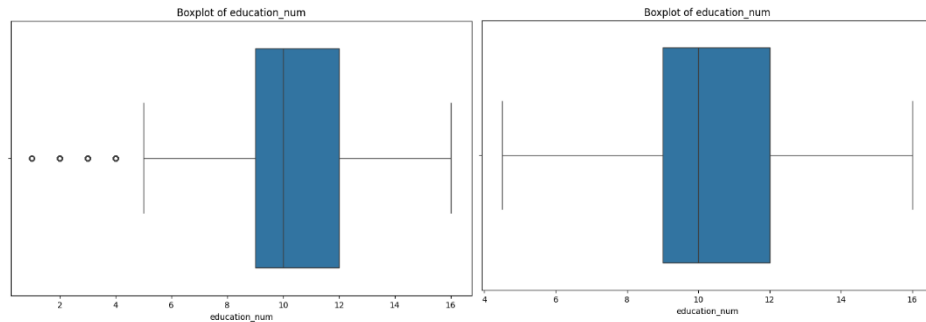
2. Sesudah dihandle *outliers*

Kesimpulan :

Pada gambar pertama, terlihat banyak **outlier** di sisi kanan, ditunjukkan oleh titik-titik yang jauh dari garis whisker, yang menunjukkan nilai *fnlwtg* yang sangat tinggi. Setelah penanganan outlier menggunakan metode **IQR** (gambar kedua), distribusi data menjadi lebih rapi, whisker memanjang hingga batas baru, dan titik-titik outlier tidak lagi muncul. Ini menunjukkan

bahwa nilai ekstrem telah diatasi atau disesuaikan agar tidak mempengaruhi analisis data.

5.2.3. Menangani *outliers* pada *education_num*



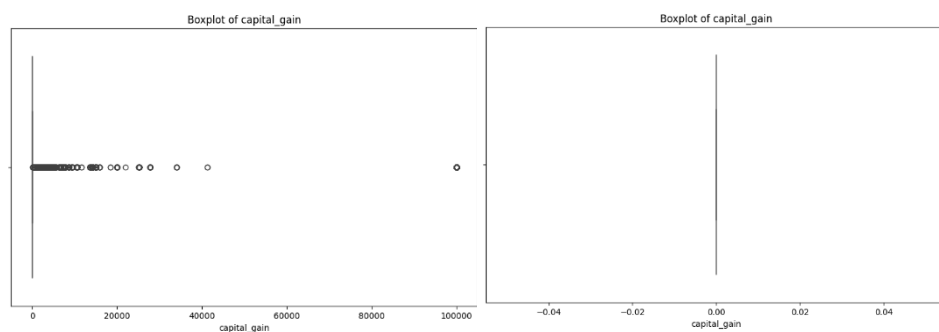
1. Sebelum handle outliers

2. Sesudah dihandle outliers

Kesimpulan :

Pada gambar pertama, terdapat beberapa **outlier** di sisi kiri (nilai lebih kecil dari whisker bawah) yang ditampilkan sebagai titik-titik terpisah. Setelah penanganan outlier pada gambar kedua, titik-titik tersebut tidak lagi muncul karena nilainya telah disesuaikan menggunakan metode **IQR**. Distribusi data menjadi lebih terpusat dan rapi, dengan whisker memanjang sesuai rentang data baru tanpa adanya nilai ekstrem yang mempengaruhi tampilan grafik.

5.2.4. Menangani *outliers* pada *capital_gain*



1. Sebelum handle outliers

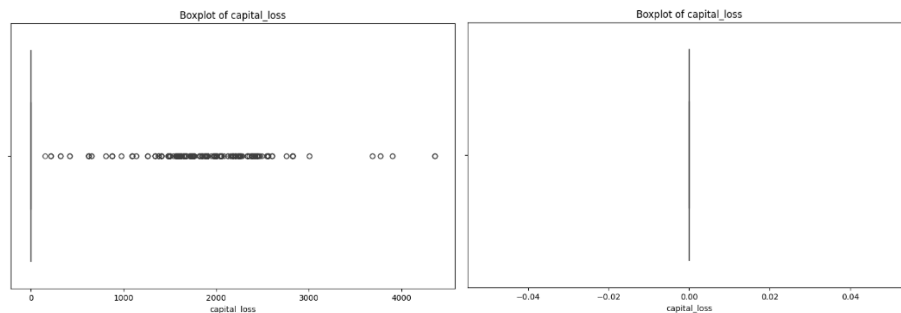
2. Sesudah dihandle outliers

Kesimpulan :

Pada gambar pertama, terdapat banyak outlier yang tersebar jauh di sisi kanan (nilai *capital_gain* sangat tinggi), terlihat dari titik-titik yang berada di luar whisker. Namun, pada gambar kedua setelah penanganan outlier, distribusi menjadi sangat sempit dengan semua nilai diatur mendekati nol,

menunjukkan bahwa nilai ekstrem telah digantikan atau disesuaikan. Hal ini membuat data terlihat terpusat tetapi dengan kehilangan variasi yang signifikan, sehingga kemungkinan besar `capital_gain` menjadi sangat seragam setelah transformasi.

5.2.5. Menangani *outliers* pada `capital_loss`



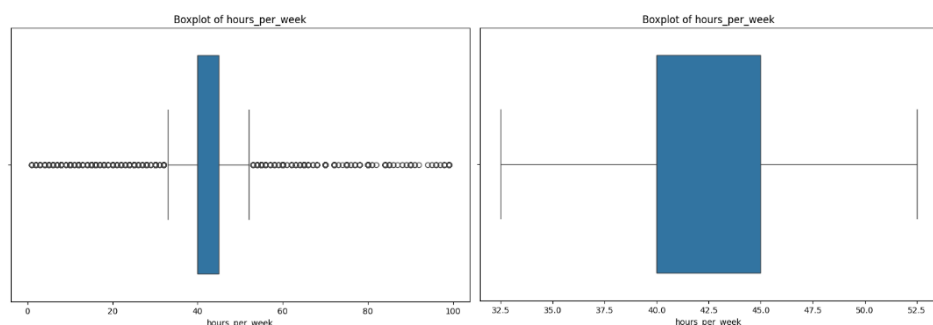
1. Sebelum handle *outliers*

2. Sesudah dihandle *outliers*

Kesimpulan :

Pada gambar pertama, terdapat banyak titik **outlier** yang tersebar di sepanjang sumbu horizontal, menunjukkan nilai-nilai `capital_loss` yang signifikan dan tersebar luas. Setelah penanganan outlier menggunakan metode IQR (gambar kedua), distribusi data menjadi sangat sempit dan mendekati nol, ditunjukkan oleh garis vertikal tipis tanpa titik outlier. Ini menunjukkan bahwa nilai-nilai ekstrem telah dikoreksi atau dibatasi, tetapi variasi data menjadi sangat kecil, membuat distribusi terlihat hampir seragam dan terpusat.

5.2.6. Menangani *outliers* pada `hours_per_week`



1. Sebelum handle *outliers*

2. Sesudah dihandle *outliers*

Kesimpulan :

Pada gambar pertama, terlihat banyak **outlier** di kedua sisi, yang menunjukkan variasi jam kerja yang sangat besar, termasuk nilai ekstrem rendah dan tinggi. Setelah penanganan outlier menggunakan metode **IQR** (gambar kedua), distribusi data menjadi lebih terpusat dengan rentang antara sekitar 33 hingga 52 jam per minggu, dan titik-titik outlier tidak lagi terlihat. Hal ini menunjukkan bahwa nilai-nilai ekstrem telah dikoreksi, sehingga data menjadi lebih rapi dan fokus pada jam kerja yang umum terjadi, seperti sekitar 40 jam per minggu.

5.3. Menangani Duplikat

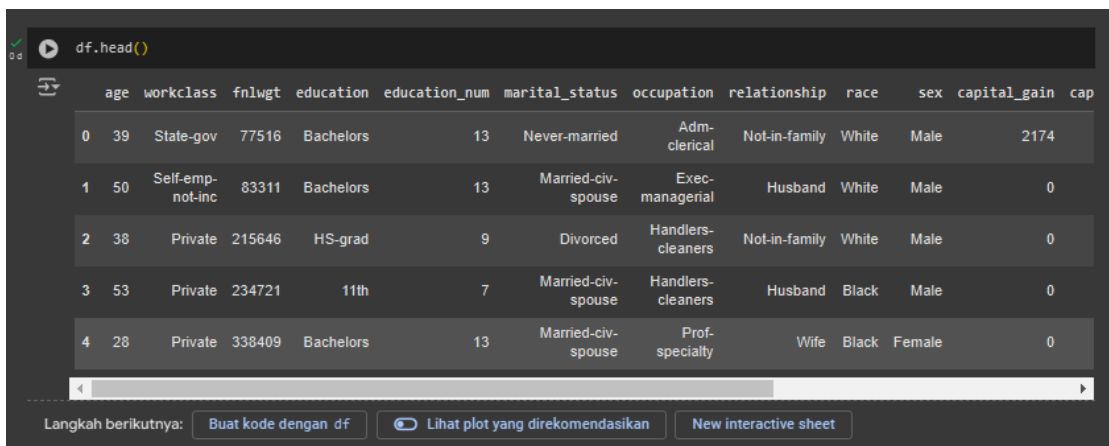
```
# Cek duplikat
duplicate_count = df.duplicated().sum()
print(f"Jumlah baris duplikat: {duplicate_count}")

# Hapus duplikat
df.drop_duplicates(inplace=True)

print(f"Jumlah baris setelah menghapus duplikat: {len(df)}")
```

Pada kode ini, kita akan menghapus baris yang duplikat sehingga tidak ada baris yang memiliki nilai yang sama.

5.4. Validasi Hasil



	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	cap
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	

Disini kita akan melakukan validasi ulang terhadap isi data yang sudah dibersihkan.

5.5. Menyimpan Dataset Yang Sudah di Bersihkan

```
# Simpan data yang telah dibersihkan
sys.path.append(f'{folder_name}')
df.to_csv(f'{folder_name}/adult_income_cleaned.csv', index=False)
print("Data yang telah dibersihkan telah disimpan sebagai 'adult_income_cleaned.csv'")
```

Data yang telah dibersihkan telah disimpan sebagai 'adult_income_cleaned.csv'

Pada kode ini, kita akan melakukan export ke csv data yang sudah dibersihkan dengan nama `adult_income_cleaned.csv` di dalam folder yang sama dengan file `adult.data`

UNIT 6 : Rekonstruksi Data

6.1. Pengelompokan Data

```
# membuat fitur baru berdasarkan usia
df['age_group'] = pd.cut(df['age'], bins=[0,18,30,45,60,100], labels=["Remaja", "Dewasa Muda", "Dewasa", "Paruh Baya", "Lansia"])

# menggabungkan capital_gain dan capital_loss
df['net_capital'] = df['capital_gain'] - df['capital_loss']

# membuat fitur rasio jam kerja terhadap rata-rata jam kerja
df['work_hours_ratio'] = df['hours_per_week'] / df['hours_per_week'].mean()

# membuat fitur kategorikal baru berdasarkan education_num
df['education_level'] = pd.cut(df['education_num'], bins=[0,8,12,16,20], labels=["Dasar", "Menengah", "Sarjana", "Pascasarjana"])

df[['age_group', 'net_capital', 'work_hours_ratio', 'education_level']].head()
```

Pada kode ini, kita melakukan pengelompokan data berdasarkan data yang ada, dan kemudian ditampilkan 5 data pertama.

Tampilan 5 data pertama pada kolom yang baru dibuat:

	age_group	net_capital	work_hours_ratio	education_level
0	Dewasa	0.0	0.970669	Sarjana
1	Paruh Baya	0.0	0.788668	Sarjana
2	Dewasa	0.0	0.970669	Menengah
3	Paruh Baya	0.0	0.970669	Dasar
4	Dewasa Muda	0.0	0.970669	Sarjana

Kemudian lakukan pengecekan info dataset :

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 32508 entries, 0 to 32560
Data columns (total 19 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   age                 32508 non-null  float64
 1   workclass           32508 non-null  object  
 2   fnlwgt              32508 non-null  float64
 3   education           32508 non-null  object  
 4   education_num       32508 non-null  float64
 5   marital_status      32508 non-null  object  
 6   occupation          32508 non-null  object  
 7   relationship        32508 non-null  object  
 8   race                32508 non-null  object  
 9   sex                 32508 non-null  object  
10   capital_gain        32508 non-null  float64
11   capital_loss        32508 non-null  float64
12   hours_per_week      32508 non-null  float64
13   native_country      32508 non-null  object  
14   income              32508 non-null  object  
15   age_group           32508 non-null  category
16   net_capital         32508 non-null  float64
17   work_hours_ratio    32508 non-null  float64
18   education_level     32508 non-null  category
dtypes: category(2), float64(8), object(9)
memory usage: 4.5+ MB
```

Terlihat dari gambar diatas kolom yang baru dibuat untuk dilakukan pengelompokan sudah ada pada dataset.

6.2. Transformasi Data

6.2.1. Pengecekan Value Unik Untuk Data kategorikal

```
# Periksa kategori unik untuk kolom kategorikal
categorical_columns = df.select_dtypes(include=['object']).columns
for col in categorical_columns:
    print(f"\nKategori unik dalam kolom {col}:")
    print(df[col].unique())
```

Pada kode diatas kita akan mencetak kategori yang unik untuk dapat menjadi patokan konversi ke numeriknya.

Hasil :

```
Kategori unik dalam kolom workclass:
['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov'
 'Self-emp-inc' 'Without-pay' 'Never-worked']

Kategori unik dalam kolom education:
['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']

Kategori unik dalam kolom marital_status:
['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']

Kategori unik dalam kolom occupation:
['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' 'Protective-serv'
 'Armed-Forces' 'Priv-house-serv']

Kategori unik dalam kolom relationship:
['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relative']

Kategori unik dalam kolom race:
['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']

Kategori unik dalam kolom sex:
['Male' 'Female']

Kategori unik dalam kolom native_country:
['United-States' 'Cuba' 'Jamaica' 'India' 'Mexico' 'South' 'Puerto-Rico'
 'Honduras' 'England' 'Canada' 'Germany' 'Iran' 'Philippines' 'Italy'
 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador' 'Laos' 'Taiwan'
 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador' 'France'
 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinidad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']

Kategori unik dalam kolom income:
['<=50K' '>50K']
```

6.2.2. Transformasi Variabel kategorikal(*object*) menjadi variabel numerik

```
# lakukan mapping data kategorikal menjadi numerik

# workclass
workclass = {'State-gov': 7, 'Self-emp-not-inc': 6, 'Private': 5, 'Federal-gov': 4, 'Local-gov': 3,
             'Self-emp-inc': 2, 'Without-pay': 1, 'Never-worked': 0}
df['workclass'] = df['workclass'].map(workclass)

# education tidak perlu dilakukan mapping, karena sudah di mapping pada education_num

# marital_status
marital_status = {
    'Never-married': 6,
    'Married-civ-spouse': 5,
    'Divorced': 4,
    'Married-spouse-absent': 3,
    'Separated': 2,
    'Married-AF-spouse': 1,
    'Widowed': 0
}
df['marital_status'] = df['marital_status'].map(marital_status)

# occupation
occupation = {
    'Adm-clerical': 13,
    'Exec-managerial': 12,
    'Handlers-cleaners': 11,
    'Prof-specialty': 10,
    'Other-service': 9,
    'Sales': 8,
    'Craft-repair': 7,
    'Transport-moving': 6,
    'Farming-fishing': 5,
    'Machine-op-inspct': 4,
    'Tech-support': 3,
    'Protective-serv': 2,
    'Armed-Forces': 1,
    'Priv-house-serv': 0
}
df['occupation'] = df['occupation'].map(occupation)

# relationship
relationship = {
    'Not-in-family': 5,
    'Husband': 4,
    'Wife': 3,
    'Own-child': 2,
    'Unmarried': 1,
    'Other-relative': 0
}
df['relationship'] = df['relationship'].map(relationship)

# race
race = {
    'White': 4,
    'Black': 3,
    'Asian-Pac-Islander': 2,
    'Amer-Indian-Eskimo': 1,
    'Other': 0
}
df['race'] = df['race'].map(race)

# sex
sex = {'Male': 1, 'Female': 0}
df['sex'] = df['sex'].map(sex)

# native_country
native_country = {
    'United-States': 40,
    'Cuba': 39,
    'Jamaica': 38,
    'India': 37,
    'Mexico': 36,
    'South': 35,
    'Puerto-Rico': 34,
    'Honduras': 33,
    'England': 32,
    'Canada': 31,
    'Germany': 30,
    'Iran': 29,
    'Philippines': 28,
    'Italy': 27,
    'Poland': 26,
    'Columbia': 25,
    'Cambodia': 24,
    'Thailand': 23,
    'Ecuador': 22,
    'Lao': 21,
    'Taiwan': 20,
    'Haiti': 19,
    'Portugal': 18,
    'Dominican-Republic': 17,
    'El-Salvador': 16,
    'France': 15,
    'Guatemala': 14,
    'China': 13,
    'Japan': 12,
    'Yugoslavia': 11,
    'Peru': 10,
    'Outlying-US(Guam-USVI-etc)': 9,
    'Scotland': 8,
    'TrinidadTobago': 7,
    'Greece': 6,
    'Nicaragua': 5,
    'Vietnam': 4,
    'Hong': 3,
    'Ireland': 2,
    'Hungary': 1,
    'Holand-Netherlands': 0
}
df['native_country'] = df['native_country'].map(native_country)

# income
income = {'<=50K': 1, '>50K': 0}
df['income'] = df['income'].map(income)
```

```
# Tambahan

# age_group
age_group = {'Remaja': 1, 'Dewasa Muda': 2, 'Dewasa': 3, 'Paruh Baya': 4, 'Lansia': 5}
df['age_group'] = df['age_group'].map(age_group)

# education_level
education_level = {'Dasar': 1, 'Menengah': 2, 'Sarjana': 3, 'Pascasarjana': 4}
df['education_level'] = df['education_level'].map(education_level)
```

Setelah kita mendapatkan data unik dari variabel kategorikal, langkah selanjutnya adalah melakukan konversi data dengan cara mapping. Mapping dilakukan pada kolom yang

ingin dikonversi dengan menggunakan dictionary yang berisi pasangan nilai unik dari variabel kategorikal sebagai kunci (key) dan nilai numerik sebagai nilai (value). Dictionary ini akan dijadikan patokan oleh Pandas untuk mengubah data kategorikal menjadi data numerik.

Proses ini dilakukan menggunakan fungsi `.map()` dari Pandas yang akan menggantikan nilai pada kolom sesuai dengan dictionary yang telah dibuat.

6.3. Menampilkan Data Yang Sudah Dikonversi

```
df.head()
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week
0	39.0	7	77516.0	Bachelors	13.0	6	13	5	4	1	0.0	0.0	40.0
1	50.0	6	83311.0	Bachelors	13.0	5	12	4	4	1	0.0	0.0	32.5
2	38.0	5	215646.0	HS-grad	9.0	4	11	5	4	1	0.0	0.0	40.0
3	53.0	5	234721.0	11th	7.0	5	11	4	3	1	0.0	0.0	40.0
4	28.0	5	338409.0	Bachelors	13.0	5	10	3	3	0	0.0	0.0	40.0

Terlihat semua value sudah berubah menjadi angka, namun kita tetap perlu melakukan pengecekan info dataset apakah semuanya benar bertipe int/float64

Melakukan pengecekan info dataset :

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 32508 entries, 0 to 32560
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   age                   32508 non-null  float64
 1   workclass             32508 non-null  int64  
 2   fnlwgt                32508 non-null  float64
 3   education             32508 non-null  object  
 4   education_num         32508 non-null  float64
 5   marital_status        32508 non-null  int64  
 6   occupation            32508 non-null  int64  
 7   relationship          32508 non-null  int64  
 8   race                  32508 non-null  int64  
 9   sex                   32508 non-null  int64  
10   capital_gain          32508 non-null  float64
11   capital_loss          32508 non-null  float64
12   hours_per_week        32508 non-null  float64
13   native_country        32508 non-null  int64  
14   income                32508 non-null  int64  
15   age_group             32508 non-null  category
16   net_capital           32508 non-null  float64
17   work_hours_ratio      32508 non-null  float64
18   education_level       32508 non-null  category
dtypes: category(2), float64(8), int64(8), object(1)
memory usage: 4.5+ MB
```

Terlihat bahwa `age_group` dan juga `education_level` memiliki tipe data categorical, untuk itu kita perlu konversi tipe datanya yang sebelumnya *category* menjadi int/float64


```

0d # konversi tipe data kategorikal menjadi int/float64
0d # contoh :
    # df['nama_kolom'] = df['nama_kolom'].astype(str).astype(float)

    df['age_group'] = df['age_group'].astype(str).astype(float)
    df['education_level'] = df['education_level'].astype(str).astype(float)

[140] # hapus education karena sudah ada education_num
    df.drop('education', axis=1, inplace=True)

```

Pada kode diatas, kita melakukan konversi mengubah *category* menjadi string dulu, kemudian dikonversikan lagi menjadi float.

Dan kita akan menghapus kolom education yang sudah tidak digunakan karena sudah ada variabel feature education_num

Lakukan pengecekan info dataset lagi apakah sudah berubah tipe datanya

```

df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 32508 entries, 0 to 32560
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  -
0   age                    32508 non-null  float64
1   workclass              32508 non-null  int64
2   fnlwgt                 32508 non-null  float64
3   education_num          32508 non-null  float64
4   marital_status         32508 non-null  int64
5   occupation             32508 non-null  int64
6   relationship           32508 non-null  int64
7   race                   32508 non-null  int64
8   sex                    32508 non-null  int64
9   capital_gain           32508 non-null  float64
10  capital_loss           32508 non-null  float64
11  hours_per_week         32508 non-null  float64
12  native_country         32508 non-null  int64
13  income                  32508 non-null  int64
14  age_group              32508 non-null  float64
15  net_capital            32508 non-null  float64
16  work_hours_ratio       32508 non-null  float64
17  education_level        32508 non-null  float64
dtypes: float64(10), int64(8)

```

Terlihat pada kode diatas bahwa sudah berubah yang sebelumnya age_group dan education_level bertipe *category* berubah menjadi float64.

6.4. Melakukan Pengecekan Kembali Korelasi Antar *Features*

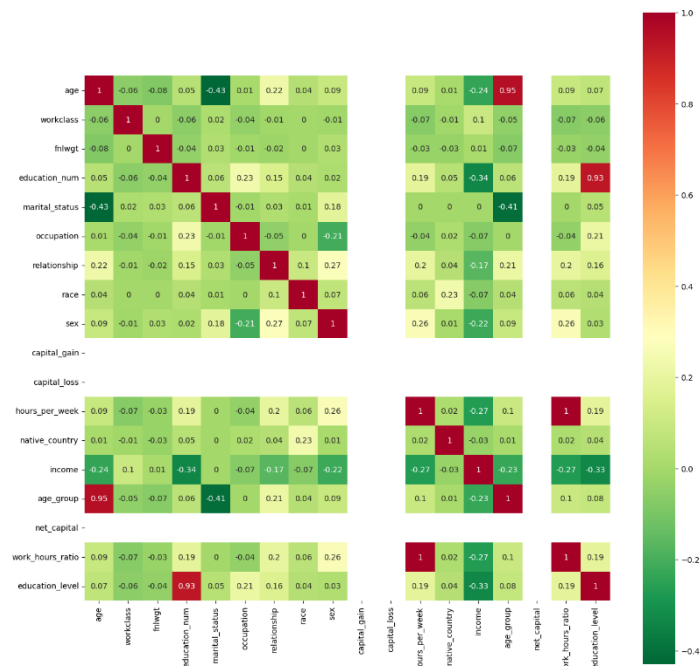
```

# Correlation Heatmap
correlation = df.corr()
plt.subplots(figsize=(15,15))
sns.heatmap(correlation.round(2),
            annot=True,
            vmax=1,
            square=True,
            cmap='RdYlGn_r')

plt.show()

```

kode diatas ini berfungsi agar dapat menampilkan korelasi terbaru pada data yang sudah dikonversi.



6.4.1. Analisis Heatmap Korelasi

Dari heatmap, terlihat bahwa beberapa fitur tidak ada karena sudah dihapus pada saat proses pembersihan data.

Features dengan korelasi tinggi :

- Age vs age_group
- Education_num vs education_level
- Work_hours_ratio vs hours_per_week

6.5. Penghapusan Features yang Bernilai Konstan

```
# menghapus kolom yang memiliki nilai konstan
df = df.loc[:, df.apply(pd.Series.nunique) != 1]
df
```

Pada kode ini, kita akan melakukan penghapusan pada kolom yang memiliki nilai konstan (nilai yang sama dari baris awal hingga baris terakhir).

Seperti pada dataset yang digunakan pada portofolio ini, ada 3 kolom yang dihapus yaitu :

- Capital_gain
- Capital_loss
- Net_capital

6.6. Penghapusan *Features* Berkorelasi Tinggi

```
# find and remove correlated features
def correlation(dataset, threshold):
    col_corr = set() # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i] # getting the name of column
                col_corr.add(colname)
    return col_corr

[145] data_tanpa_target = df.drop('income', axis=1)
      correlated_features = correlation(data_tanpa_target, 0.8)
      print("fitur berkorelasi tinggi yang akan dihapus :")
      correlated_features

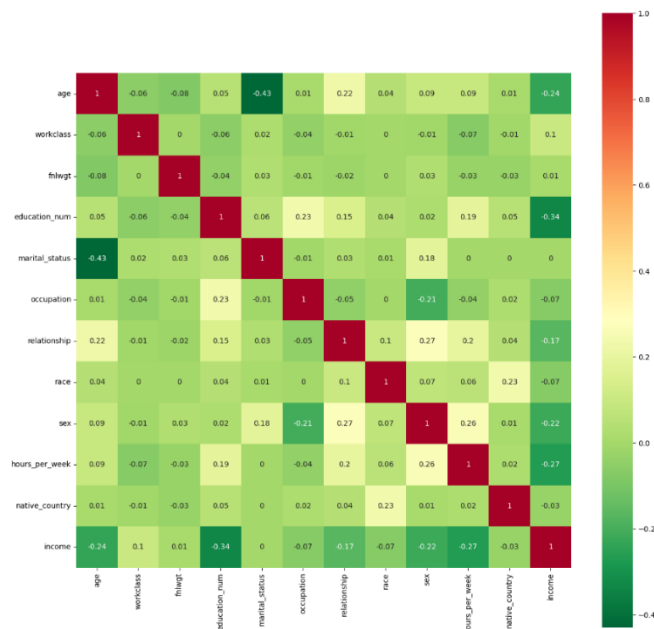
fitur berkorelasi tinggi yang akan dihapus :
{'age_group', 'education_level', 'work_hours_ratio'}

[146] df.drop(labels=correlated_features, axis=1, inplace=True)
```

Fungsi `correlation` dibuat untuk menghitung matriks korelasi antar fitur, lalu mengecek nilai korelasi absolut di atas ambang batas yang ditentukan (0.8). Fitur-fitur yang memiliki korelasi tinggi ditambahkan ke dalam sebuah set bernama `col_corr` untuk dihapus nanti.

Selanjutnya, kolom target 'income' dihapus terlebih dahulu dari dataset agar tidak ikut dalam perhitungan korelasi. Fitur-fitur dengan korelasi tinggi yang terdeteksi, seperti 'age_group', 'education_level', dan 'work_hours_ratio', kemudian dihapus dari dataset menggunakan metode `df.drop()`. Hal ini dilakukan untuk mengurangi redundansi fitur, mengatasi multikolinearitas, dan meningkatkan performa model analisis atau machine learning.

6.7. Pengecekan Ulang Korelasi



```
# Correlation Heatmap
correlation = df.corr()
plt.subplots(figsize=(15,15))
sns.heatmap(correlation.round(2),
            annot=True,
            vmax=1,
            square=True,
            cmap='RdYlGn_r')
plt.show()
```

Terlihat pada gambar tidak ada korelasi yang tinggi, sehingga data sudah bersih.

UNIT 7 : MEMBANGUN MODEL

Tujuan dari membangun model adalah untuk membuat dan melatih model *machine learning* untuk memprediksi *income* berdasarkan dataset yang telah dipersiapkan.

7.1.Persiapan Data

```
[47] from sklearn.model_selection import train_test_split
# pisahkan fitur dan target
x = df.drop('income', axis=1)
y = df['income']

# pisahkan data menjadi data latih dan data uji
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=42, stratify=y)
```

Pada kode diatas kita memisahkan antara data *feature* dengan data target, kemudian variabel yang telah dipisah itu dimasukkan ke dalam *train test* yang dimana mengatur *test_size* bernilai 20% (0,2) dan memiliki *random_state* = 42.

7.2.Membangun Dengan Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_report, confusion_matrix

# inisialisasi model
dt_model = DecisionTreeClassifier(random_state=42)

# melatih model
dt_model.fit(x_train, y_train)

# melakukan prediksi
y_pred = dt_model.predict(x_test)

# evaluasi
print("Decision Tree Performance")
print("Akurasi:", accuracy_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred, average="micro"))
print("Precision:", precision_score(y_test, y_pred, average="micro"))
```

Disini lakukan import library yang dibutuhkan yaitu *DecisionTreeClassifier* dari modul *sklear.tree* dan juga penghitungg metrics dari *sklearn.metrics*, kemudian melakukan inisialisasi model dengan *random_state=42*, dan melatih model dengan *x* training dan *y* training. Kemudian lakukan prediksi dengan variabel test yang sudah dibuat sebelumnya.

```
Decission Tree Performance
Akurasi: 0.7782220855121501
Recall: 0.7782220855121501
Precision: 0.7782220855121501
```

```
Classification Report:
              precision    recall  f1-score   support

     0       0.54         0.54         0.54        1567
     1       0.85         0.85         0.85        4935

 accuracy          0.78         0.78         0.78        6502
 macro avg         0.70         0.70         0.70        6502
 weighted avg      0.78         0.78         0.78        6502
```

Disini didapatkan report performance dan klasifikasinya, kemudian didapatkan data sebagai berikut :

Performa Decision Tree :

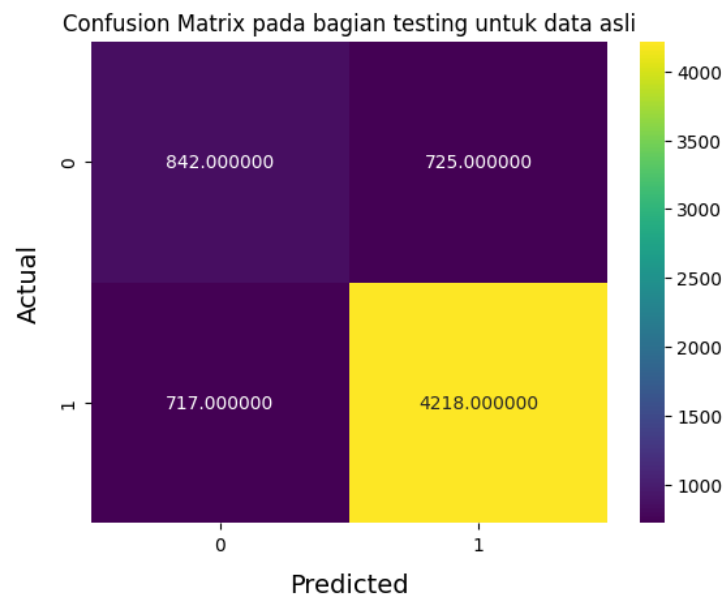
- Akurasi : 0.7782220855121501 (77,82%)
- Recall : 0.7782220855121501 (77,82%)
- Precision : 0.7782220855121501 (77,82%)

Laporan Klasifikasi :

Berapapun hasil accuracy, precission, recall selama nilainya di range yang sama (yaitu **0.7782220855121501**), menunjukkan bahwa preprocessing yang kita gunakan sudah benar dan sesuai.

7.2.1.Membuat Heatmap

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap="viridis", fmt="0f")
plt.xlabel("Predicted", fontdict={'size':14}, labelpad=10)
plt.ylabel("Actual", fontdict={'size':14}, labelpad=10)
plt.title("Confusion Matrix pada bagian testing untuk data asli")
plt.show()
```



Komponen – komponennya :

a) Sumbu X : Nilai Prediksi

- 0 : Model memprediksi income \leq 50K
- 1 : Model memprediksi income $>$ 50K

b) Sumbu Y : Nilai sebenarnya

- 0 : Model memprediksi income \leq 50K

- 1 : Model memprediksi $income > 50K$

c) Nilai dalam setiap sel :

- Kiri atas (842) : True negatives (TN) – Benar diprediksi $\leq 50K$
- Kanan Atas (725): False Positives (FP) – Salah diprediksi $> 50K$
- Kiri Bawah (717): False Negatives (FN) – Salah diprediksi $\leq 50K$
- Kanan Bawah (4218): True Positives (TP) – Benar diprediksi $> 50K$

d) Interpretasi :

- Akurasi : Model benar dalam $(4218/842) / (842+725+717+4218) = 0.778222086 = 77,8\%$ kasus.
- Presisi untuk $> 50K$: $4218 / (725 + 4218) = 0.853327938 = 85,3\%$
- Recall untuk $> 50K$: $4218 / (717 + 4218) = 0.854711246 = 85,4\%$
- Model cenderung lebih baik dalam memprediksi $income > 50K$ (4218 benar vs 725 salah)
- Ada jumlah signifikan false positives (725) dan false negatives (717), menunjukkan ada ruang untuk peningkatan.

e) Kesimpulan :

- Model memiliki performa cukup baik dalam mengidentifikasi $income > 50K$. Namun, model mengalami kesulitan dalam memprediksi $income \leq 50K$ dengan akurat.
- Ada keseimbangan yang cukup baik antara *false positives* dan *false negatives*, menunjukkan model tidak terlalu bias ke salah satu kelas.
- Untuk meningkatkan model, fokus mungkin perlu diberikan pada fitur-fitur yang lebih baik membedakan

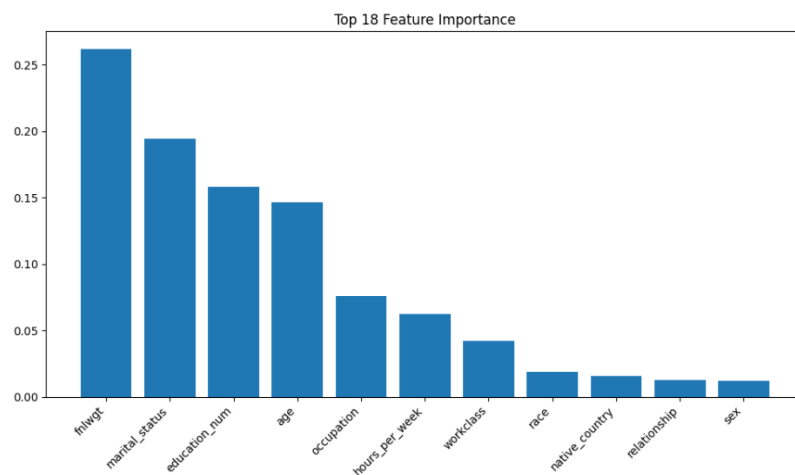
antara 2 kategori *income* terutama untuk kasus *income* $\leq 50K$

7.3.Feature Importance

```
[50] feature_importance = pd.DataFrame({'feature': x.columns, 'importance': dt_model.feature_importances_})
      feature_importance = feature_importance.sort_values(by='importance', ascending=False)

      plt.figure(figsize=(10, 6))
      plt.bar(feature_importance['feature'][:18], feature_importance['importance'][:18])
      plt.title('Top 18 Feature Importance')
      plt.xticks(rotation=45, ha='right')
      plt.tight_layout()
      plt.show()
```

Pada *feature_importance* kita memanggil *dataframe* *pandas* yang dimana *feature* adalah *x_columns* dan *importance* adalah dari *dt model*. Kemudian kita melakukan pembuatan grafik yang diurutkan berdasarkan *importance*.



Disini kita melihat variabel apa yang paling mempengaruhi keputusan dari grafik yang didapat yaitu ada *fnlwgt*(*final weight*) yang paling mempengaruhi keputusan.

7.4.Hyperparameter Tuning

Hyperparameter adalah parameter yang tidak dipelajari oleh model selama proses pelatihan, melainkan diatur sebelum proses pelatihan dimulai

Hyperparameter tuning adalah proses mencari nilai optimal dari *hyperparameter* suatu model *machine learning* untuk meningkatkan kinerja model tersebut

```

] from sklearn.model_selection import GridSearchCV

# Definisikan parameter yang akan diuji
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [5,10,15,20],
    'min_samples_split': [2,5,10],
    'min_samples_leaf': [1,2,4]
}

# inisialisasi GridSearchCV
grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')

# lakukan pencarian
grid_search.fit(x_train, y_train)

# ambil model terbaik
best_model = grid_search.best_estimator_

# tampilkan parameter terbaik
print("Parameter terbaik untuk Decission Tree:")
print(grid_search.best_params_)

print('Best Cross-Validation score: ', grid_search.best_score_)

```

Pertama melakukan *import* terhadap library yang dibutuhkan, kemudian defenisikan parameter yang akan diuji dengan membuat variabel `param_grid`, dan inisialisasi `gridSearchCV`, kemudian lakukan pencarian dan ambil model terbaiknya setelah itu tampilkan parameter terbaiknya.

```

Parameter terbaik untuk Decission Tree:
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10}
Best Cross-Validation score: 0.8223103592372478

```

Kemudian didapatkan hasil seperti gambar diatas dengan data sebagai berikut :

- Parameter terbaik untuk Decission Tree:
 1. 'criterion': 'entropy',
 2. 'max_depth': 10,
 3. 'min_samples_leaf': 4,
 4. 'min_samples_split': 10
- Best Cross-Validation score: **0.8223103592372478**

7.4.1. Menggunakan Model Terbaik Untuk Prediksi Data

```

# Gunakan model terbaik
best_dt_model = grid_search.best_estimator_

# prediksi degan model terbaik
y_pred_best_dt = best_dt_model.predict(x_test)

```

disini kita menggunakan best model untuk melakukan prediksi

```

# Evaluasi model terbaik
print("\nBest Decission Tree Performance")
print("Akurasi:", accuracy_score(y_test, y_pred_best_dt))
print("Recall:", recall_score(y_test, y_pred_best_dt, average="micro"))
print("Precision:", precision_score(y_test, y_pred_best_dt, average="micro"))
print("\nClassification Report:\n", classification_report(y_test, y_pred_best_dt))

```

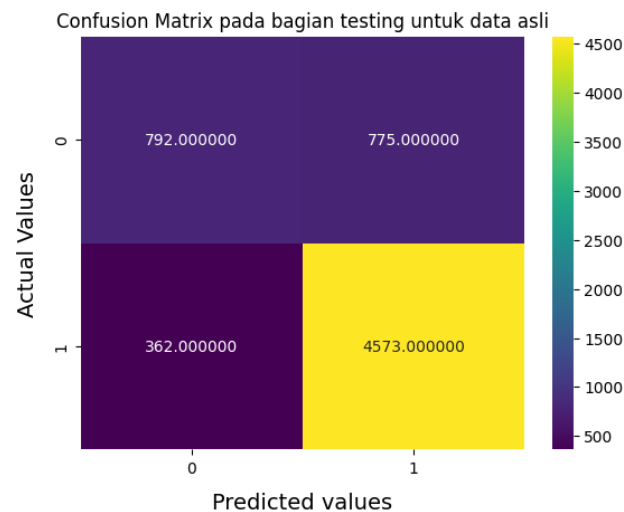
Kemudian lakukan pengecekan performa decision tree dengan best model, dan didapatkan data sebagai berikut :

Best Decision Tree Performance

- Akurasi: 0.8251307290064596 (82,51%)
- Recall: 0.8251307290064596 (82,51%)
- Precision: 0.8251307290064596 (82,51%)

7.4.2. Membuat Heatmap

```
sns.heatmap(confusion_matrix(y_test, y_pred_best_dt), annot=True, cmap="viridis", fmt="0f")
plt.xlabel("Predicted values", fontdict={'size':14}, labelpad=10)
plt.ylabel('Actual Values', fontdict={'size':14}, labelpad=10)
plt.title("Confusion Matrix pada bagian testing untuk data asli")
plt.show()
```



Analisis : Komponen Confusion Matrix :

- True Negatives (TN) : 792 (Kiri Atas)
- False Positives (FP) : 775 (Kanan Atas)
- False Negatives (FN) : 362 (Kiri Bawah)
- True Positives (TP) : 4573 (Kanan Bawah)

Interpretasi :

- Akurasi : $(792 + 4573)/(792+775+362+4573) = 0.825130729 = 82.5\%$
- Presisi untuk > 50K : $4573/(775+4573) = 0.855086013 = 85.5\%$
- Recall untuk >50K : $4573/(362+4573) = 0.926646403 = 92.6\%$

Perbedaan dengan gambar sebelumnya :

- TN menurun dari yang sebelumnya 842 vs 792,
- FP meningkat dari yang sebelumnya 725 vs 775,

- FN menurun dari 717 vs 362,
- TP Meningkat pesat dari 4218 vs 4573

Kesimpulan :

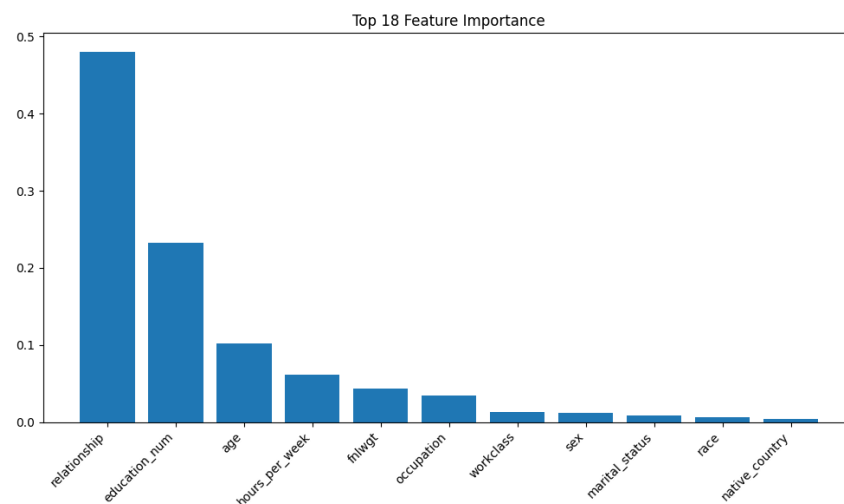
- Model ini memiliki performa yang lebih baik dibandingkan dengan model yang sebelumnya
- akurasi meningkat dari 77,7% menjadi 82,5%
- Presisi untuk kelas > 50K juga meningkat
- Model ini lebih baik dalam mengurangi False Negatives

7.5.Feature Importance Setelah Hyperparameter Tuning

```
feature_importance = pd.DataFrame({'feature':x.columns, 'importance': best_dt_model.feature_importances_})
feature_importance = feature_importance.sort_values(by='importance', ascending=False)

plt.figure(figsize=(10,6))
plt.bar(feature_importance['feature'][:18], feature_importance['importance'][:18])
plt.title('Top 18 Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Proses ini sama seperti sebelumnya, tetapi kita mengambil data dari hasil prediksi best model.



Didapatkan data diatas, yaitu sekarang yang akan mempengaruhi keputusan adalah *relationship* dan bukan *finalweight* lagi.

7.6.Simpan Model Terbaik

```
import joblib

joblib.dump(best_dt_model, f'{folder_name}/best_income_predictor_model.joblib')
print("Model terbaik telah disimpan sebagai 'best_income_predictor_model.joblib'")
```

Model terbaik telah disimpan sebagai 'best_income_predictor_model.joblib'

Lakukan penyimpanan model terbaik dengan ekstensi `joblib` yang didapat dari library `joblib` yang di *import*.

7.7.Cek Underfitting Dan Overfitting Untuk Hyperparameter

7.7.1. Underfitting

Terjadi ketika model terlalu sederhana untuk menangkap pola yang kompleks dalam data. Model ini tidak mampu memetakan hubungan yang sebenarnya antara fitur dengan target variabel

Ciri – ciri :

- Akurasi rendah baik pada data pelatihan maupun data uji
- Model terlalu umum dan tidak spesifik untuk data yang sedang dipelajari
- Bias tinggi

Penyebab :

- Model yang terlalu sederhana (misalkan jumlah pohon keputusan yang terlalu sedikit)
- Data pelatihan terlalu sedikit atau tidak representatif
- Hyperparameter yang terlalu membatasi kompleksitas model

Contoh : anda mencoba melatih model untuk memprediksi harga jumlah rumah berdasarkan luas dan lokasi. Namun, model hanya mempertimbangkan luas saja, sehingga tidak dapat menangkap pengaruh lokasi terhadap harga.

7.7.2. Overfitting

Terjadi ketika model terlalu kompleks dan menghafal noise atau variasi acak dalam data pelatihan. Akibatnya, model menjadi terlalu spesifik untuk data pelatihan dan tidak dapat generalisasi dengan baik pada data baru.

Ciri-ciri :

- Akurasi tinggi pada data pelatihan, tetapi rendah pada data uji.
- Model terlalu kompleks dan rentan terhadap noise
- Variasi tinggi

Penyebab :

- Model yang terlalu kompleks (misalkan, jumlah pohon keputusan yang terlalu banyak)
- Data pelatihan yang terlalu sedikit untuk mengandung noise
- Hyperparameter yang terlalu memungkinkan kompleksitas model

Contoh : Anda melatih model untuk mengenali tulisan tangan. Model berhasil mengenali tulisan tangan dari penulis tertentu dalam data pelatihan, tetapi model gagal dalam mengenali tulisan tangan dari penulis lain.

7.7.3. Hubungan Dengan Hyperparameter Tuning

Hyperparameter Tuning bertujuan untuk menemukan nilai optimal dari hyperparameter agar model dapat generalisasi dengan baik pada data baru. Jika nilai *Hyperparameter* terlalu kecil, model akan mengalami *UNDERFITTING*. sebaliknya, jika nilai *Hyperparameter* terlalu besar, maka model akan mengalami *OVERFITTING*.

Tujuan utama dari *hyperparameter tuning* adalah :

- Mencegah overfitting: Dengan memilih hyperparameter yang tepat, kita dapat membatasi kompleksitas model dan mencegahnya menghafal noise dalam data.
- Mencegah underfitting: Dengan memilih hyperparameter yang cukup kompleks, kita dapat memastikan model mampu menangkap pola yang kompleks dalam data

7.8. Membandingkan Akurasi Training vs Testing

```
from sklearn.metrics import accuracy_score

# untuk model Decision Tree terbaik
y_train_pred = best_dt_model.predict(x_train)
y_test_pred = best_dt_model.predict(x_test)

# Hitung akurasi pelatihan dan pengujian
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print("Akurasi pelatihan:", train_accuracy)
print("Akurasi pengujian:", test_accuracy)
```

Akurasi pelatihan: 0.8449204029839268
 Akurasi pengujian: 0.8251307290064596

Disini kita melakukan import `accuracy_score` dan kemudian lakukan prediksi baik dengan data *training* maupun data *testing* dan hitung akurasi dari pelatihan dan pengujian.

Akurasi pelatihan: 0.8449204029839268

Akurasi pengujian: 0.8251307290064596

Dan dapatlah hasil akurasi sebagai berikut :

- Akurasi Pelatihan : 0.8449204029839268 (84,49%)
- Akurasi Pengujian : 0.8251307290064596 (82,51%)

Interpretasi :

- jika *train_accuracy* jauh lebih tinggi dari *test_accuracy*, ini indikasi overfitting,
- jika keduanya rendah dan hampir sama, maka ini bisa jadi indikasi underfitting

7.9.Learning Curve

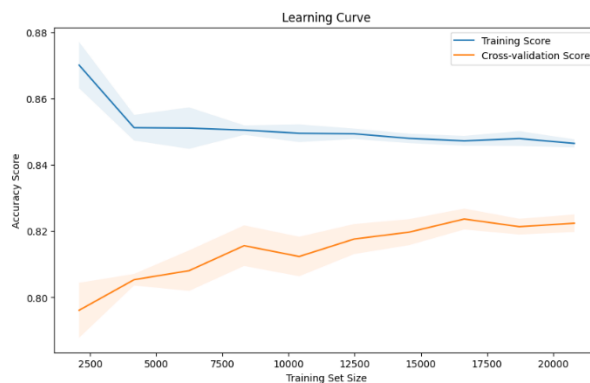
```
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt

train_sizes, train_scores = learning_curve(
    best_dt_model, x_train, y_train, cv=5, train_sizes=np.linspace(0.1,1.0,10), n_jobs=-1)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.figure(figsize=(10,6))
plt.plot(train_sizes, train_mean, label='Training Score')
plt.plot(train_sizes, test_mean, label='Cross-validation Score')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.1)
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.1)
plt.xlabel('Training Set Size')
plt.ylabel('Accuracy Score')
plt.title('Learning Curve')
plt.legend()
plt.show()
```

Bagian kode ini kita akan menampilkan learning curve terhadap model yang kita buat.



Interpretasi :

- Jika kurva training terus naik tetapi kurva testing mendatar atau turun, ini indikasi *overfitting*.
- Jika kedua kurva rendah dan berdekatan, ini indikasi *underfitting*

7.10. Cross-Validation Score

```
from sklearn.model_selection import cross_val_score

# Hitung cross-validation score
cv_scores = cross_val_score(best_dt_model, x_train, y_train, cv=5)
print('Cross-validation Score : ', cv_scores)
print('Mean Cross-validation Score : ', np.mean(cv_scores))
print('Standard deviation of CV Score : ', np.std(cv_scores))
```

Pada kode diatas kita akan melakukan perhitungan `cross_validation_score` yang di import dari modul `sklearn.model_selection`.

```
Cross-validation Score : [0.81910804 0.82580273 0.82234186 0.82003461 0.82426456]
Mean Cross-validation Score : 0.8223103592372478
Standard deviation of CV Score : 0.002508006544471124
```

Kemudian dapatlah hasil cross-validation seperti ini :

- Cross-validation Score:[0.81910804, 0.82580273, 0.82234186, 0.82003461, 0.82426456]
- Mean Cross-validation Score : 0.8223103592372478
- Standard deviation of CV Score : 0.002508006544471124

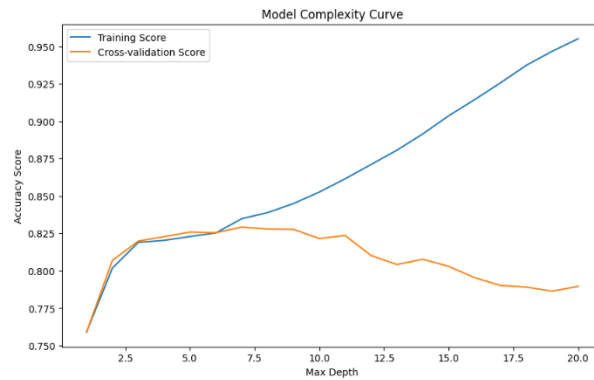
7.11. Complexity Curve

```
max_depths = range(1,21) #digunakan untuk nilai integer dari 1 - 20
train_scores = []
test_scores = []

for max_depth in max_depths:
    dt = DecisionTreeClassifier(max_depth=max_depth, random_state=42)
    dt.fit(x_train, y_train)
    train_scores.append(dt.score(x_train, y_train))
    test_scores.append(dt.score(x_test, y_test))

plt.figure(figsize=(10,6))
plt.plot(max_depths, train_scores, label='Training Score')
plt.plot(max_depths, test_scores, label='Cross-validation Score')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy Score')
plt.title('Model Complexity Curve')
plt.legend()
plt.show()
```

Pada kode ini kita akan menampilkan *complexity curve* untuk *decision tree*. Menggunakan grafik plot.



Interpretasi:

- Jika kurva training terus naik tapi kurva testing mulai turun setelah titik tertentu, ini indikasi overfitting.
- Jika kedua kurva rendah dan berdekatan, ini indikasi underfitting.

Observasi:

- Underfitting: Terlihat di bagian kiri grafik (kedalaman 1-3) dimana kedua skor rendah.
- Overfitting: Mulai terjadi setelah kedalaman sekitar 7, dimana garis biru terus naik tapi garis oranye mulai turun.
- Sweet spot (titik optimal): Berada di sekitar kedalaman 5-7, dimana testing score mencapai puncak.

Kesimpulan:

- Model dengan kedalaman sekitar 5-7 mungkin memberikan keseimbangan terbaik antara bias dan varians.
- Setelah kedalaman 7, model mulai overfitting: performa pada data training terus meningkat, tapi menurun pada data testing.
- Kedalaman pohon di atas 10 menunjukkan overfitting yang signifikan, dengan gap besar antara training dan testing score.

Rekomendasi:

- Gunakan max_depth antara 5-7 untuk model final untuk menghindari overfitting.

7.12. Visualisasi Decision Tree dengan max_depth=5

```
[61] from sklearn.tree import plot_tree

# buat gambar dengan ukuran yang sangat besar - disini maxdepth dibuat 5 agar lebih cepat terbuat
plt.figure(figsize=(100,50))

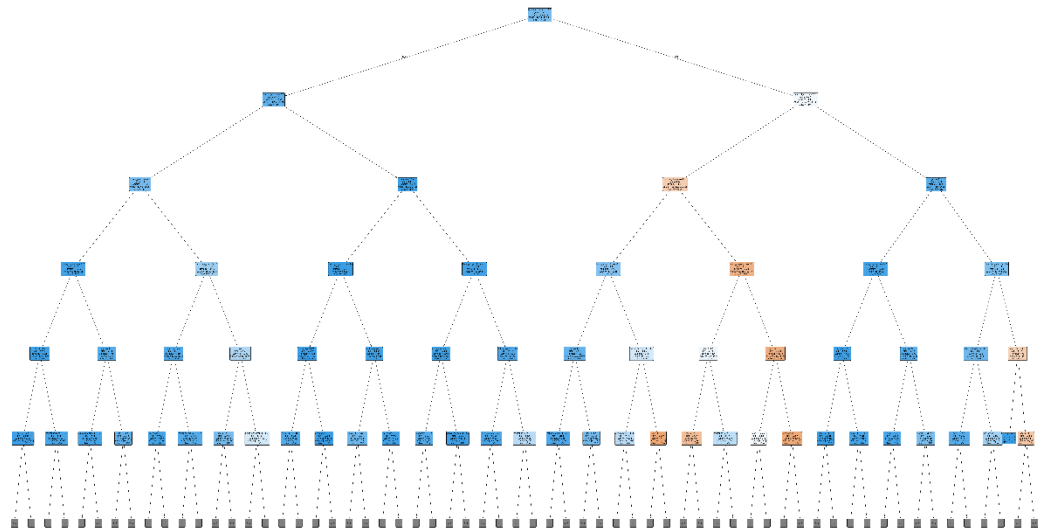
# plot untuk pohon keputusan
plot_tree(dt_model, feature_names=x.columns, class_names=['<=50K', '>50K'], filled=True, rounded=True, fontsize=10, max_depth=5)

# simpan gambar dengan dpi tinggi
plt.savefig(f'{folder_name}/decision_tree_visualization.png', dpi=300, bbox_inches='tight')

print('Gambar pohon keputusan telah disimpan sebagai "decision_tree_visualization.png")
```

Pada kode ini kita akan melakukan visualisasi decision tree dari best model yang telah dibuat diatas.

Dan ini adalah hasil dari visualisasi treenya.



7.13. Visualisasi Decision Tree versi Hyperparameter Tanpa max_depth

```
from sklearn.tree import plot_tree

# buat gambar dengan ukuran yang sangat besar - disini maxdepth dibuat 5 agar lebih cepat terbuat
plt.figure(figsize=(100,50))

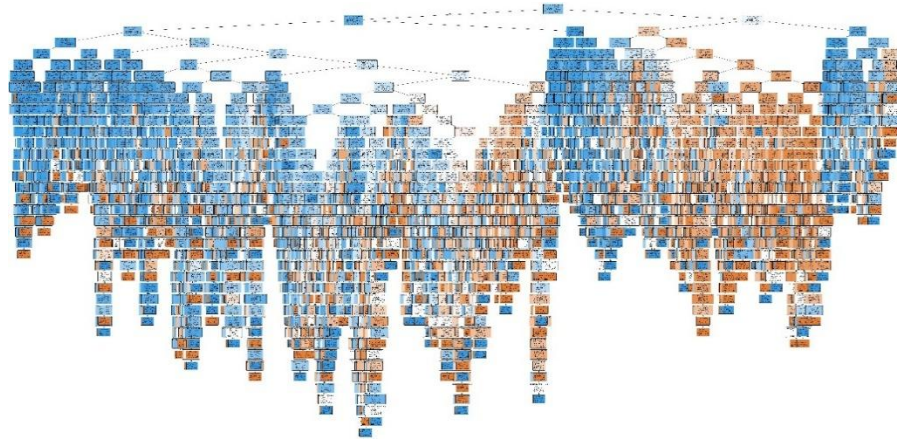
# plot untuk pohon keputusan
plot_tree(dt_model, feature_names=x.columns, class_names=['<=50K', '>50K'], filled=True, rounded=True, fontsize=10)

# simpan gambar dengan dpi tinggi
plt.savefig(f'{folder_name}/best_decision_tree_visualization.png', dpi=300, bbox_inches='tight')

print('Gambar pohon keputusan telah disimpan sebagai "best_decision_tree_visualization.png")
```

Sama seperti sebelumnya, pada tahap ini menampilkan data visualisasi *decision tree* tanpa *max depth*.

Berikut adalah hasil dari visualisasi treenya:



DEPLOYMENT MODEL

1. Load Drive

```
folder_name = "/content/drive/My Drive/Irfan/Kuliah/Semester 5/Machine Learning/Pertemuan 03 - (15-10-24)/01_adult_classification/"
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

import sys
sys.path.append(f'{folder_name}')
```

Mounted at /content/drive

Pertama kita perlu melakukan import drive nya terlebih dahulu

2. Import Library Yang Dibutuhkan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
pd.set_option('display.max_columns', None)
```

Melakukan import terhadap library yang dibutuhkan pada jupyter collab

3. Membaca Metode Terbaik Yang Telah Disimpan

```
classifier_dt=joblib.load(f'{folder_name}/best_income_predictor_model.joblib')
classifier_dt
```

DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10, min_samples_leaf=4, min_samples_split=10, random_state=42)

Kita membaca file dengan ekstensi joblib yang telah di export sebelumnya.

4. Mendeklarasikan Informasi Yang Dibutuhkan Untuk Melakukan Prediksi

informasi yang dipakai

1. age float64
2. workclass int64
3. fnlwgt float64
4. education-num float64
5. marital-status int64
6. occupation int64
7. relationship int64
8. race int64
9. sex int64
10. hours-per-week float64
11. native-country int64
12. income

5. Meminta Pengguna Memasukkan Informasi Yang Dibutuhkan

▼ Masukkan age

```
[ ] age = float(input('Masukkan Age: '))
```

↳ Masukkan Age: 50

▼ Masukkan WorkClass

```
workclass = {'State-gov': 7, 'Self-emp-not-inc': 6, 'Private': 5, 'Federal-gov': 4, 'Local-gov': 3, 'Self-emp-inc': 2, 'Without-pay': 1, 'Never-worked': 0}
print(workclass)
```

↳ {'State-gov': 7, 'Self-emp-not-inc': 6, 'Private': 5, 'Federal-gov': 4, 'Local-gov': 3, 'Self-emp-inc': 2, 'Without-pay': 1, 'Never-worked': 0}

```
[ ] workclass = int(input('Masukkan workclass: '))
```

↳ Masukkan workclass: 7

▼ Masukkan Final Weight

```
[ ] fnlwgt = float(input('Masukkan Final Weight: '))
```

↳ Masukkan Final weight: 100

▼ Masukkan Education

```
education = {'Preschool': 1.0, '1st-4th': 2.0, '5th-6th': 3.0, '7th-8th': 4.0, '9th': 5.0, '10th': 6.0, '11th': 7.0, '12th': 8.0, 'HS-grad': 9.0, 'Some-college': 10.0, 'Assoc-voc': 11.0, 'Assoc-acdm': 12.0, 'Bachelors': 13.0, 'Masters': 14.0, 'Doctorate': 15.0}
print(education)
```

↳ {'Preschool': 1.0, '1st-4th': 2.0, '5th-6th': 3.0, '7th-8th': 4.0, '9th': 5.0, '10th': 6.0, '11th': 7.0, '12th': 8.0, 'HS-grad': 9.0, 'Some-college': 10.0, 'Assoc-voc': 11.0, 'Assoc-acdm': 12.0, 'Bachelors': 13.0, 'Masters': 14.0, 'Doctorate': 15.0}

```
[ ] education_num = float(input('Masukkan education_num: '))
```

↳ Masukkan education_num: 7.0

▼ Masukkan Marital Status

```
[ ] marital_status_info = {  
    'Never-married': 6,  
    'Married-civ-spouse': 5,  
    'Divorced': 4,  
    'Married-spouse-absent': 3,  
    'Separated': 2,  
    'Married-AF-spouse': 1,  
    'Widowed': 0  
}  
print(marital_status_info)
```

```
{'Never-married': 6, 'Married-civ-spouse': 5, 'Divorced': 4, 'Married-spouse-absent': 3, 'Separated': 2, 'Married-AF-spouse': 1, 'Widowed': 0}
```

```
[ ] marital_status = int(input('Masukkan angka marital_status: '))
```

```
Masukkan angka marital_status: 4
```

▼ Masukkan Occupation

```
● occupation = {  
    'Adm-clerical': 13,  
    'Exec-managerial': 12,  
    'Handlers-cleaners': 11,  
    'Prof-specialty': 10,  
    'Other-service': 9,  
    'Sales': 8,  
    'Craft-repair': 7,  
    'Transport-moving': 6,  
    'Farming-fishing': 5,  
    'Machine-op-inspct': 4,  
    'Tech-support': 3,  
    'Protective-serv': 2,  
    'Armed-Forces': 1,  
    'Priv-house-serv': 0  
}  
print(occupation)
```

```
{'Adm-clerical': 13, 'Exec-managerial': 12, 'Handlers-cleaners': 11, 'Prof-specialty': 10, 'Other-service': 9, 'Sales': 8, 'Craft-repair': 7, 'Transport-moving': 6, 'Farming-fishing': 5, 'Machine-op-inspct': 4, 'Tech-sup  
< >
```

```
[ ] occupation = int(input('Masukkan occupation: '))
```

```
Masukkan occupation: 11
```

▼ Masukkan Relationship

```
▶ relationship = {  
    'Not-in-family': 5,  
    'Husband': 4,  
    'Wife': 3,  
    'Own-child': 2,  
    'Unmarried': 1,  
    'Other-relative': 0  
}  
print(relationship)
```

```
{'Not-in-family': 5, 'Husband': 4, 'Wife': 3, 'Own-child': 2, 'Unmarried': 1, 'Other-relative': 0}
```

```
[ ] relationship=int(input('Masukkan angka relationship: '))
```

```
Masukkan angka relationship: 4
```

▼ Masukkan Race

```
race = {  
    'White': 4,  
    'Black': 3,  
    'Asian-Pac-Islander': 2,  
    'Amer-Indian-Eskimo': 1,  
    'Other': 0  
}  
print(race)
```

↔ {'White': 4, 'Black': 3, 'Asian-Pac-Islander': 2, 'Amer-Indian-Eskimo': 1, 'Other': 0}

```
[ ] race = int(input('Masukkan angka race: '))
```

↔ Masukkan angka race: 3

▼ Masukkan Sex

```
[ ] sex = {'Female' : 0, 'Male' : 1}  
print(sex)
```

↔ {'Female': 0, 'Male': 1}

```
[ ] sex = int(input('Masukkan angka sex: '))
```

↔ Masukkan angka sex: 1

✓ Masukkan Hours Per Week

```
hours_per_week = float(input('Masukkan hours_per_we'))
```

Masukkan hours_per_week: 90

✓ Masukkan Native Country

```
[ ] native_country = {  
    'United-States': 40,  
    'Cuba': 39,  
    'Jamaica': 38,  
    'India': 37,  
    'Mexico': 36,  
    'South': 35,  
    'Puerto-Rico': 34,  
    'Honduras': 33,  
    'England': 32,  
    'Canada': 31,  
    'Germany': 30,  
    'Iran': 29,  
    'Philippines': 28,  
    'Italy': 27,  
    'Poland': 26,  
    'Columbia': 25,  
    'Cambodia': 24,  
    'Thailand': 23,  
    'Ecuador': 22,  
    'Laos': 21,  
    'Taiwan': 20,  
    'Haiti': 19,  
    'Portugal': 18,  
    'Dominican-Republic': 17,  
    'El-Salvador': 16,  
    'France': 15,  
    'Guatemala': 14,  
    'China': 13,  
    'Japan': 12,  
    'Yugoslavia': 11,  
    'Peru': 10,  
    'Outlying-US(Guam-USVI-etc)': 9,  
    'Scotland': 8,  
    'TrinidadTobago': 7,  
    'Greece': 6,  
    'Nicaragua': 5,  
    'Vietnam': 4,  
    'Hong': 3,  
    'Ireland': 2,  
    'Hungary': 1,  
    'Holand-Netherlands': 0  
}
```

```
print(native_country)
```

{'United-States': 40, 'Cuba': 39, 'Jamaica': 38, 'India': 37, 'Mexico': 36, 'South': 35, 'Puerto-Rico': 34, 'Honduras': 33, 'England': 32, 'Canada': 31, 'Germany': 30, 'Iran': 29, 'Philippines': 28, 'Italy': 27, 'Poland': 26, 'Columbia': 25, 'Cambodia': 24, 'Thailand': 23, 'Ecuador': 22, 'Laos': 21, 'Taiwan': 20, 'Haiti': 19, 'Portugal': 18, 'Dominican-Republic': 17, 'El-Salvador': 16, 'France': 15, 'Guatemala': 14, 'China': 13, 'Japan': 12, 'Yugoslavia': 11, 'Peru': 10, 'Outlying-US(Guam-USVI-etc)': 9, 'Scotland': 8, 'TrinidadTobago': 7, 'Greece': 6, 'Nicaragua': 5, 'Vietnam': 4, 'Hong': 3, 'Ireland': 2, 'Hungary': 1, 'Holand-Netherlands': 0}

```
[ ] native_country = int(input('Masukkan angka native_country: '))
```

Masukkan angka native_country: 34

Misalkan seperti pada gambar diatas kita memasukkan informasi yang akan diprediksi adalah:

- Age : 50
- Workclass : 7
- Final weight : 100
- Education : 7.0

- Marital_status : 4
- Occupation: 11
- Relationship : 4
- Race: 3
- Sex : 1
- Hours Per Week : 90
- Native Country : 34

6. Proses Prediksi Data Baru

```
[ ] pred_args = np.array([age, workclass, fnlwgt, education_num, marital_status, occupation, relationship, race, sex, hours_per_week, native_country]).reshape(1, -1)

pred_args
array([[ 50.,   7., 100.,   7.,   4.,  11.,   4.,   3.,   1.,  90.,  34.]])

[ ] model_prediction = classifier_dt.predict(pred_args)[0]
model_prediction
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
warnings.warn(
1
```

Membuat variabel `pred_args` untuk membuat array dengan membentuk data menjadi array NumPy dalam format matriks (1 sampel dengan banyak fitur), kode ini memastikan kompatibilitas dengan model machine learning yang memerlukan input dalam format terstruktur.

7. Hasil Prediksi

```
labels = {0: '<=50K', 1: '>50K'}
prediction = labels[model_prediction]
print('Hasil Prediksi adalah :', prediction)

Hasil Prediksi adalah : >50K
```

berfungsi untuk memetakan *output* model ke label kategori dan menampilkan hasil prediksi dengan cara yang mudah dipahami oleh pengguna. Dan mendapatkan hasil prediksinya >50K dari informasi yang kita isi sebelumnya.