

Introduction

Dalam praktikum ini, tujuan utama dari Data Scientist di Google Colab adalah untuk menerapkan teknik klasifikasi menggunakan pohon keputusan untuk memprediksi pendapatan masyarakat. Pendekatan ini melibatkan penggunaan berbagai atribut seperti usia, kategori kelas pekerja, status pernikahan, jenis kelamin, dan ras untuk mengidentifikasi pola yang terkait dengan pendapatan kurang dari atau sama dengan 50.000 USD atau lebih besar dari 50.000 USD.

Langkah pertama dalam mengembangkan model ini adalah membersihkan dan menyiapkan data untuk dianalisis. Sebuah pohon keputusan kemudian dibangun dengan menggunakan hiperparameter default. Proses selanjutnya termasuk memahami dan mengeksplorasi hiperparameter yang dapat disetel dalam pohon keputusan, serta memilih hiperparameter yang optimal dengan menggunakan teknik validasi silang pencarian grid.

Tujuan utama dari pendekatan Data Science dalam konteks ini ada dua. Pertama, identifikasi variabel-variabel kunci yang memiliki pengaruh signifikan terhadap pendapatan masyarakat. Kedua, terciptanya model prediksi yang akurat yang dapat memberikan wawasan berharga bagi bank. Dengan model yang solid, diharapkan dapat memberikan informasi untuk membantu bank dalam mengoptimalkan pendapatan masyarakat.

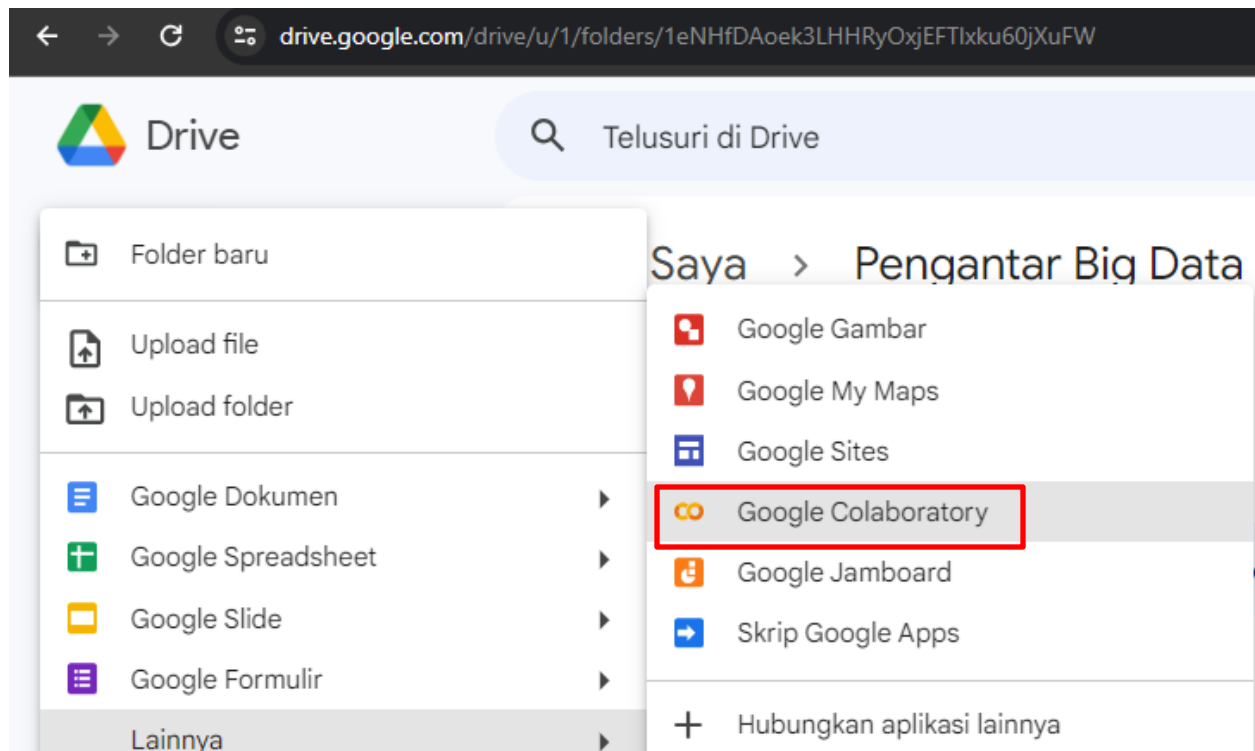
Dataset yang digunakan dalam kerja magang ini dapat diakses melalui tautan berikut:

https://drive.google.com/file/d/1x3eL_wKasCr0ROVU5CyTRUGwCvIkR_3Q/view?usp=drive_link

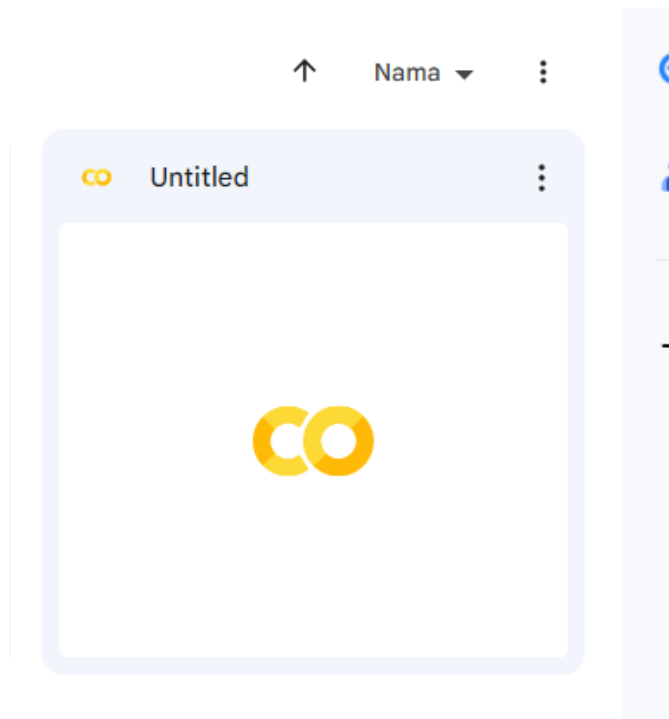
Dataset ini berperan penting dalam menggambarkan dan menggeneralisasi model yang relevan yang membantu membuat prediksi tentang pendapatan masyarakat.

Untuk mengidentifikasi hal-hal tersebut diatas akan digunakan Bahasa Pemrograman Python pada fasilitas yang tersedia pada google colab (<https://colab.research.google.com/>).

Untuk masuk ke google colab :



Dan secara otomatis pada google drive (<https://drive.google.com/>) muncul 1 folder baru : Colab Notebooks, folder ini dapat dipergunakan untuk meyimpan file script program, dataset dan file lainnya yang mendukung dalam proses identifikasi yang akan dilakukan.



Mengumpulkan Data

Sebelum melakukan proses investigasi terhadap permasalahan yang ada, langkah awal yang dapat dilakukan adalah melakukan pengumpulan data sesuai dengan permasalahan dan tujuan yang akan dicapai. Proses pengumpulan data dapat dilakukan dengan cara :

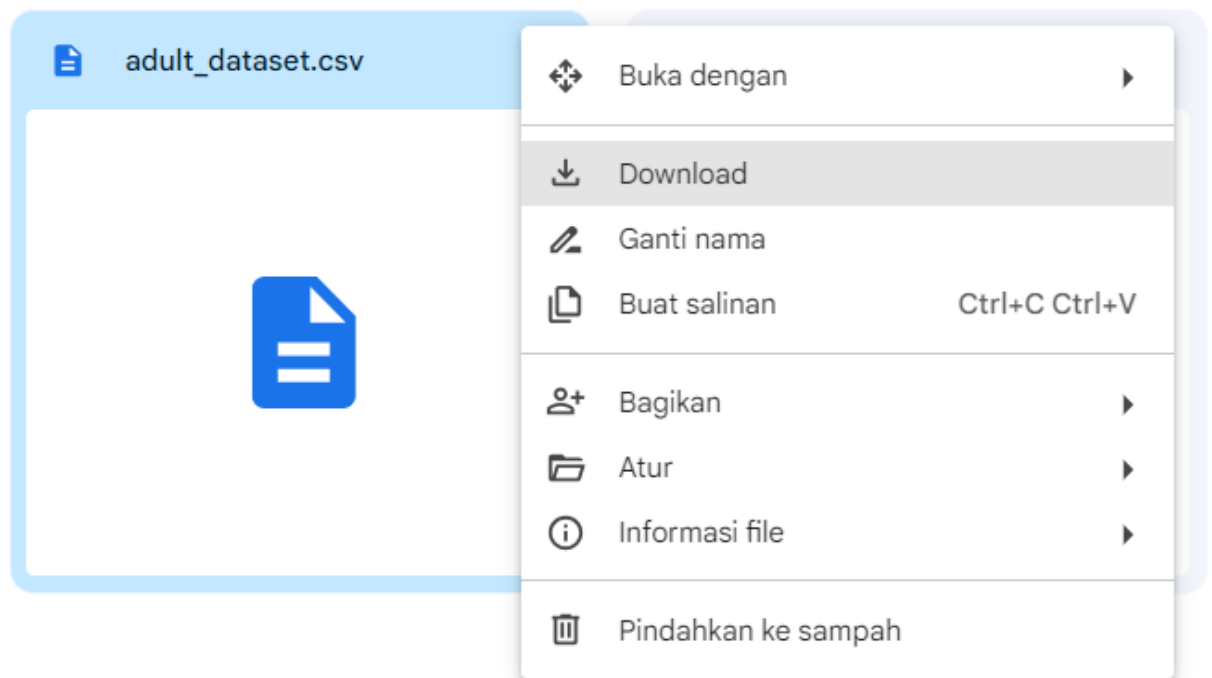
1. Wawancara
2. Observasi
3. Angket (Kuisisioner) / Survey
4. Studi Dokumen

Sesuai pada bagian Introduction diatas, sumber data akan diperoleh dari :

https://drive.google.com/file/d/1x3eL_wKasCr0ROVU5CyTRUGwCvIkR_3Q/view?usp=drive_link




Proses load data dilakukan dengan cara :

1. Download dataset sesuai link diatas :



2. Pindahkan dataset tersebut ke folder Colab Notebooks pada akun google drive yang telah terkoneksi dengan Google Colab sebelumnya.

Jenis ▾ Orang ▾ Dimodifikasi ▾

Nama	↑	Pemilik	Terakhi...	Ukuran file	⋮
 adult_dataset.csv		 saya	29 Nov 2023	3,4 MB	⋮

3. Untuk melakukan proses import data dari file dataset yang telah ada di dalam folder Colab Notebooks tersebut, dapat digunakan perintah berikut :

```
[68] import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import missingno

from sklearn.model_selection import train_test_split

from sklearn.decomposition import PCA

from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

from sklearn.metrics import confusion_matrix

[69] import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler

pd.set_option('display.max_columns', None) #melihat semua kolom

[71] from google.colab import drive
drive.mount('/content/gdrive')

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).

[72] import pandas as pd
dataset=pd.read_csv('gdrive/My Drive/Pengantar Big Data/adult_dataset.csv')

dataset #baris pertama dianggap nama feature padahal data pertama
```

Setelah ke 6 perintah tersebut diatas dijalankan pastikan status google drive telah Mounted (Mounted at /content/drive)

4. Sekarang, kita dapat memuat data dari file CSV di Google Drive dengan menggunakan kode berikut:

```
[72] import pandas as pd
      dataset=pd.read_csv('gdrive/My Drive/Pengantar Big Data/adult_dataset.csv')
```

5. Mari kita lihat beberapa baris pertama dari dataset untuk memastikan data sudah dimuat dengan benar.

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	Unnamed: 14
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States	<=50K
1	82	Private	132670	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
4	41	Private	284663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K
...
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	0	0	40	United-States	<=50K
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K

32561 rows x 15 columns

Dataset yang disediakan terdiri dari 32,561 baris dan 15 kolom. Setiap baris mewakili entri data yang mencakup berbagai informasi, sedangkan setiap kolom mencerminkan atribut atau fitur tertentu. Berikut adalah beberapa contoh data dari beberapa kolom:

Contoh Baris Pertama:

- Umur (age): 90
- Kelas Pekerja (workclass): ?
- Fnlwgt: 77053
- Pendidikan (education): HS-grad
- Pendidikan (education.num): 9
- Status Perkawinan (marital.status): Widowed
- Pekerjaan (occupation): ?
- Hubungan (relationship): Not-in-family
- Ras (race): White
- Jenis Kelamin (sex): Female
- Capital Gain: 0
- Capital Loss: 4356
- Jam Kerja per Minggu (hours.per.week): 40
- Negara Asal (native.country): United-States
- Kategori Penghasilan (income): <=50K

Contoh Baris Terakhir:

- Umur (age): 22
- Kelas Pekerja (workclass): Private
- Fnlwgt: 201490
- Pendidikan (education): HS-grad
- Pendidikan (education.num): 9
- Status Perkawinan (marital.status): Never-married
- Pekerjaan (occupation): Adm-clerical


- Hubungan (relationship): Own-child
- Ras (race): White
- Jenis Kelamin (sex): Male
- Capital Gain: 0
- Capital Loss: 0
- Jam Kerja per Minggu (hours.per.week): 20
- Negara Asal (native.country): United-States
- Kategori Penghasilan (income): <=50K

Dataset ini mencakup berbagai informasi demografis dan pekerjaan, serta kategori penghasilan sebagai target variabel. Harap diperhatikan bahwa beberapa kolom seperti "Occupation" dan "Native Country" memiliki nilai yang tidak diketahui (?).

6. Dengan memberi label ulang tersebut, setiap kolom sekarang memiliki nama yang lebih singkat dan dapat diidentifikasi dengan mudah. Berikut adalah keterangan singkat untuk setiap kolom baru:

A1: Umur (age)
 A2: Kelas Pekerja (workclass)
 A3: Fnlwgt
 A4: Pendidikan (education)
 A5: Jumlah Tahun Pendidikan (education.num)
 A6: Status Perkawinan (marital.status)
 A7: Pekerjaan (occupation)
 A8: Hubungan (relationship)
 A9: Ras (race)
 A10: Jenis Kelamin (sex)
 A11: Capital Gain
 A12: Capital Loss
 A13: Jam Kerja per Minggu (hours.per.week)
 A14: Negara Asal (native.country)
 Class: Kategori Penghasilan (income)

```
[76] dataset.columns=["A1", "A2", "A3", "A4", "A5", "A6", "A7", "A8", "A9", "A10",
.....: "A11", "A12", "A13", "A14", "Class"]
```

 dataset

Menelaah Data

1. Analisis karakteristik data pada dataset menggunakan metode `info()` memberikan pemahaman awal yang sangat berharga tentang struktur dan kualitas data. Berikut adalah beberapa aspek penting yang dapat diambil dari hasil `dataset.info()`:

Jumlah Data:

- Dataset ini terdiri dari 32,561 entri, mencakup sejumlah besar informasi yang dapat digunakan untuk analisis.

Jumlah Kolom:

- Terdapat 15 kolom dalam dataset, masing-masing merepresentasikan berbagai atribut atau fitur.

Nama Kolom dan Tipe Data:

- Setiap kolom memiliki nama unik dan tipe data tertentu. Misalnya, kolom "A1" memiliki tipe data integer (`int64`).

Jumlah Nilai Non-Null:

- Informasi ini menunjukkan bahwa tidak ada nilai null pada dataset untuk setiap kolom. Semua kolom memiliki 32,561 nilai non-null, menandakan integritas data yang baik.

Tipe Data:

- Tipe data yang beragam mencerminkan keberagaman informasi dalam dataset, mulai dari variabel kategoris hingga variabel numerik.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
#   Column  Non-Null Count  Dtype  
---  -
0    A1      32561 non-null  int64  
1    A2      32561 non-null  object  
2    A3      32561 non-null  int64  
3    A4      32561 non-null  object  
4    A5      32561 non-null  int64  
5    A6      32561 non-null  object  
6    A7      32561 non-null  object  
7    A8      32561 non-null  object  
8    A9      32561 non-null  object  
9    A10     32561 non-null  object  
10   A11     32561 non-null  int64  
11   A12     32561 non-null  int64  
12   A13     32561 non-null  int64  
13   A14     32561 non-null  object  
14   Class   32561 non-null  object  
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

2. Dalam kode ini, kita mengidentifikasi kolom-kolom kategoris dengan mengamati tipe data objek dan jumlah level unik kurang dari atau sama dengan 50. Setiap kolom kategoris yang terdeteksi kemudian dicetak bersama dengan nilai-nilai uniknya untuk memahami lebih lanjut karakteristiknya.

```
categorical_col = []
for column in dataset.columns:
    if dataset[column].dtype == object and len(dataset[column].unique()) <= 50: #mengecek level/isi data kolom (bisa object, int, dan tipe data lainnya)
        categorical_col.append(column)
        print(f"{column} : {dataset[column].unique()}")
        print("=====")

A2 : ['?' 'Private' 'State-gov' 'Federal-gov' 'Self-emp-not-inc' 'Self-emp-inc'
      'Local-gov' 'Without-pay' 'Never-worked']
=====
A4 : ['HS-grad' 'Some-college' '7th-8th' '10th' 'Doctorate' 'Prof-school'
      'Bachelors' 'Masters' '11th' 'Assoc-acdm' 'Assoc-voc' '1st-4th' '5th-6th'
      '12th' '9th' 'Preschool']
=====
A6 : ['Widowed' 'Divorced' 'Separated' 'Never-married' 'Married-civ-spouse'
      'Married-spouse-absent' 'Married-AF-spouse']
=====
A7 : ['?' 'Exec-managerial' 'Machine-op-inspct' 'Prof-specialty'
      'Other-service' 'Adm-clerical' 'Craft-repair' 'Transport-moving'
      'Handlers-cleaners' 'Sales' 'Farming-fishing' 'Tech-support'
      'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
=====
A8 : ['Not-in-family' 'Unmarried' 'Own-child' 'Other-relative' 'Husband' 'Wife']
=====
A9 : ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
=====
A10 : ['Female' 'Male']
=====
A14 : ['United-States' '?' 'Mexico' 'Greece' 'Vietnam' 'China' 'Taiwan' 'India'
       'Philippines' 'TrinidadTobago' 'Canada' 'South' 'Holand-Netherlands'
       'Puerto-Rico' 'Poland' 'Iran' 'England' 'Germany' 'Italy' 'Japan' 'Hong'
       'Monduras' 'Cuba' 'Ireland' 'Cambodia' 'Peru' 'Nicaragua'
       'Dominican-Republic' 'Haiti' 'El-Salvador' 'Hungary' 'Columbia'
       'Guatemala' 'Jamaica' 'Ecuador' 'France' 'Yugoslavia' 'Scotland'
       'Portugal' 'Laos' 'Thailand' 'Outlying-US(Guan-USVI-etc)']
=====
Class : ['<50K' '>50K']
=====
```

Analisis Kolom Kategoris:

A2 (Kelas Pekerja):

- "Nilai unik: ['?' 'Private' 'State-gov' 'Federal-gov' 'Self-emp-not-inc' 'Self-emp-inc' 'Local-gov' 'Without-pay' 'Never-worked']"
- "Temuan: Terdapat nilai '?' yang menunjukkan data yang tidak diketahui atau hilang."

A4 (Pendidikan):

- "Nilai unik: ['HS-grad' 'Some-college' ... 'Preschool']"
- "Temuan: Tidak terlihat masalah, nilai unik mencakup berbagai tingkat pendidikan."

A6 (Status Perkawinan):

- "Nilai unik: ['Widowed' 'Divorced' ... 'Married-AF-spouse']"
- "Temuan: Data status perkawinan tampak lengkap."

A7 (Pekerjaan):

- "Nilai unik: ['?' 'Exec-managerial' ... 'Priv-house-serv']"
- "Temuan: Terdapat nilai '?' yang menunjukkan data yang tidak diketahui atau hilang."

A8 (Hubungan):

- "Nilai unik: ['Not-in-family' 'Unmarried' ... 'Wife']"
- "Temuan: Data hubungan keluarga tampak lengkap."

A9 (Ras):

- "Nilai unik: ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']"
- "Temuan: Data ras tampak lengkap."

A10 (Jenis Kelamin):

- "Nilai unik: ['Female' 'Male']"
- "Temuan: Data jenis kelamin tampak lengkap."

A14 (Negara Asal):

- "Nilai unik: ['United-States' '?' 'Mexico' ... 'Outlying-US(Guam-USVI-etc)']"
- "Temuan: Terdapat nilai '?' yang menunjukkan data yang tidak diketahui atau hilang."

Class (Kategori Penghasilan):

- "Nilai unik: ['<=50K' '>50K']"
- "Temuan: Kategori penghasilan tampak lengkap."

Temuan Umum:

- Beberapa kolom mengandung nilai '?' yang menandakan keberadaan data yang tidak diketahui atau hilang. Perlu dipertimbangkan apakah nilai ini akan diatasi atau dihapus selama pemrosesan data selanjutnya.

3. Dengan menggunakan perintah `dataset['Class'].value_counts()`, kita dapat melihat distribusi kelas pada kolom 'Class'.

"Hasilnya menunjukkan bahwa terdapat dua kelas: '<=50K' dengan jumlah 24,720 entri dan '>50K' dengan jumlah 7,841 entri."

```
[80] dataset['Class'].value_counts()

<=50K    24720
>50K      7841
Name: Class, dtype: int64
```

4. Temuan: Kelas sudah ditemukan dan distribusinya dapat memberikan pemahaman awal tentang proporsi kelas dalam dataset.

```
Temuan:
• Kelas sudah ditemukan
```

5. Temuan: Kelas seimbang (balanced)

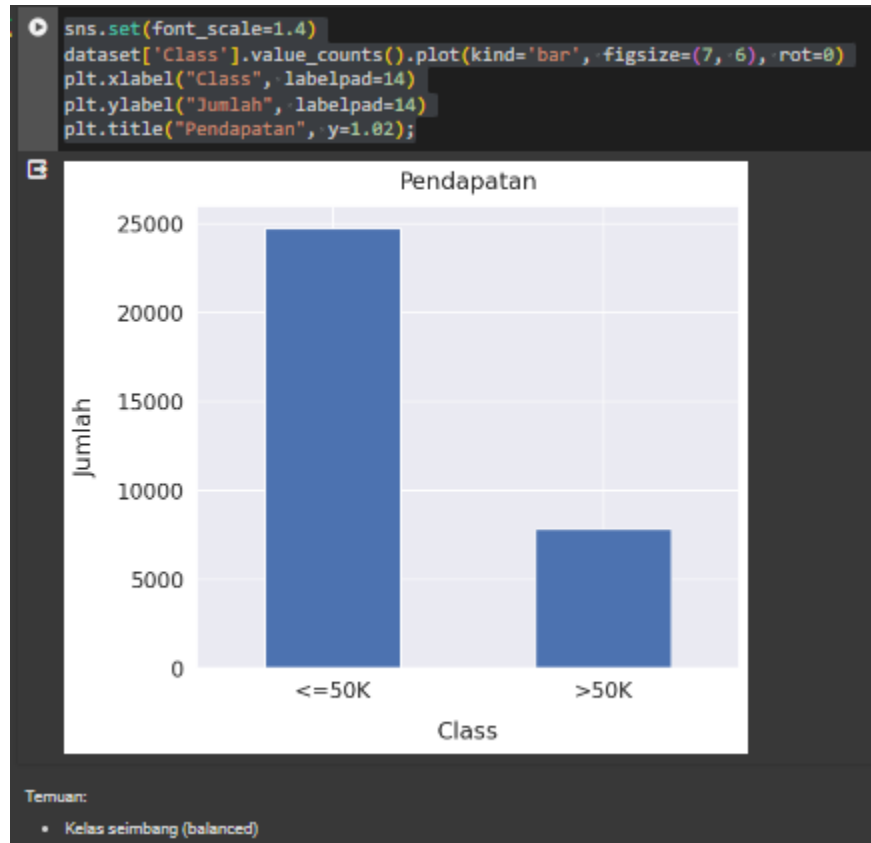
Visualisasi Distribusi Kelas:

- Menggunakan `sns.set(font_scale=1.4)`, kita setel skala font untuk visualisasi.
- Dengan memanfaatkan `dataset['Class'].value_counts().plot(kind='bar', figsize=(7, 6), rot=0)`, kita buat diagram batang untuk menampilkan distribusi kelas.
- Akses Label dan Judul: `plt.xlabel("Class", labelpad=14)`, `plt.ylabel("Jumlah", labelpad=14)`, dan `plt.title("Pendapatan", y=1.02)`.

- Visualisasi ini memberikan gambaran langsung tentang jumlah entri untuk setiap kelas ('<=50K' dan '>50K').

Temuan:

Berdasarkan visualisasi, terlihat bahwa distribusi kelas pada kolom 'Class' cenderung seimbang (balanced), di mana jumlah entri untuk setiap kelas tidak memiliki perbedaan yang signifikan.

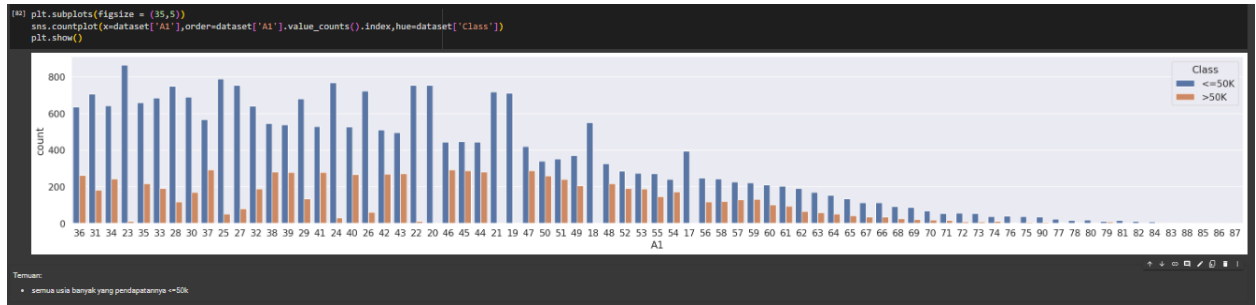


6. Visualisasi Distribusi Kelas berdasarkan Usia:

- Menggunakan `plt.subplots(figsize=(35, 5))`, kita setel ukuran plot untuk visualisasi yang lebih baik."
- Dengan `sns.countplot(x=dataset['A1'], order=dataset['A1'].value_counts().index, hue=dataset['Class'])`, kita buat count plot untuk distribusi kelas berdasarkan usia.
- Hasilnya ditampilkan menggunakan `plt.show()`.

Temuan:

Berdasarkan visualisasi, terlihat bahwa sebagian besar usia memiliki pendapatan <=50K.

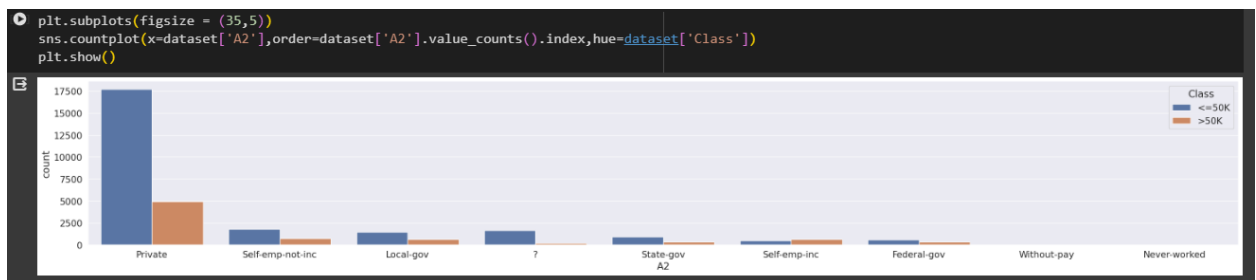


7. Visualisasi Distribusi Kelas berdasarkan Kelas Pekerja:

- Menggunakan `plt.subplots(figsize=(35, 5))`, kita setel ukuran plot untuk visualisasi yang lebih baik.
- Dengan `sns.countplot(x=dataset['A2'], order=dataset['A2'].value_counts().index, hue=dataset['Class'])`, kita buat count plot untuk distribusi kelas berdasarkan kelas pekerja.
- Hasilnya ditampilkan menggunakan `plt.show()`.

Catatan Singkat:

Visualisasi ini memberikan gambaran distribusi kelas berdasarkan kelas pekerja dalam dataset.

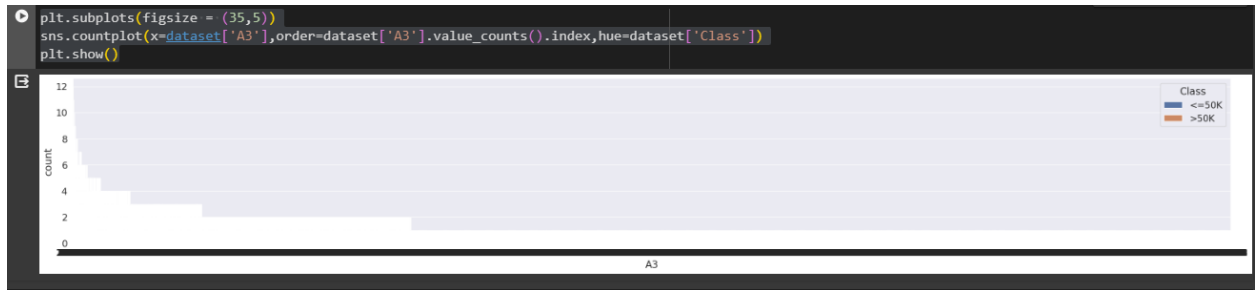


8. Visualisasi Distribusi Kelas berdasarkan Pendidikan:

- Menggunakan `plt.subplots(figsize=(35, 5))`, kita atur ukuran plot untuk visualisasi yang optimal.
- Dengan `sns.countplot(x=dataset['A3'], order=dataset['A3'].value_counts().index, hue=dataset['Class'])`, kita buat count plot untuk distribusi kelas berdasarkan tingkat pendidikan.
- Hasilnya ditampilkan menggunakan `plt.show()`.

Catatan Singkat:

Visualisasi ini memberikan gambaran distribusi kelas berdasarkan tingkat pendidikan dalam dataset.

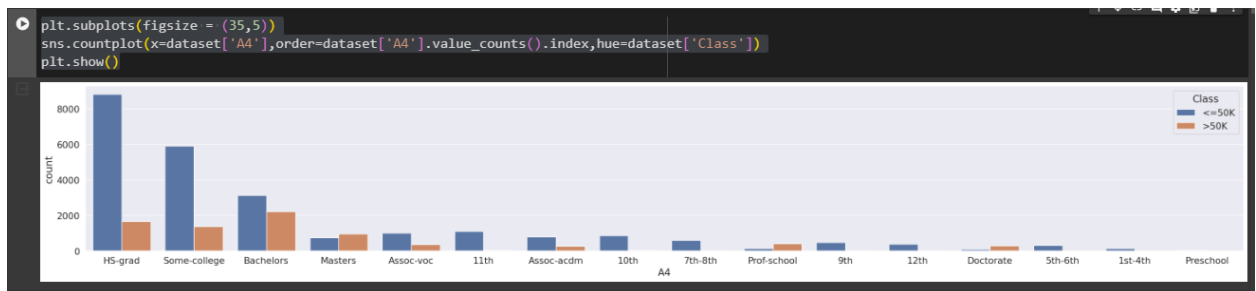


9. Visualisasi Distribusi Kelas berdasarkan Tingkat Pendidikan:

- Menggunakan **plt.subplots(figsize=(35, 5))**, ukuran plot disetel agar visualisasi optimal.
- Dengan **sns.countplot(x=dataset['A4'], order=dataset['A4'].value_counts().index, hue=dataset['Class'])**, count plot dibuat untuk distribusi kelas berdasarkan tingkat pendidikan.
- Hasilnya ditampilkan melalui **plt.show()**.

Catatan Singkat:

Visualisasi ini memberikan gambaran distribusi kelas berdasarkan tingkat pendidikan dalam dataset.

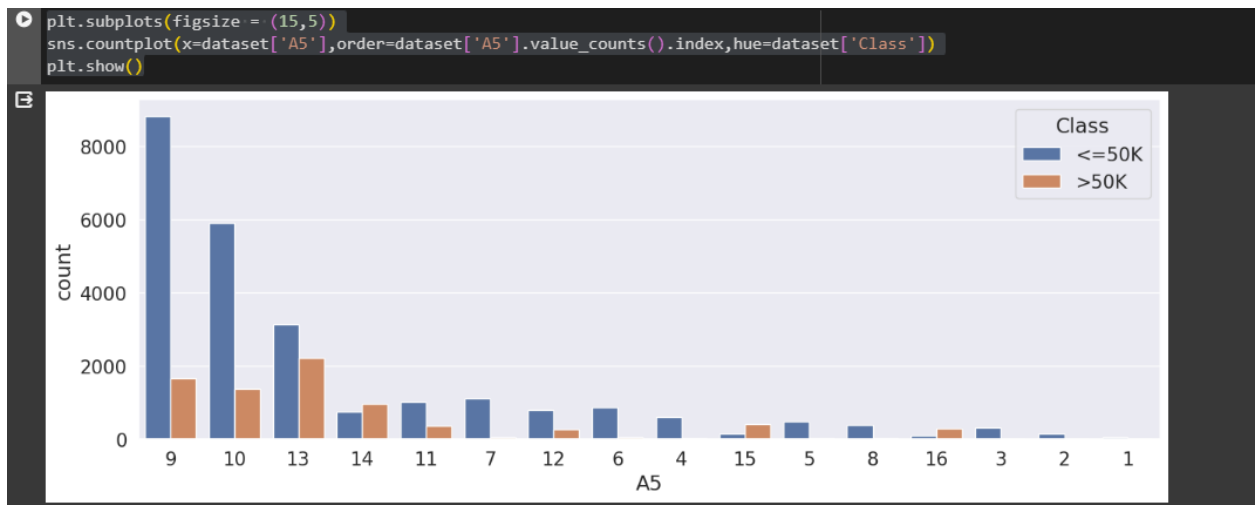


10. Visualisasi Kelas berdasarkan Status Perkawinan:

- Menggunakan **plt.subplots(figsize=(15, 5))** dan **sns.countplot(x=dataset['A5'], order=dataset['A5'].value_counts().index, hue=dataset['Class'])**, distribusi kelas ditampilkan berdasarkan status perkawinan.
- Hasilnya dapat dilihat melalui **plt.show()**.

Temuan:

- Distribusi kelas berdasarkan status perkawinan terlihat pada visualisasi.

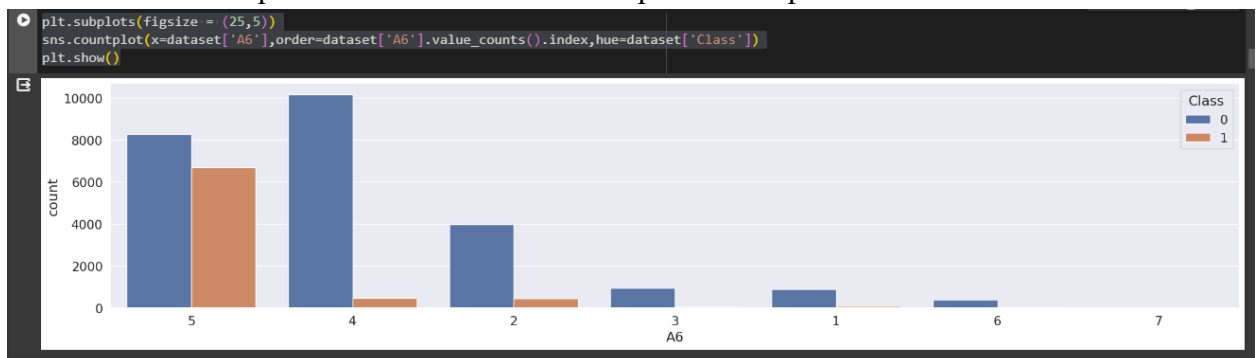


11. Visualisasi Distribusi Kelas berdasarkan Status Pernikahan:

- "Menggunakan `plt.subplots(figsize=(25, 5))`, ukuran plot disetel untuk visualisasi yang baik."
- "Dengan `sns.countplot(x=dataset['A6'], order=dataset['A6'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan status pernikahan."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan status pernikahan pada visualisasi ini."

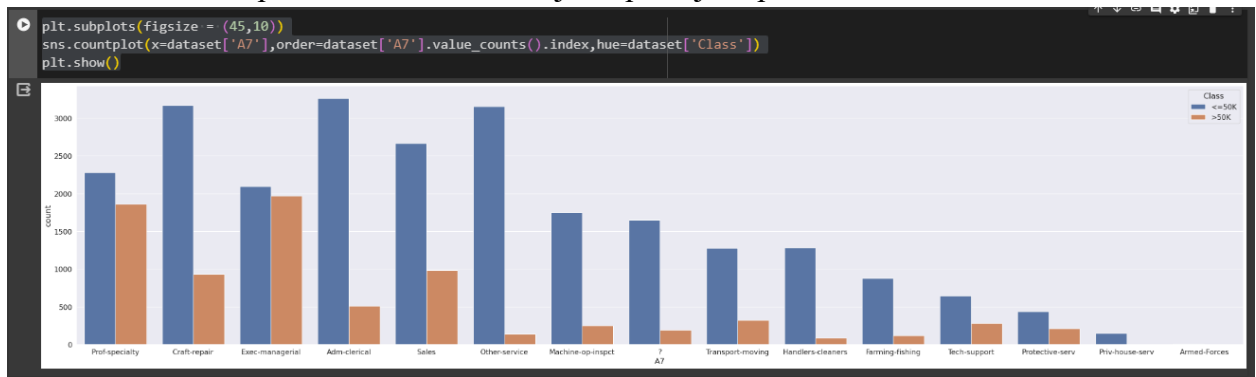


12. Visualisasi Distribusi Kelas berdasarkan Pekerjaan:

- "Menggunakan `plt.subplots(figsize=(45, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A7'], order=dataset['A7'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan pekerjaan."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan jenis pekerjaan pada visualisasi ini."

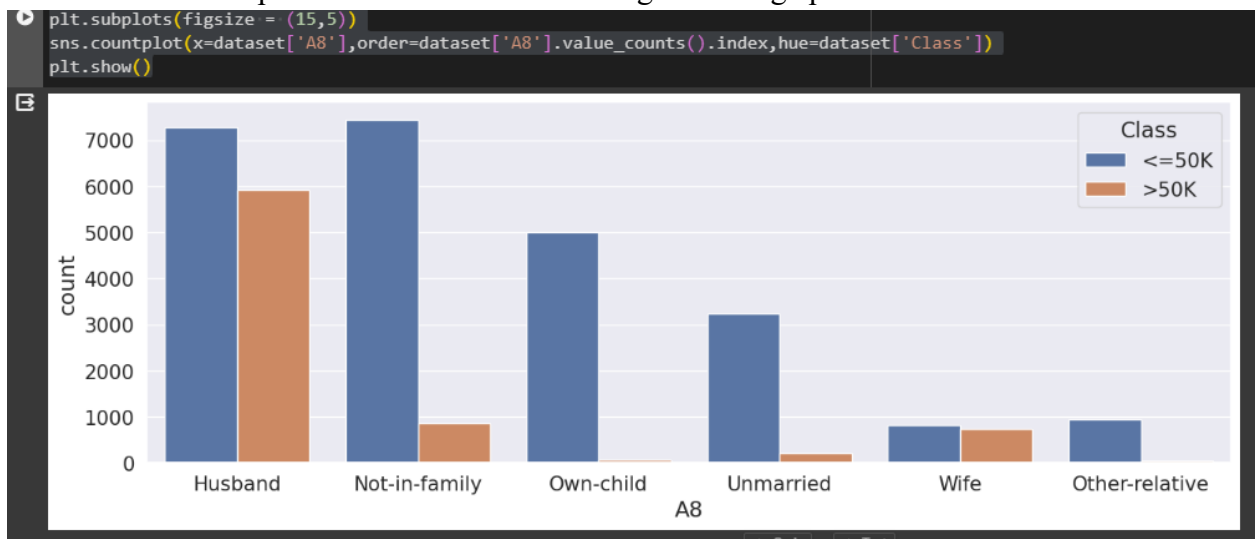


13. Visualisasi Distribusi Kelas berdasarkan Hubungan Keluarga:

- "Menggunakan `plt.subplots(figsize=(15, 5))`, ukuran plot disetel untuk visualisasi yang baik."
- "Dengan `sns.countplot(x=dataset['A8'], order=dataset['A8'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan hubungan keluarga."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan hubungan keluarga pada visualisasi ini."

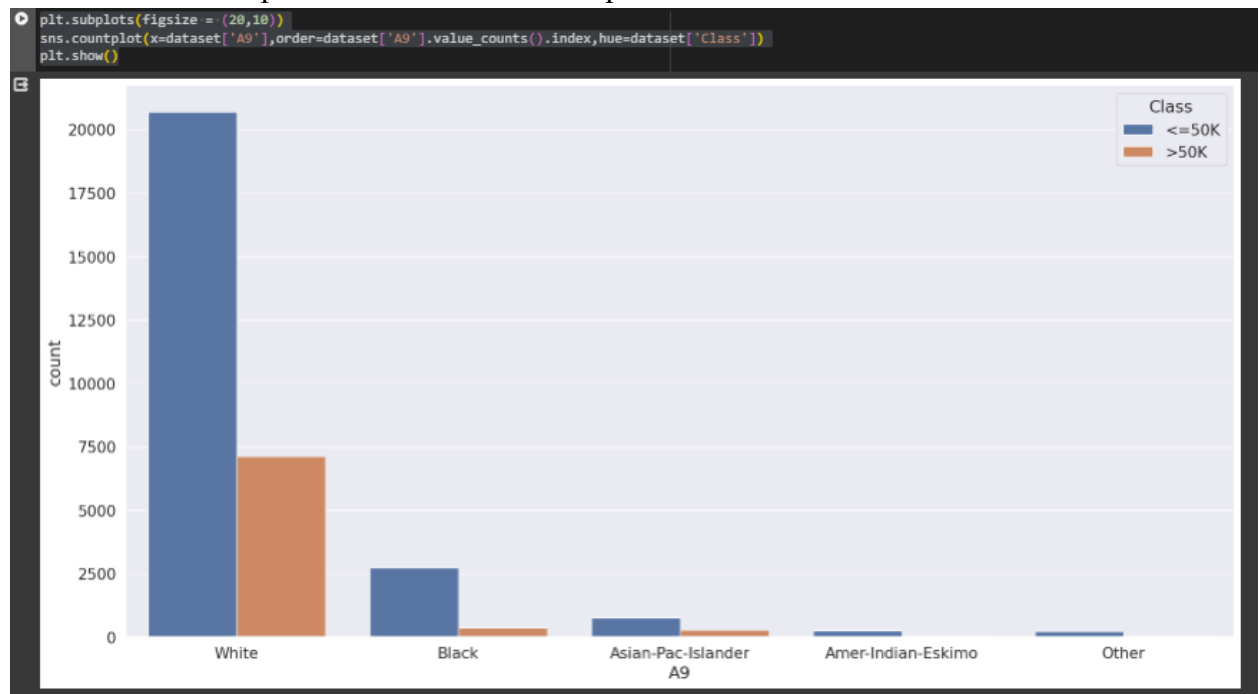


14. Visualisasi Distribusi Kelas berdasarkan Ras:

- "Menggunakan `plt.subplots(figsize=(20, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A9'], order=dataset['A9'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan ras."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan ras pada visualisasi ini."

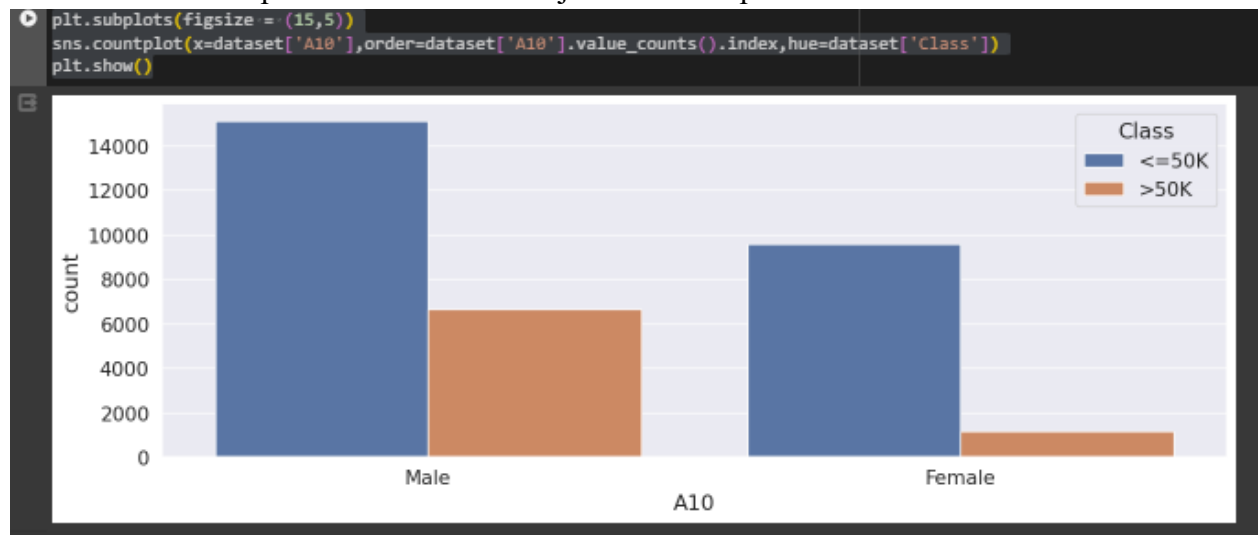


15. Visualisasi Distribusi Kelas berdasarkan Jenis Kelamin:

- "Menggunakan `plt.subplots(figsize=(15, 5))`, ukuran plot disetel untuk visualisasi yang baik."
- "Dengan `sns.countplot(x=dataset['A10'], order=dataset['A10'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan jenis kelamin."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan jenis kelamin pada visualisasi ini."



16. Visualisasi Distribusi Kelas berdasarkan Keuntungan Kapital:

- "Menggunakan `plt.subplots(figsize=(100, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A11'], order=dataset['A11'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan keuntungan kapital."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan keuntungan kapital pada visualisasi ini."

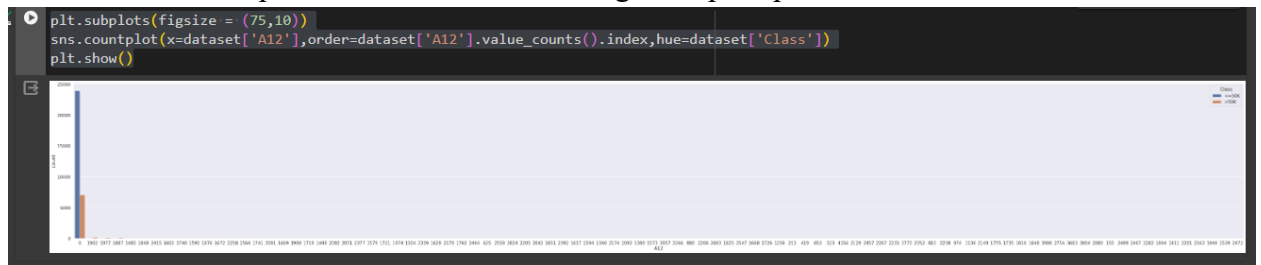


17. Visualisasi Distribusi Kelas berdasarkan Kerugian Kapital:

- "Menggunakan `plt.subplots(figsize=(75, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A12'], order=dataset['A12'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan kerugian kapital."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan kerugian kapital pada visualisasi ini."

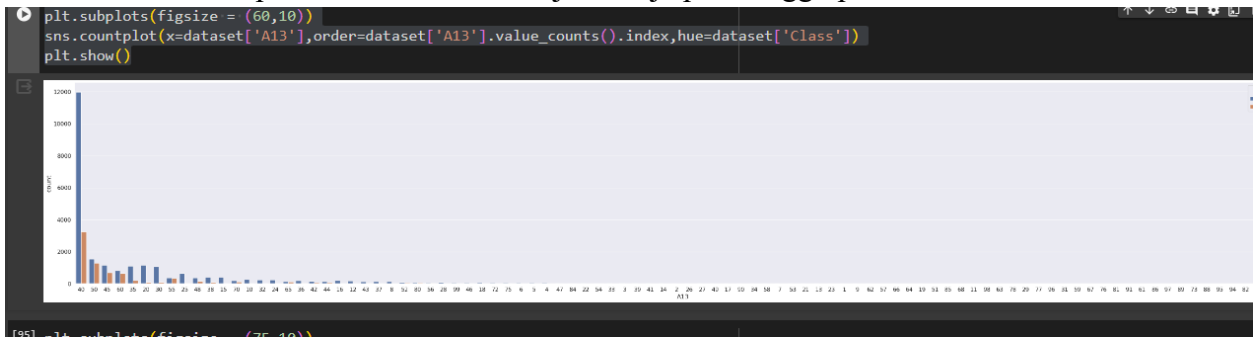


18. Visualisasi Distribusi Kelas berdasarkan Jam Kerja per Minggu:

- "Menggunakan `plt.subplots(figsize=(60, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A13'], order=dataset['A13'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan jam kerja per minggu."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

- "Distribusi kelas dapat dilihat berdasarkan jam kerja per minggu pada visualisasi ini."

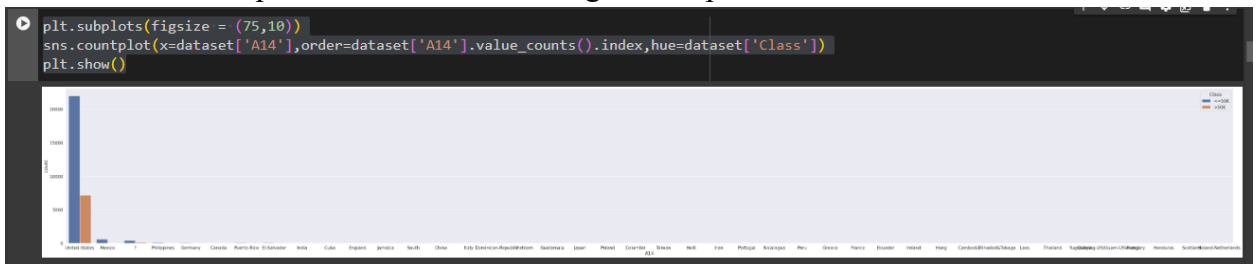


19. Visualisasi Distribusi Kelas berdasarkan Negara Asal:

- "Menggunakan `plt.subplots(figsize=(75, 10))`, ukuran plot disetel agar visualisasi optimal."
- "Dengan `sns.countplot(x=dataset['A14'], order=dataset['A14'].value_counts().index, hue=dataset['Class'])`, count plot dibuat untuk distribusi kelas berdasarkan negara asal."
- "Hasilnya ditampilkan melalui `plt.show()`."

Temuan:

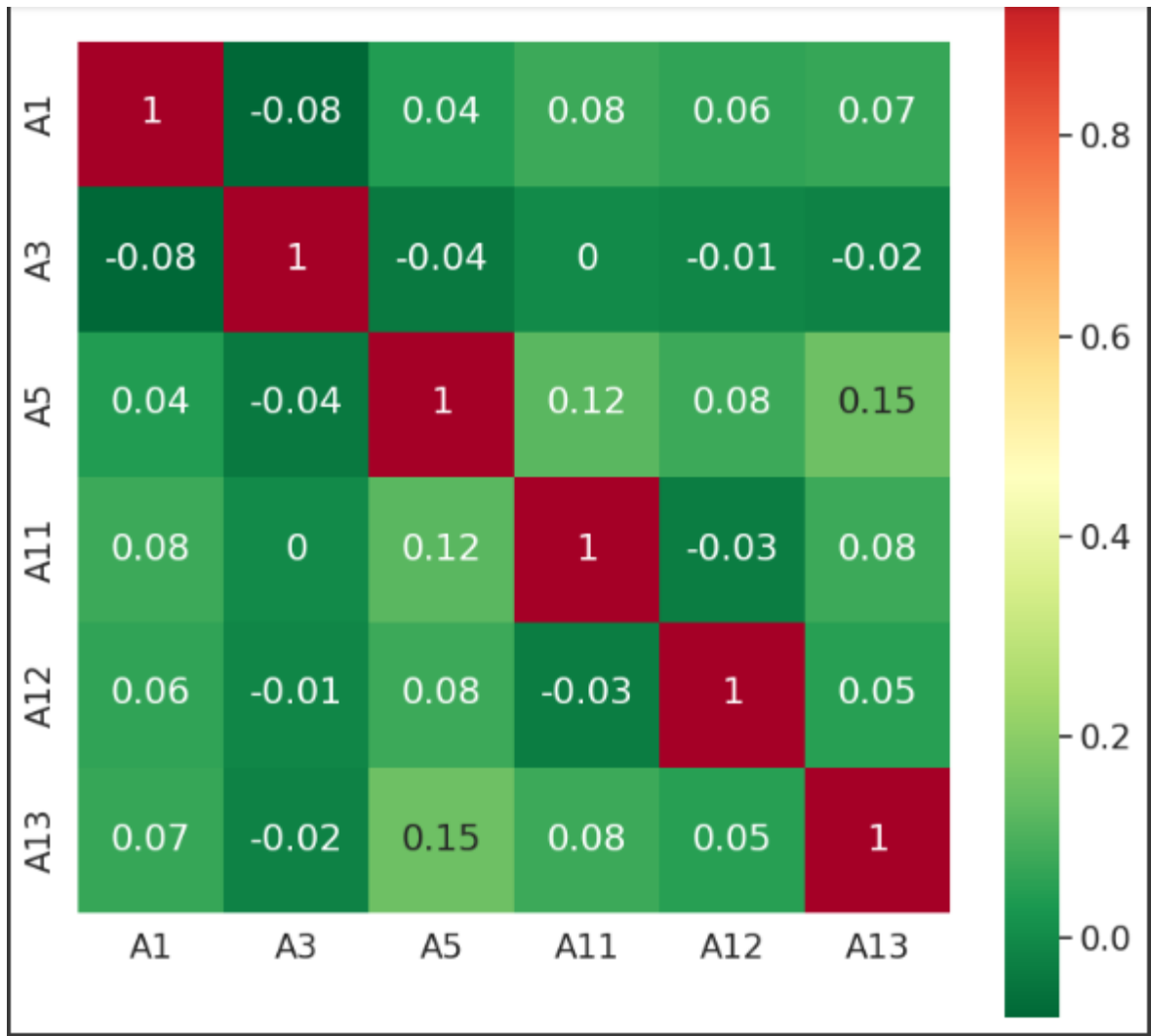
- "Distribusi kelas dapat dilihat berdasarkan negara asal pada visualisasi ini."



20. **Analisis Korelasi Tahap 1:** "Menggunakan `dataset.corr()`, korelasi antar variabel dihitung, dan hasilnya divisualisasikan melalui heatmap dengan ukuran plot yang disetel sesuai, memberikan gambaran korelasi pada dataset."

Cek Korelasi Tahap 1

```
# Correlation Heatmap
correlation = dataset.corr()
plt.subplots(figsize = (9,9))
sns.heatmap(correlation.round(2),
            annot = True,
            vmax = 1,
            square = True,
            cmap = 'RdYlGn_r')
plt.show()
```



Memvalidasi Data

Validasi Data:

- "Dilakukan pengecekan untuk melihat apakah masih ada data yang tersisa atau nilai yang hilang."
- "Kelas terlihat hampir seimbang."
- "Tidak semua variabel muncul karena data masih dalam bentuk objek dan belum diklasifikasikan."
- "Selanjutnya, perlu dilakukan transformasi objek menjadi angka dan proses menangani data yang hilang."

Menentukan Object Data

1. Feature yg akan digunakan pada case ini adalah adalah A1-A15 dan label target yang digunakan adalah Class. Untuk memberikan keterangan keterangan object data dapat digenerate menggunakan table generator yang dapat di akses di https://www.tablesgenerator.com/markdown_tables

Tables Generator

LaTeX

HTML

File

Edit

Table

Column

Row

Help

B

I

☐

A

B

C

1

2

3

Generate

☐

 Put tabs between columns

☐

 Compact mode

☐

 Line breaks as

2.

File Edit Table Column Row Help

B

I

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	cap
1	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0
2	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0
3	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0
4	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0
5	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0
6	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0
7	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0
8	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	0
9	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	0
10	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	0
11											

Generate

Expand

3.

Result (click "Generate" to refresh)

Preview **Copy to clipboard**

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain
1	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0
2	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0
3	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0
4	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0
5	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0
6	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0
7	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0
8	74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	0
9	68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	0
10	41	Private	78037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	0
11	45	Private	172274	Doctorate	16	Divorced	Prof-specialty	Unmarried	Black	Female	0
12	38	Self-emp-not-inc	164526	Prof-school	15	Never-married	Prof-specialty	Not-in-family	White	Male	0
13	52	Private	129177	Bachelors	13	Widowed	Other-service	Not-in-family	White	Female	0
14	32	Private	136204	Masters	14	Separated	Exec-managerial	Not-in-family	White	Male	0
15	51	?	172175	Doctorate	16	Never-married	?	Not-in-family	White	Male	0

4. Paste pada text di Phyton

```

| age | workclass | fnlwgt | education | education.num | marital.
status | occupation | relationship | race |
sex | capital.gain | capital.loss | hours.per.week | native.
country |
|-----|-----|-----|-----|-----|
| 90 | ? | 77053 | HS-grad | 9 |
Widowed | ? | Not-in-family |
White | Female | 0 | 4356 | 40 |
United-States | <=50K |
| 82 | Private | 132870 | HS-grad | 9 |
Widowed | Exec-managerial | Not-in-family |
White | Female | 0 | 4356 | 18 |
United-States | <=50K |
| 66 | ? | 186061 | Some-college | 10 |
Widowed | ? | Unmarried |
Black | Female | 0 | 4356 | 40 |
United-States | <=50K |
| 54 | Private | 140359 | 7th-8th | 4 |
Divorced | Machine-op-inspct | Unmarried |
White | Female | 0 | 3900 | 40 |
United-States | <=50K |
| 41 | Private | 264663 | Some-college | 10 |

```

Membersihkan Data

Pada tahap ini, langkah-langkah pembersihan data yang dilakukan meliputi:

1. Perintah **dataset.replace("?", np.nan, inplace=True)** digunakan untuk mengganti nilai "?" dengan NaN (Not a Number) pada dataset, membersihkan data dari nilai yang tidak diketahui.

```
[97] dataset.replace("?", np.nan, inplace=True) #membersihkan data yang masih ada tanda tanya
```

2. Dapat melihat lima baris pertama dari dataset menggunakan perintah **dataset.head()**. Perintah ini memberikan tampilan singkat tentang struktur dan nilai-nilai dalam dataset.

```
[98] dataset.head()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
0	90	NaN	77053	HS-grad	9	Widowed	NaN	Not-in-family	White	Female	0	4356	40	United-States	<=50K
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
2	66	NaN	186061	Some-college	10	Widowed	NaN	Unmarried	Black	Female	0	4356	40	United-States	<=50K
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K

3. Perintah **dataset.tail()** digunakan untuk menampilkan lima baris terakhir dari dataset, memberikan gambaran singkat tentang struktur dan nilai-nilai terakhir dalam dataset.

```
dataset.tail()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male	0	0	40	United-States	<=50K
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K

4. Data telah berhasil dideteksi dan diubah menjadi NaN pada kolom-kolom kategori yang mengandung nilai "?". Ini merupakan langkah awal dalam membersihkan data dari nilai yang tidak valid atau tidak diketahui.

```
categorical_col = []
for column in dataset.columns:
    if dataset[column].dtype == object and len(dataset[column].unique()) <= 50: #mengecek level/isi data kolom (bisa object, int, dan tipe data lainnya)
        categorical_col.append(column)
        print(f"{column} : {dataset[column].unique()}")
        print("=====")
```

```

A2 : [nan 'Private' 'State-gov' 'Federal-gov' 'Self-emp-not-inc' 'Self-emp-inc'
      'Local-gov' 'Without-pay' 'Never-worked']
=====
A4 : ['HS-grad' 'Some-college' '7th-8th' '10th' 'Doctorate' 'Prof-school'
      'Bachelors' 'Masters' '11th' 'Assoc-acdm' 'Assoc-voc' '1st-4th' '5th-6th'
      '12th' '9th' 'Preschool']
=====
A6 : ['Widowed' 'Divorced' 'Separated' 'Never-married' 'Married-civ-spouse'
      'Married-spouse-absent' 'Married-AF-spouse']
=====
A7 : [nan 'Exec-managerial' 'Machine-op-inspct' 'Prof-specialty'
      'Other-service' 'Adm-clerical' 'Craft-repair' 'Transport-moving'
      'Handlers-cleaners' 'Sales' 'Farming-fishing' 'Tech-support'
      'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
=====
A8 : ['Not-in-family' 'Unmarried' 'Own-child' 'Other-relative' 'Husband' 'Wife']
=====
A9 : ['White' 'Black' 'Asian-Pac-Islander' 'Other' 'Amer-Indian-Eskimo']
=====
A10 : ['Female' 'Male']
=====
A14 : ['United-States' nan 'Mexico' 'Greece' 'Vietnam' 'China' 'Taiwan' 'India'
       'Philippines' 'Trinidad&Tobago' 'Canada' 'South' 'Holand-Netherlands'
       'Puerto-Rico' 'Poland' 'Iran' 'England' 'Germany' 'Italy' 'Japan' 'Hong'
       'Honduras' 'Cuba' 'Ireland' 'Cambodia' 'Peru' 'Nicaragua'
       'Dominican-Republic' 'Haiti' 'El-Salvador' 'Hungary' 'Columbia'
       'Guatemala' 'Jamaica' 'Ecuador' 'France' 'Yugoslavia' 'Scotland'
       'Portugal' 'Laos' 'Thailand' 'Outlying-US(Guam-USVI-etc)']
=====
Class : ['<=50K' '>50K']
=====

```

Note : Data sudah berhasil dideteksi dan berubah menjadi NaN

5. Jika hasil eksekusi kode `dataset.isnull().values.any()` mengembalikan nilai **True**, itu berarti ada setidaknya satu nilai yang hilang (NaN) dalam dataset.

```

dataset.isnull().values.any() #Jika True artinya ada data yg hilang
True

```

6. Kode `dataset.loc[:, dataset.isnull().any()].columns` akan memberikan daftar kolom (feature) yang memiliki setidaknya satu nilai yang hilang (NaN) dalam dataset.

```

[102] dataset.loc[:, dataset.isnull().any()].columns #Feature yg terdiri data hilang
Index(['A2', 'A7', 'A14'], dtype='object')

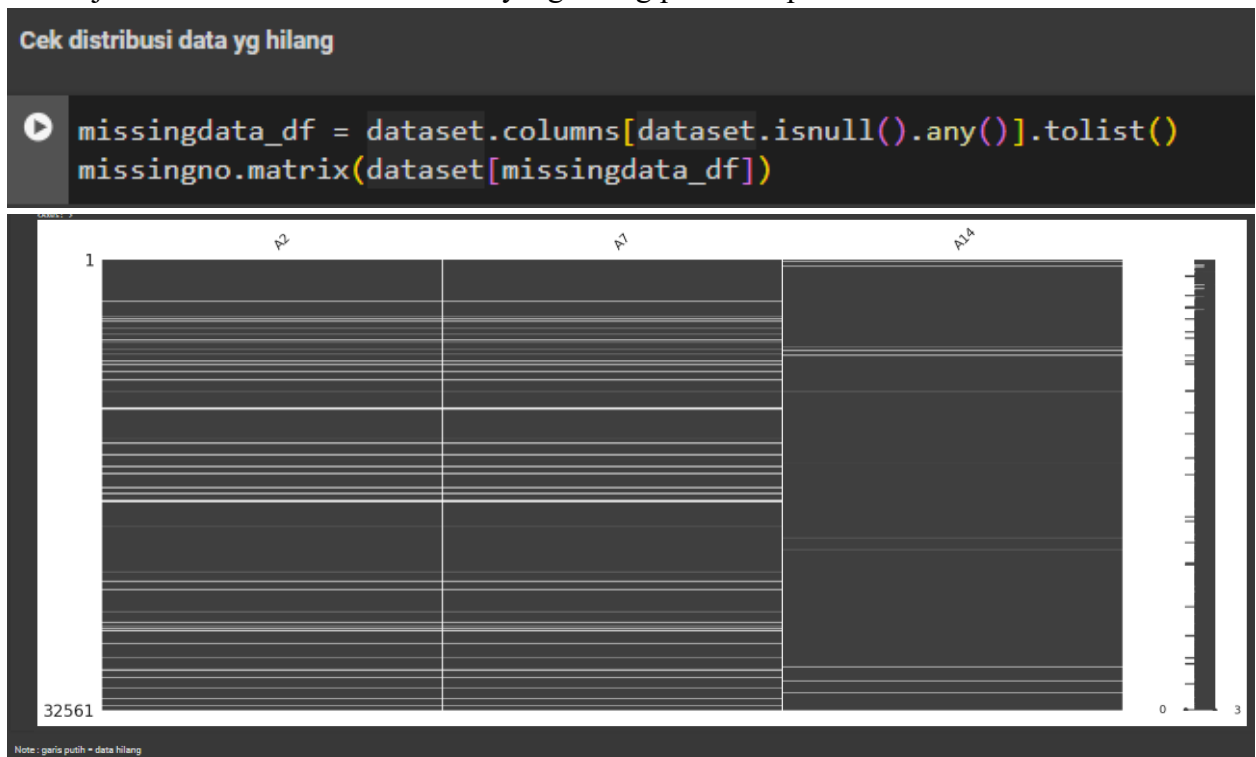
```

7. `dataset.info()` memberikan informasi rinci tentang dataset, termasuk jumlah entri, tipe data, dan apakah ada nilai yang hilang dalam setiap kolom.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0    A1      32561 non-null  int64  
 1    A2      30725 non-null  object  
 2    A3      32561 non-null  int64  
 3    A4      32561 non-null  object  
 4    A5      32561 non-null  int64  
 5    A6      32561 non-null  object  
 6    A7      30718 non-null  object  
 7    A8      32561 non-null  object  
 8    A9      32561 non-null  object  
 9   A10     32561 non-null  object  
10   A11     32561 non-null  int64  
11   A12     32561 non-null  int64  
12   A13     32561 non-null  int64  
13   A14     31978 non-null  object  
14  Class   32561 non-null  object  
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

8. **missingno.matrix(dataset[missingdata_df])** digunakan untuk memvisualisasikan pola data yang hilang (missing values) dalam bentuk matriks. Matriks tersebut akan menunjukkan di mana letak nilai-nilai yang hilang pada setiap kolom dataset.



9. Kode di bawah mengubah kolom kategorikal menjadi numerik dengan menggunakan peta atau kamus. Setiap kategori diberikan nilai numerik sesuai dengan kamus yang telah ditentukan. Proses ini memungkinkan penggunaan data kategorikal dalam model machine learning yang memerlukan input numerik.

```
Mengubah kategorik menjadi numerik jika 2 kategori ganti dengan 0,1; jika >2 kategori ganti dengan 1,2,3,dst

# Mendefinisikan kamus
A2 = {'Private': 1, 'State-gov': 2, 'Federal-gov': 3, 'Self-emp-not-inc': 4, 'Self-emp-inc': 5, 'Local-gov': 6, 'Without-pay': 7, 'Never-worked': 8}
A4 = {'HS-grad': 1, 'Some-college': 2, '7th-8th': 3, '10th': 4, 'Doctorate': 5, 'Prof-school': 6, 'Bachelors': 7, 'Masters': 8, '11th': 9, 'Assoc-voc': 10}
A6 = {'Widowed': 1, 'Divorced': 2, 'Separated': 3, 'Never-married': 4, 'Married-civ-spouse': 5, 'Married-spouse-absent': 6, 'Married-AF-spouse': 7}
A7 = {'Exec-managerial': 1, 'Machine-op-inspct': 2, 'Prof-specialty': 3, 'Other-service': 4, 'Adm-clerical': 5, 'Craft-repair': 6, 'Transport-moving': 7}
A8 = {'Not-in-family': 1, 'Unmarried': 2, 'Own-child': 3, 'Other-relative': 4, 'Husband': 5, 'Wife': 6}
A9 = {'White': 1, 'Black': 2, 'Asian-Pac-Islander': 3, 'Other': 4, 'Amer-Indian-Eskimo': 6}
A10 = {'Female': 1, 'Male': 0}
A14 = {'United-States': 1, 'Mexico': 2, 'Greece': 3, 'Vietnam': 4, 'China': 5, 'Taiwan': 6, 'India': 7, 'Philippines': 8, 'Trinidad&Tobago': 9, 'Other': 10}

# Menggunakan fungsi peta
dataset['A2'] = dataset['A2'].map(A2)
dataset['A4'] = dataset['A4'].map(A4)
dataset['A6'] = dataset['A6'].map(A6)
dataset['A7'] = dataset['A7'].map(A7)
dataset['A8'] = dataset['A8'].map(A8)
dataset['A9'] = dataset['A9'].map(A9)
dataset['A10'] = dataset['A10'].map(A10)
dataset['A14'] = dataset['A14'].map(A14)
```

10. Output dari **dataset.info()** memberikan informasi mengenai dataset, termasuk tipe data setiap kolom, jumlah nilai non-null, dan penggunaan memori. Dengan melihat output tersebut, kita dapat mengetahui struktur umum dari dataset, apakah terdapat nilai yang hilang, serta tipe data masing-masing kolom.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
#   Column   Non-Null Count  Dtype
---  -
0   A1        32561 non-null  int64
1   A2        29427 non-null  float64
2   A3        32561 non-null  int64
3   A4        32561 non-null  int64
4   A5        32561 non-null  int64
5   A6        32561 non-null  int64
6   A7        30718 non-null  float64
7   A8        32561 non-null  int64
8   A9        32561 non-null  int64
9   A10       32561 non-null  int64
10  A11       32561 non-null  int64
11  A12       32561 non-null  int64
12  A13       32561 non-null  int64
13  A14       31978 non-null  float64
14  Class     32561 non-null  object
dtypes: float64(3), int64(11), object(1)
memory usage: 3.7+ MB
```

11. Data hilang pada kolom A2, A7, dan A14 telah diisi dengan nilai median.

Mengisi data hilang dengan nilai median

```
[107] #Data hilang: A2, A7, A14

median_value=dataset['A2'].median()
dataset['A2']=dataset['A2'].fillna(median_value)

median_value=dataset['A7'].median()
dataset['A7']=dataset['A7'].fillna(median_value)

median_value=dataset['A14'].median()
dataset['A14']=dataset['A14'].fillna(median_value)
```

12. Tidak ada data yang berjenis objek (non-numerik) pada dataset setelah dilakukan pengubahan tipe data dan pengisian nilai data yang hilang.

Cek kembali



dataset.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
#   Column  Non-Null Count  Dtype
---  -
0    A1      32561 non-null   int64
1    A2      32561 non-null   float64
2    A3      32561 non-null   int64
3    A4      32561 non-null   int64
4    A5      32561 non-null   int64
5    A6      32561 non-null   int64
6    A7      32561 non-null   float64
7    A8      32561 non-null   int64
8    A9      32561 non-null   int64
9    A10     32561 non-null   int64
10   A11     32561 non-null   int64
11   A12     32561 non-null   int64
12   A13     32561 non-null   int64
13   A14     32561 non-null   float64
14   Class   32561 non-null   object
dtypes: float64(3), int64(11), object(1)
memory usage: 3.7+ MB
```

TEMUAN : Tidak ada data yang numerik /dianggap objek

Mengkonstruksi Data

Mengkonstruksi data dilakukan dengan mengonversi data yang masih dalam bentuk objek menjadi tipe data numerik dan memastikan bahwa semua data telah diperlakukan secara sesuai.

1. Melakukan konversi tipe data float menjadi integer setelah mengisi nilai NaN dengan 0.

```
# Convert float to int after filling NaN with 0
dataset['A2'] = pd.to_numeric(dataset['A2'].fillna(0), errors='coerce').astype(int)

dataset['A4'] = pd.to_numeric(dataset['A4'].fillna(0), errors='coerce').astype(int)

dataset['A6'] = pd.to_numeric(dataset['A6'].fillna(0), errors='coerce').astype(int)

dataset['A7'] = pd.to_numeric(dataset['A7'].fillna(0), errors='coerce').astype(int)

dataset['A8'] = pd.to_numeric(dataset['A8'].fillna(0), errors='coerce').astype(int)

dataset['A9'] = pd.to_numeric(dataset['A9'].fillna(0), errors='coerce').astype(int)

dataset['A10'] = pd.to_numeric(dataset['A10'].fillna(0), errors='coerce').astype(int)

dataset['A14'] = pd.to_numeric(dataset['A14'].fillna(0), errors='coerce').astype(int)
```

2. Menampilkan dataset setelah dilakukan konversi tipe data dan pengisian nilai NaN.

```
#cek dataset
dataset
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class
0	90	1	77053	1	9	1	5	1	1	1	0	4356	40	1	<=50K
1	82	1	132870	1	9	1	1	1	1	1	0	4356	18	1	<=50K
2	66	1	186061	2	10	1	5	2	2	1	0	4356	40	1	<=50K
3	54	1	140359	3	4	2	2	2	1	1	0	3900	40	1	<=50K
4	41	1	264663	2	10	3	3	3	1	1	0	3900	40	1	<=50K
...
32556	22	1	310152	2	10	4	12	1	1	0	0	0	40	1	<=50K
32557	27	1	257302	10	12	5	11	6	1	1	0	0	38	1	<=50K
32558	40	1	154374	1	9	5	2	5	1	0	0	0	40	1	>50K
32559	58	1	151910	1	9	1	5	2	1	1	0	0	40	1	<=50K
32560	22	1	201490	1	9	4	5	3	1	0	0	0	20	1	<=50K

32561 rows × 15 columns

3. Menampilkan informasi tentang dataset setelah dilakukan konversi tipe data dan pengisian nilai NaN.

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
#   Column  Non-Null Count  Dtype  
---  -
0   A1      32561 non-null   int64  
1   A2      32561 non-null   int64  
2   A3      32561 non-null   int64  
3   A4      32561 non-null   int64  
4   A5      32561 non-null   int64  
5   A6      32561 non-null   int64  
6   A7      32561 non-null   int64  
7   A8      32561 non-null   int64  
8   A9      32561 non-null   int64  
9   A10     32561 non-null   int64  
10  A11     32561 non-null   int64  
11  A12     32561 non-null   int64  
12  A13     32561 non-null   int64  
13  A14     32561 non-null   int64  
14  Class   32561 non-null   object  
dtypes: int64(14), object(1)
memory usage: 3.7+ MB
```

Catatan: Data telah berhasil dimodifikasi dan tidak memiliki objek

Pada tahap ini, dilakukan penghapusan nilai konstan dari dataset. Langkah ini bertujuan untuk mengeliminasi kolom-kolom yang tidak memberikan variasi informasi karena hanya memiliki satu nilai unik. Hal ini dapat mempermudah analisis dan pemodelan data, mengurangi kompleksitas, dan memastikan fokus pada fitur-fitur yang lebih informatif. Meskipun langkah ini dapat membantu membersihkan dataset, tetapi tetap perlu diperhatikan agar informasi yang dihilangkan tidak krusial untuk analisis lebih lanjut. Sehingga, sebaiknya selalu lakukan pengecekan sebelum dan sesudah penghapusan untuk memastikan integritas dan keberlanjutan analisis data.

4. Pada tahap ini, dilakukan penghapusan kolom yang memiliki nilai konstan dalam dataset menggunakan perintah **dataset = dataset.loc[:, dataset.apply(pd.Series.nunique) != 1]**, bertujuan untuk menyederhanakan struktur data dengan fokus pada fitur-fitur yang lebih informatif

```
[112] dataset = dataset.loc[:, dataset.apply(pd.Series.nunique) != 1]
```

5. Info dataset setelah menghapus nilai konstan menunjukkan bahwa sejumlah kolom telah dihapus, dan data telah berhasil disederhanakan.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0   A1      32561 non-null  int64  
 1   A2      32561 non-null  int64  
 2   A3      32561 non-null  int64  
 3   A4      32561 non-null  int64  
 4   A5      32561 non-null  int64  
 5   A6      32561 non-null  int64  
 6   A7      32561 non-null  int64  
 7   A8      32561 non-null  int64  
 8   A9      32561 non-null  int64  
 9   A10     32561 non-null  int64  
10  A11     32561 non-null  int64  
11  A12     32561 non-null  int64  
12  A13     32561 non-null  int64  
13  A14     32561 non-null  int64  
14  Class   32561 non-null  object  
dtypes: int64(14), object(1)
memory usage: 3.7+ MB
```

6. Temuan: Setelah menghapus nilai konstan, jumlah atribut berkurang menjadi 6, menunjukkan bahwa sebelumnya terdapat atribut dengan nilai konstan yang telah dihapus.

Temuan : Karena jumlah atribut berkurang menjadi 14 artinya ada data bernilai konstan

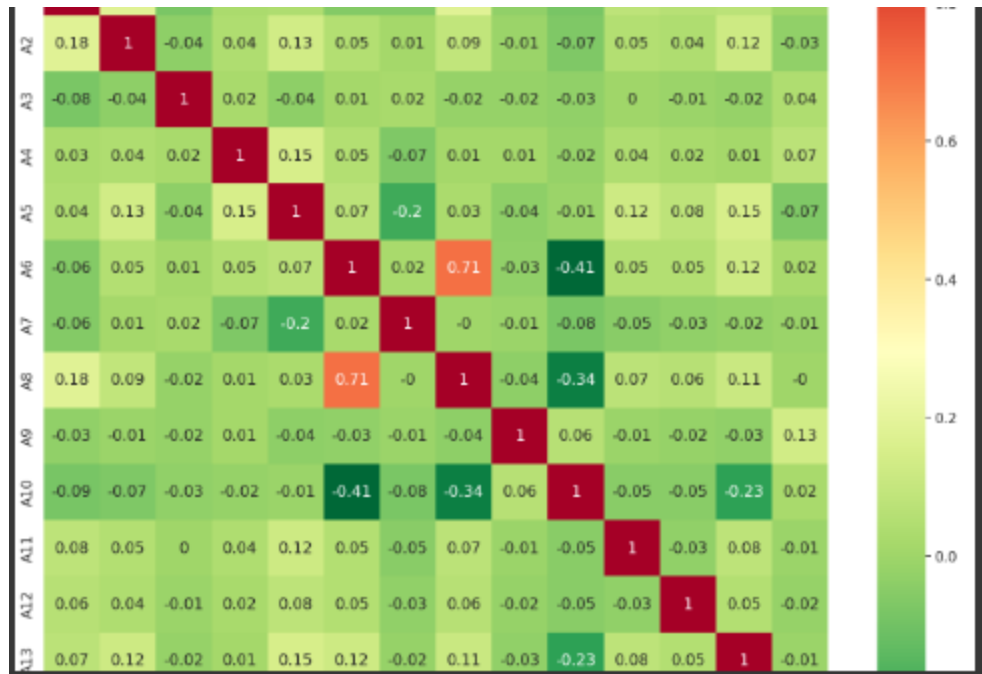
Cek data apakah ada data yang bernilai konstan. Jika iya, maka atribut anda akan berkurang dari jumlah atribut sebelumnya, dalam hal ini menjadi 6 data.

7. Mengecek korelasi antar variabel dengan menggunakan heatmap korelasi.

Cek korelasi tahap 2

```
# Correlation Heatmap
correlation = dataset.corr()
plt.subplots(figsize = (19,19))
sns.heatmap(correlation.round(2),
             annot = True,
             vmax = 1,
             square = True,
             cmap = 'RdYlGn_r')

plt.show()
```



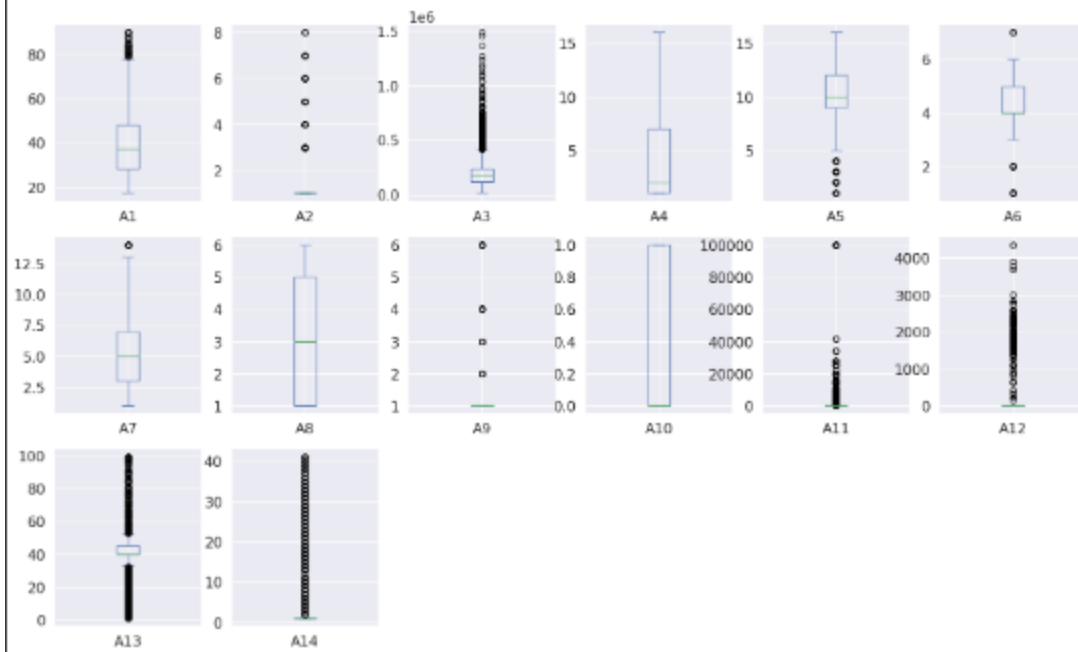
Temuan: Tidak ada atribut yang sama

- Menganalisis distribusi data menggunakan boxplot untuk setiap fitur.

Cek Boxplot

```
[ ] dataset.plot(kind='box',subplots=True,layout=(5,6), sharex=False,figsize = (20,20),
                  title='figure 1: Data distributions of all features')
plt.show()
```

figure 1: Data distributions of all features



Temuan: Ada banyak outlier di A3, A11, A12

Menentukan Label Data

Dalam bab ini, langkah-langkah untuk menentukan label data atau jumlah grup atau cluster dijelaskan. Penentuan ini disesuaikan dengan kasus yang ada. Jika tujuan utama adalah klasifikasi, langkah pertama adalah memberi label pada data (dalam hal ini, metode klasifikasi digunakan karena data sudah memiliki label pada atribut 'Class'). Namun, jika tujuan utamanya adalah pengelompokan, langkah pertama adalah mengidentifikasi jumlah cluster yang optimal dan menetapkan label kategori untuk setiap cluster. Proses ini memberikan dasar untuk memahami struktur data dan mempersiapkan langkah-langkah selanjutnya dalam analisis atau pemodelan data.

1. Kode `y = dataset['Class']` digunakan untuk menyimpan atribut kelas dari dataset, menandakan persiapan untuk melakukan klasifikasi atau membangun model yang memprediksi atribut kelas ('Class').

```
y = dataset['Class'] #case jika akan melakukan klasifikasi untuk membangun model
```

2. Kode `y` digunakan untuk mengecek isi variabel `y`, yang merupakan atribut kelas dari dataset, sebagai langkah verifikasi.

```
y #cek variable y
0      <=50K
1      <=50K
2      <=50K
3      <=50K
4      <=50K
...
32556  <=50K
32557  <=50K
32558  >50K
32559  <=50K
32560  <=50K
Name: Class, Length: 32561, dtype: object
```

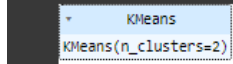
3. Kode ini menentukan jumlah cluster (dalam hal ini 2 cluster) dan menggunakan metode KMeans untuk melakukan klasterisasi pada dataset. Pastikan library yang dibutuhkan untuk klasterisasi telah diimport dan diperiksa pada bab sebelumnya.

```
[ ] #menentukan jumlah cluster (jumlah cluster = 2, dan metode clustering yg digunakan adalah KMeans)
km = KMeans(n_clusters=2)
#case klasterisasi
#untuk menjalankan perintah ini anda perlu memeriksa kembali library anda pada bab 1 untuk klasterisasi
```

4. Kode tersebut mendefinisikan objek `km` yang merupakan model KMeans dengan jumlah cluster sebanyak 2. Objek ini dapat digunakan untuk melakukan proses klasterisasi pada data.


```
#menentukan jumlah cluster (jumlah cluster = 2, dan metode clustering yg digunakan adalah KMeans)
km = KMeans(n_clusters=2)
#case klasterisasi
#untuk menjalankan perintah ini anda perlu memeriksa kembali library anda pada bab 1 untuk klasterisasi

[ ] km
```



5. Kode tersebut mengimpor library **pandas** dan modul **KMeans** dari library **sklearn.cluster** yang diperlukan untuk melakukan klasterisasi menggunakan algoritma KMeans.

```
[ ] import pandas as pd
    from sklearn.cluster import KMeans
```

6. Kode tersebut membuat list bernama **selected_features** yang berisi satu fitur, yaitu 'Class'. List ini kemungkinan akan digunakan untuk memilih fitur-fitur tertentu dari dataset.

```
[ ] selected_features = ['Class']
```

7. Kode tersebut membuat DataFrame baru bernama **features** yang berisi kolom 'Class' dari dataset. Data ini mungkin akan digunakan sebagai fitur dalam proses klasterisasi atau pemodelan.

```
[ ] features = dataset[selected_features].copy()
```

8. Kode ini menggunakan fungsi **get_dummies** dari pandas untuk mengubah variabel kategorikal dalam **features** menjadi variabel biner (dummy/indikator). Hal ini membantu dalam proses pemodelan atau klasterisasi yang memerlukan input numerik.

```
[ ] features = pd.get_dummies(features)
```

9. KMeans dengan jumlah klaster (clusters) sebanyak 2 dan menggunakan nilai seed (random_state) 42 untuk memastikan reproduktibilitas hasil.

```
[ ] kmeans = KMeans(n_clusters=2, random_state=42)
```

10. Menambahkan kolom 'cluster' ke dalam dataset dengan menggunakan hasil klasterisasi dari KMeans yang telah dilakukan pada fitur yang telah dipilih sebelumnya.

```
dataset['cluster'] = kmeans.fit_predict(features)

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warning
warnings.warn(
```

11. Mengganti nilai klaster (0 dan 1) dengan label yang lebih deskriptif ('ditolak' dan 'diterima') pada kolom 'cluster'.

```
dataset['cluster'] = dataset['cluster'].map({0: 'ditolak', 1: 'diterima'})
```

12. Hasil cetakan dari variabel **features** akan menunjukkan data yang telah diubah menjadi representasi one-hot encoding setelah penerapan **pd.get_dummies(features)**.

```
print(features)
```

	Class_<=50K	Class_>50K
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0
...
32556	1	0
32557	1	0
32558	0	1
32559	1	0
32560	1	0

[32561 rows x 2 columns]

13. Output dari variabel **dataset** akan menunjukkan kumpulan data dengan penambahan kolom baru, yaitu **cluster**, yang mengindikasikan hasil dari proses klasterisasi menggunakan algoritma KMeans. Label dari klaster tersebut juga telah diubah menjadi "ditolak" dan "diterima".

[] dataset

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class	cluster
0	90	1	77053	1	9	1	5	1	1	1	0	4356	40	1	<=50K	ditolak
1	82	1	132870	1	9	1	1	1	1	1	0	4356	18	1	<=50K	ditolak
2	66	1	186061	2	10	1	5	2	2	1	0	4356	40	1	<=50K	ditolak
3	54	1	140359	3	4	2	2	2	1	1	0	3900	40	1	<=50K	ditolak
4	41	1	264663	2	10	3	3	3	1	1	0	3900	40	1	<=50K	ditolak
...
32556	22	1	310152	2	10	4	12	1	1	0	0	0	40	1	<=50K	ditolak
32557	27	1	257302	10	12	5	11	6	1	1	0	0	38	1	<=50K	ditolak
32558	40	1	154374	1	9	5	2	5	1	0	0	0	40	1	>50K	diterima
32559	58	1	151910	1	9	1	5	2	1	1	0	0	40	1	<=50K	ditolak
32560	22	1	201490	1	9	4	5	3	1	0	0	0	20	1	<=50K	ditolak

32561 rows x 16 columns

Membangun Model

Dalam Bab 8 ini, langkah-langkah untuk membangun model akan dijelaskan dengan detail. Ini termasuk pemilihan algoritma yang sesuai dengan jenis masalah yang dihadapi, pembagian data menjadi set pelatihan dan pengujian, dan proses pelatihan model. Setelah model dibangun, evaluasi kinerja akan dilakukan menggunakan metrik yang relevan, dan model kemudian dapat disesuaikan atau dioptimalkan sesuai kebutuhan. Langkah-langkah ini akan memberikan pemahaman yang kuat tentang bagaimana model diterapkan pada data dan bagaimana kinerjanya dapat diukur.

Berikut adalah beberapa langkah prapemrosesan data sebelum membangun model:

Pertama, kita perlu memperhatikan bahwa dataset kita memiliki fitur kategorikal dan numerik sebagai prediktor. Untuk model-model sebelumnya seperti regresi linier dan logistik, kita harus membuat variabel boneka untuk variabel kategorikal karena model-model tersebut hanya dapat memproses variabel numerik.

Namun, pohon keputusan dapat dengan mudah memproses variabel kategorikal tanpa perlu membuat variabel boneka. Meskipun demikian, kita masih perlu mengkodekan variabel kategorikal ke dalam format standar agar sklearn dapat memahaminya dan membangun pohon keputusan. Kita akan menggunakan kelas `LabelEncoder()` dari modul `sklearn.preprocessing` untuk melakukan ini.

Berikut adalah langkah-langkah untuk mengkodekan variabel kategorikal menggunakan `LabelEncoder`:

1. Dengan menggunakan `LabelEncoder`, kita dapat mempersiapkan variabel kategorikal untuk digunakan dalam membangun model.

```
from sklearn import preprocessing

dataset_categorical = dataset.select_dtypes(include=['object'])
dataset_categorical.head()
```

	Class	cluster
0	<=50K	ditolak
1	<=50K	ditolak
2	<=50K	ditolak
3	<=50K	ditolak
4	<=50K	ditolak

2. Menerapkan encoder Label pada variabel kategorikal untuk mengubah nilai-nilai menjadi representasi numerik, memungkinkan penggunaan variabel tersebut dalam membangun model pohon keputusan.

```
le = preprocessing.LabelEncoder()
dataset_categorical = dataset_categorical.apply(le.fit_transform)
dataset_categorical.head()
```

	Class	cluster
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1

3. Menggabungkan dataset yang telah dikodekan numerik dari variabel kategorikal dengan dataset asli, mempersiapkan data untuk pembangunan model pohon keputusan.

```
dataset = dataset.drop(dataset_categorical.columns, axis=1)
dataset = pd.concat([dataset, dataset_categorical], axis=1)
dataset.head()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Class	cluster
0	90	1	77053	1	9	1	5	1	1	1	0	4356	40	1	0	1
1	82	1	132870	1	9	1	1	1	1	1	0	4356	18	1	0	1
2	66	1	186061	2	10	1	5	2	2	1	0	4356	40	1	0	1
3	54	1	140359	3	4	2	2	2	1	1	0	3900	40	1	0	1
4	41	1	264663	2	10	3	3	3	1	1	0	3900	40	1	0	1

4. Output dari `dataset.info()` menunjukkan informasi tentang dataset, termasuk jumlah entri, jenis data, dan jumlah nilai non-null untuk setiap kolom, yang penting untuk memastikan keseluruhan kualitas dan konsistensi data sebelum membangun model.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   A1          32561 non-null  int64  
 1   A2          32561 non-null  int64  
 2   A3          32561 non-null  int64  
 3   A4          32561 non-null  int64  
 4   A5          32561 non-null  int64  
 5   A6          32561 non-null  int64  
 6   A7          32561 non-null  int64  
 7   A8          32561 non-null  int64  
 8   A9          32561 non-null  int64  
 9   A10         32561 non-null  int64  
10  A11         32561 non-null  int64  
11  A12         32561 non-null  int64  
12  A13         32561 non-null  int64  
13  A14         32561 non-null  int64  
14  Class       32561 non-null  int64  
15  cluster     32561 non-null  int64  
dtypes: int64(16)
memory usage: 4.0 MB
```

5. Perubahan tipe data pada kolom-kolom tertentu ke tipe data kategorikal dapat membantu memastikan bahwa variabel yang sesuai dengan kategori atau label tidak dianggap sebagai variabel numerik, yang mungkin menyebabkan kesalahan dalam interpretasi dan pemrosesan model.

```
[ ] dataset[['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'A11',
            'A12', 'A13', 'A14', 'Class']] = dataset[['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'A11',
            'A12', 'A13', 'A14', 'Class']].astype('category')

Sekarang semua variabel kategorikal telah dikodekan dengan tepat. Mari kita bangun modelnya.
```

Pada tahap ini, saya memulai proses pembangunan model dengan membuat pohon keputusan menggunakan hiperparameter default. Langkah ini memberikan kita gambaran awal tentang performa model tanpa penyetelan tambahan. Selanjutnya, saya akan melibatkan teknik validasi silang untuk menyetel hiperparameter pohon keputusan agar sesuai dengan data pelatihan saya. Proses ini membantu meningkatkan kinerja model dan mengoptimalkan keakuratannya. Evaluasi model dilakukan dengan mengukur sejumlah metrik kinerja yang relevan, memungkinkan kita untuk memahami sejauh mana model kita berhasil dalam memprediksi target kelas pada dataset yang diberikan. Hasil evaluasi ini menjadi dasar untuk pengambilan keputusan lebih lanjut terkait peningkatan model dan pengoptimalkan parameter.

6. Gunakan library **train_test_split** dari scikit-learn untuk membagi dataset menjadi dua bagian, yaitu data pelatihan (training) dan data uji (testing), memungkinkan kita untuk mengevaluasi kinerja model pada data yang tidak pernah dilihat selama proses pelatihan.

```
from sklearn.model_selection import train_test_split
```

7. Memisahkan dataset menjadi variabel fitur (X) dan variabel target (y) dengan cara menghilangkan kolom-kolom yang tidak diperlukan dari dataset. Variabel fitur (X) akan berisi atribut-atribut yang digunakan untuk membuat prediksi, sedangkan variabel target (y) akan berisi label atau nilai yang ingin diprediksi.

```
columns_to_drop = ['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'A11', 'A12', 'A13', 'A14', 'Class']
X = dataset.drop(labels=columns_to_drop, axis=1)

y = dataset['Class']
```

8. Memisahkan data menjadi data latih dan data uji menggunakan fungsi **train_test_split** dari scikit-learn. Data latih (X_train, y_train) digunakan untuk melatih model, sedangkan data uji (X_test, y_test) digunakan untuk menguji performa model.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.30,
                                                    random_state = 99)

X_train.head()
```

	cluster
5728	0
10700	1
29425	1
2088	0
16292	1

9. Mengimpor pengklasifikasi pohon keputusan dari pustaka scikit-learn dan memasang pohon keputusan dengan hyperparameter default, kecuali **max_depth** yang diatur menjadi 5 agar dapat melakukan plotting dan pembacaan pohon.

```
from sklearn.tree import DecisionTreeClassifier

dt_default = DecisionTreeClassifier(max_depth=5)
dt_default.fit(X_train, y_train)
```

DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)

10. Mari kita periksa metrik evaluasi dari model default kita. Pertama, kita mengimpor laporan klasifikasi dan matriks kebingungan dari metrik scikit-learn. Selanjutnya, kita membuat prediksi menggunakan model default dan mencetak laporan klasifikasi.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

y_pred_default = dt_default.predict(X_test)
print(classification_report(y_test, y_pred_default))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7475
1	1.00	1.00	1.00	2294
accuracy			1.00	9769
macro avg	1.00	1.00	1.00	9769
weighted avg	1.00	1.00	1.00	9769

11. Mencetak matriks kekeliruan dan akurasi dari model default:

```
print(confusion_matrix(y_test, y_pred_default))
print(accuracy_score(y_test, y_pred_default))
```

[[7475 0]
 [0 2294]]
1.0

12. Mengimpor package yang diperlukan untuk visualisasi dan menempatkan fitur:


```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydot, graphviz

features = list(dataset.columns[1:])
features
```

```
['A2',
 'A3',
 'A4',
 'A5',
 'A6',
 'A7',
 'A8',
 'A9',
 'A10',
 'A11',
 'A12',
 'A13',
 'A14',
 'Class',
 'cluster']
```

13. Visualisasi pohon keputusan telah dibuat menggunakan model default dengan kedalaman maksimum 5, menampilkan fitur dan struktur keputusan secara grafis.

```
from sklearn.tree import export_graphviz
from IPython.display import Image
import pydotplus

# Specify the feature names based on the columns you used in X_train
features = X_train.columns

# Export the decision tree visualization
dot_data = export_graphviz(dt_default, out_file=None,
                           feature_names=features, filled=True, rounded=True)

# Create a graph from dot_data
graph = pydotplus.graph_from_dot_data(dot_data)

# Display the decision tree visualization
Image(graph.create_png())
```

```
graph TD
    A["cluster <= 0.5  
gini = 0.368  
samples = 22792  
value = [17245, 5547]"] -- True --> B["gini = 0.0  
samples = 5547  
value = [0, 5547]"]
    A -- False --> C["gini = 0.0  
samples = 17245  
value = [17245, 0]"]
```

14. Visualisasi pohon keputusan dengan kedalaman maksimum 3 telah dibuat, memperjelas struktur dan keputusan yang diambil oleh model.

```
# memplot pohon dengan max_depth=3
dot_data = StringIO()
export_graphviz(dt_default, out_file=dot_data,
               feature_names=features, filled=True, rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```

```
graph TD
    A["cluster <= 0.5  
gini = 0.368  
samples = 22792  
value = [17245, 5547]"] -- True --> B["gini = 0.0  
samples = 5547  
value = [0, 5547]"]
    A -- False --> C["gini = 0.0  
samples = 17245  
value = [17245, 0]"]
```

15. Kita menggunakan GridSearchCV untuk menemukan kedalaman maksimal yang optimal untuk pohon keputusan. Kedalaman yang terbaik akan digunakan untuk membangun model akhir.

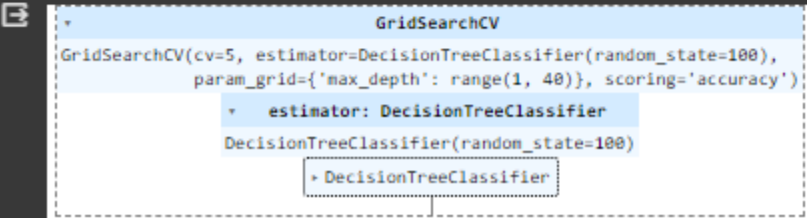
```
# GridSearchCV untuk menemukan kedalaman maksimal yang optimal
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

# tentukan jumlah lipatan untuk CV k-lipatan
n_folds = 5

# parameter untuk membangun model
parameters = {'max_depth': range(1, 40)}

# instantiate model
dtree = DecisionTreeClassifier(criterion = "gini",
                               random_state = 100)

# cocokkan pohon pada data pelatihan
tree = GridSearchCV(dtree, parameters,
                    cv=n_folds,
                    scoring="accuracy")
tree.fit(X_train, y_train)
```

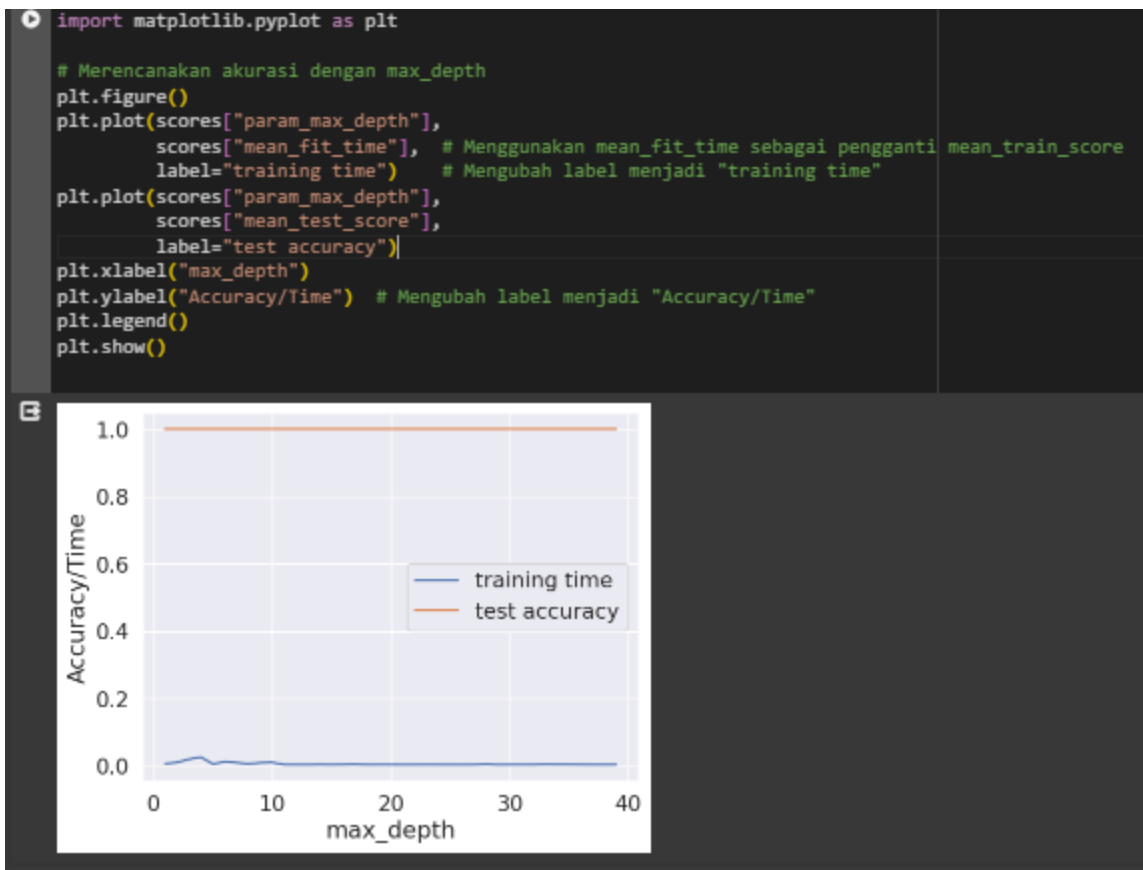


16. Hasil dari GridSearchCV, termasuk skor validasi silang (CV scores) untuk setiap nilai kedalaman pohon yang diuji. Skor CV dapat membantu kita memilih kedalaman pohon yang optimal.

```
# skor CV GridSearch
scores = tree.cv_results_
pd.DataFrame(scores).head()
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	0.009002	0.000678	0.003762	0.000606	1	{'max_depth': 1}	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
1	0.010562	0.003788	0.00647	0.005215	2	{'max_depth': 2}	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
2	0.020149	0.004189	0.00668	0.003017	3	{'max_depth': 3}	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
3	0.026165	0.010313	0.010142	0.005271	4	{'max_depth': 4}	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
4	0.006079	0.000157	0.003737	0.000811	5	{'max_depth': 5}	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1

17. Grafik menunjukkan hubungan antara nilai **max_depth** dengan waktu pelatihan dan akurasi pengujian pada model pohon keputusan, membantu pemilihan nilai hiperparameter yang optimal.



18. Skor CV GridSearch untuk kedalaman maksimal menunjukkan hubungan antara jumlah minimal sampel di daun dan akurasi pengujian, membantu menentukan nilai hiperparameter optimal untuk model pohon keputusan.

```

# GridSearchCV untuk menemukan kedalaman maksimal yang optimal
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

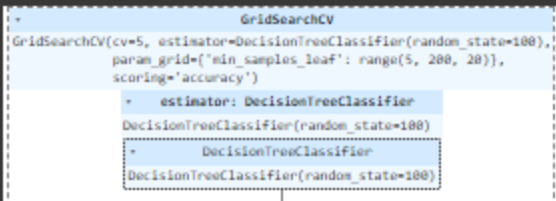
# tentukan jumlah lipatan untuk CV k-lipatan
n_folds = 5

# parameter untuk membangun model
parameters = {'min_samples_leaf': range(5, 200, 20)}

# instantiate model
dtree = DecisionTreeClassifier(criterion = "gini",
                              random_state = 100)

# cocokkan pohon pada data pelatihan
tree = GridSearchCV(dtree, parameters,
                    cv=n_folds,
                    scoring="accuracy")
tree.fit(X_train, y_train)

```



```

+-----+
| GridSearchCV                               |
| GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=100), |
|   param_grid=[{'min_samples_leaf': range(5, 200, 20)},               |
|   scoring='accuracy'])                                               |
| +-----+ +-----+ +-----+ +-----+ +-----+ +-----+         |
| | estimator: DecisionTreeClassifier | | estimator: DecisionTreeClassifier | | estimator: DecisionTreeClassifier | | estimator: DecisionTreeClassifier | | estimator: DecisionTreeClassifier | | estimator: DecisionTreeClassifier |
| | DecisionTreeClassifier(random_state=100) | | DecisionTreeClassifier(random_state=100) | | DecisionTreeClassifier(random_state=100) | | DecisionTreeClassifier(random_state=100) | | DecisionTreeClassifier(random_state=100) | | DecisionTreeClassifier(random_state=100) |
| +-----+ +-----+ +-----+ +-----+ +-----+ +-----+         |
+-----+

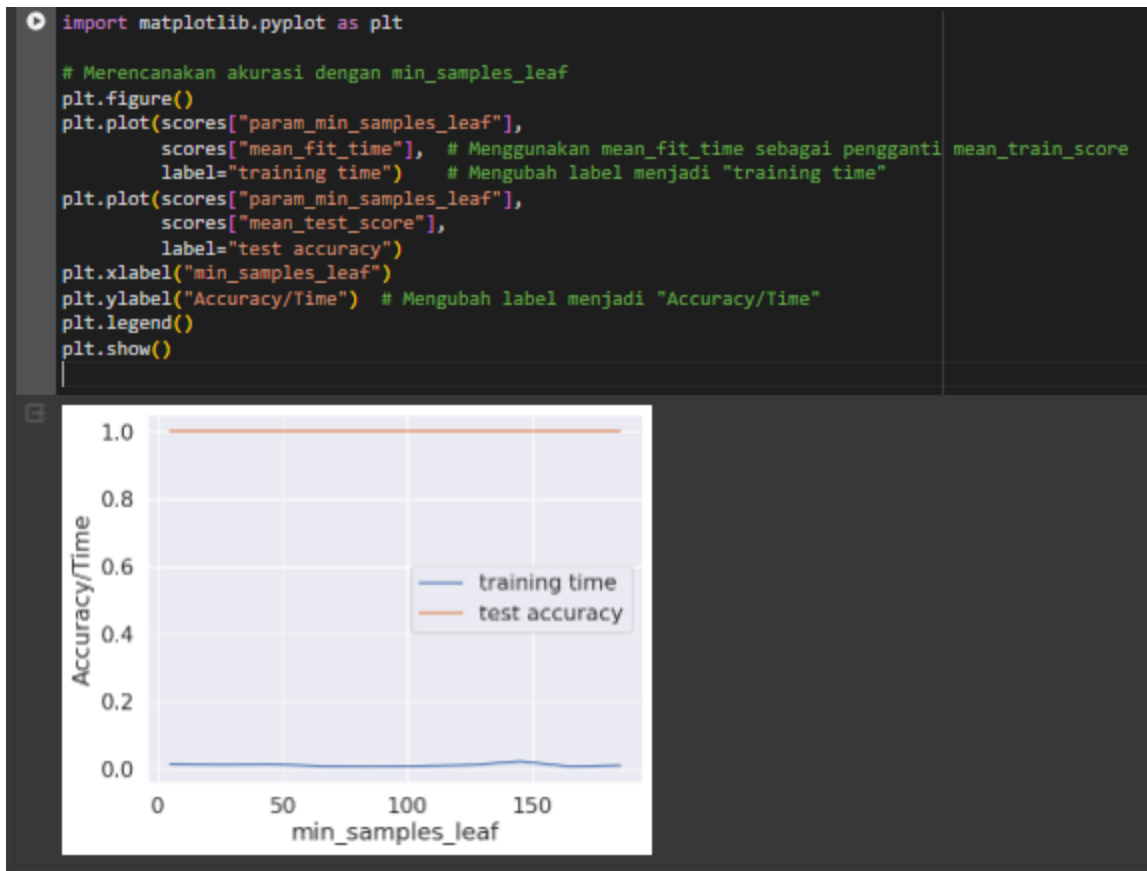
```

19. Berikut adalah beberapa nilai skor yang dihasilkan dari GridSearchCV, yang mencakup parameter `min_samples_leaf` dan hasil pengujian yang relevan, membantu dalam mengevaluasi dan memilih hiperparameter yang optimal untuk membangun model pohon keputusan.

```
# skor CV GridSearch
scores = tree.cv_results_
pd.DataFrame(scores).head()
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_leaf	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score	
0	0.013852	0.000125	0.007410	0.003658	5	(min_samples_leaf: 5)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
1	0.011840	0.002470	0.005379	0.003329	25	(min_samples_leaf: 25)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
2	0.013111	0.003765	0.007820	0.002808	45	(min_samples_leaf: 45)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
3	0.007462	0.001570	0.003965	0.000103	65	(min_samples_leaf: 65)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
4	0.006484	0.000137	0.004100	0.000545	85	(min_samples_leaf: 85)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1

20. Berikut adalah grafik yang mencerminkan akurasi dan waktu pelatihan dengan variasi nilai `min_samples_leaf` pada model pohon keputusan. Grafik ini membantu dalam mengevaluasi kinerja model dengan mengamati dampak perubahan hiperparameter tertentu terhadap akurasi dan waktu pelatihan.



21. Setelah menjalankan GridSearchCV dengan variasi nilai `min_samples_split` pada model pohon keputusan, kami dapat mengevaluasi kinerja model dengan melihat grafik akurasi dan waktu pelatihan untuk setiap nilai `min_samples_split`. Grafik ini memberikan pandangan yang lebih komprehensif tentang bagaimana model bereaksi terhadap perubahan hiperparameter ini.

```
# GridSearchCV untuk menemukan min_samples_split yang optimal
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV

# tentukan jumlah lipatan untuk CV k-lipatan
n_folds = 5

# parameter untuk membangun model
parameters = {'min_samples_split': range(5, 200, 20)}

# instantiate model
dtree = DecisionTreeClassifier(criterion = "gini",
                              random_state = 100)

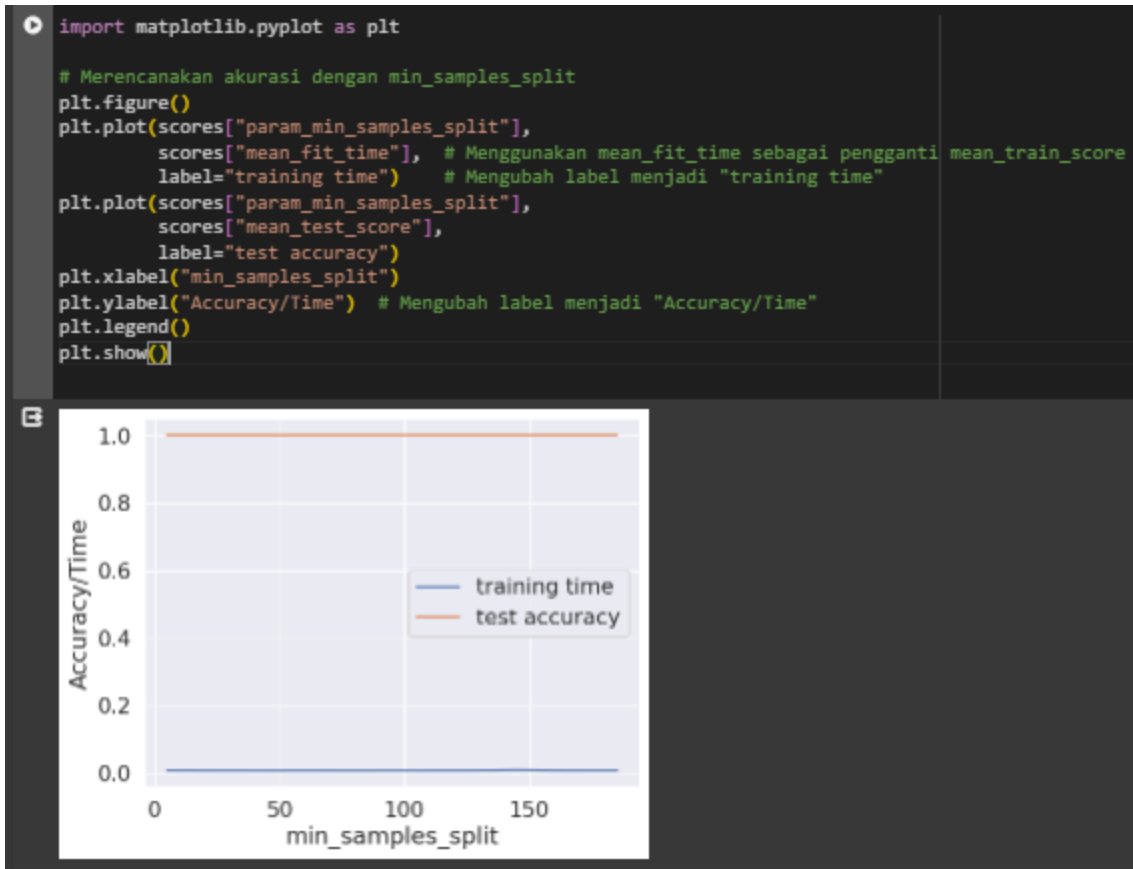
# cocokkan pohon pada data pelatihan
tree = GridSearchCV(dtree, parameters,
                    cv=n_folds,
                    scoring="accuracy")
tree.fit(X_train, y_train)
```

22. Data frame di atas menunjukkan skor Cross-Validation untuk setiap kombinasi nilai `min_samples_split` yang diuji dalam `GridSearchCV`, termasuk parameter seperti waktu pelatihan dan skor uji rata-rata. Dengan menganalisis data frame ini, kita dapat memilih nilai `min_samples_split` yang optimal untuk meningkatkan kinerja model pohon keputusan.

```
# skor CV GridSearch
scores = tree.cv_results_
pd.DataFrame(scores).head(5)
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_split	param_split0_test_score	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	0.007478	0.000758	0.004328	0.000519	5 ('min_samples_split': 5)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
1	0.006548	0.000124	0.003774	0.000266	25 ('min_samples_split': 25)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
2	0.006512	0.000261	0.003874	0.000113	45 ('min_samples_split': 45)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
3	0.006685	0.000228	0.003909	0.000046	65 ('min_samples_split': 65)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
4	0.006783	0.000084	0.003817	0.000165	85 ('min_samples_split': 85)	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1

23. Grafik di bawah menunjukkan hubungan antara waktu pelatihan dan akurasi uji rata-rata terhadap nilai `min_samples_split` yang diuji dalam `GridSearchCV`. Analisis grafik ini dapat membantu kita memilih nilai `min_samples_split` yang optimal untuk memperbaiki kinerja model pohon keputusan.



24. GridSearchCV digunakan untuk mencari kombinasi hyperparameter yang optimal dalam model Decision Tree. Parameter yang diuji melibatkan kedalaman maksimal pohon (**max_depth**), jumlah minimum sampel di daun (**min_samples_leaf**), jumlah minimum sampel yang diperlukan untuk membagi simpul internal (**min_samples_split**), dan kriteria pemilihan fitur terbaik (**criterion**). Proses ini membantu menentukan kombinasi hyperparameter yang memberikan kinerja terbaik pada data pelatihan dan meminimalkan risiko overfitting atau underfitting.


```
# Membuat grid parameter
param_grid = {
    'max_depth': range(5, 15, 5),
    'min_samples_leaf': range(50, 150, 50),
    'min_samples_split': range(50, 150, 50),
    'criterion': ["entropy", "gini"]
}

n_folds = 5

# Menginstansiasi model pencarian grid
dtree = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator = dtree, param_grid = param_grid,
                           cv = n_folds, verbose = 1)

# Sesuaikan pencarian grid dengan data
grid_search.fit(X_train,y_train)
```

Fitting 5 folds for each of 16 candidates, totalling 80 fits

```
GridSearchCV
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['entropy', 'gini'],
                         'max_depth': range(5, 15, 5),
                         'min_samples_leaf': range(50, 150, 50),
                         'min_samples_split': range(50, 150, 50)},
             verbose=1)
- estimator: DecisionTreeClassifier
  DecisionTreeClassifier()
    - DecisionTreeClassifier
```

25. DataFrame **cv_results** menyajikan hasil dari GridSearchCV yang mencakup kombinasi hyperparameter yang diuji beserta skor akurasi dan informasi lainnya pada setiap lipatan cross-validation. Data ini membantu untuk memahami bagaimana performa model Decision Tree bervariasi dengan setiap kombinasi hyperparameter yang diuji, memandu pemilihan hyperparameter yang optimal untuk model.

```
# Hasil cv
cv_results = pd.DataFrame(grid_search.cv_results_)
cv_results
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_depth	param_min_samples_leaf	param_min_samples_split	param__	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	0.01010	0.00731	0.01030	0.00415	entropy	5	50	50	(criterion='entropy', max_depth=5, min...	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
1	0.00762	0.00468	0.00804	0.00282	entropy	5	50	100	(criterion='entropy', max_depth=5, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
2	0.00680	0.001028	0.00762	0.00170	entropy	5	100	50	(criterion='entropy', max_depth=5, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
3	0.00681	0.00193	0.00470	0.00126	entropy	5	100	100	(criterion='entropy', max_depth=5, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
4	0.006747	0.000794	0.00372	0.00111	entropy	10	50	50	(criterion='entropy', max_depth=10, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
5	0.00610	0.00077	0.00304	0.00080	entropy	10	50	100	(criterion='entropy', max_depth=10, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
6	0.00615	0.000746	0.00304	0.00048	entropy	10	100	50	(criterion='entropy', max_depth=10, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
7	0.00670	0.001146	0.004348	0.001275	entropy	10	100	100	(criterion='entropy', max_depth=10, min...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
8	0.006137	0.000170	0.003759	0.00063	gini	5	50	50	(criterion='gini', max_depth=5, min_sam...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
9	0.006343	0.000147	0.003849	0.00033	gini	5	50	100	(criterion='gini', max_depth=5, min_sam...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
10	0.006280	0.000253	0.003844	0.000254	gini	5	100	50	(criterion='gini', max_depth=5, min_sam...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
11	0.007021	0.000507	0.004622	0.00075	gini	5	100	100	(criterion='gini', max_depth=5, min_sam...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
12	0.006025	0.000332	0.006775	0.00281	gini	10	50	50	(criterion='gini', max_depth=10, min_sa...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
13	0.006807	0.000388	0.003874	0.00121	gini	10	50	100	(criterion='gini', max_depth=10, min_sa...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
14	0.006301	0.000335	0.003960	0.00038	gini	10	100	50	(criterion='gini', max_depth=10, min_sa...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1
15	0.006188	0.000111	0.003880	0.00041	gini	10	100	100	(criterion='gini', max_depth=10, min_sa...	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1

26. Print ini menunjukkan skor akurasi terbaik yang ditemukan oleh GridSearchCV dan kombinasi hyperparameter yang memberikan skor tersebut. Informasi ini berguna untuk memahami kinerja model yang dihasilkan dengan hyperparameter yang dioptimalkan.

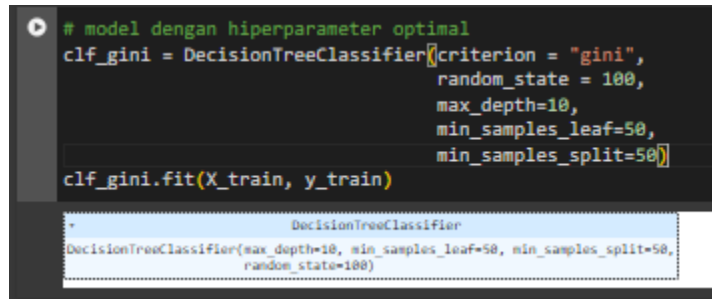
```
# mencetak skor akurasi dan hiperparameter yang optimal
print("best accuracy", grid_search.best_score_)
print(grid_search.best_estimator_)

best accuracy 1.0
DecisionTreeClassifier(criterion='entropy', max_depth=5, min_samples_leaf=50,
                       min_samples_split=50)
```

27. Langkah ini melibatkan pembuatan model dengan hyperparameter yang dioptimalkan. Pohon keputusan diinisialisasi dengan hyperparameter terbaik yang ditemukan oleh GridSearchCV, dan kemudian dipasang pada data pelatihan. Model yang dihasilkan dengan hyperparameter yang dioptimalkan ini dapat digunakan untuk melakukan prediksi pada data uji.

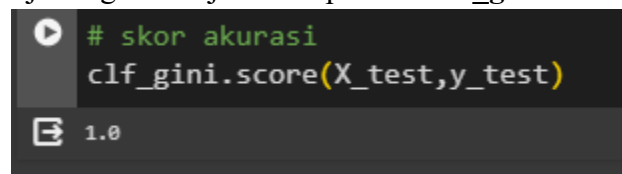
```
# model dengan hiperparameter optimal
clf_gini = DecisionTreeClassifier(criterion = "gini",
                                random_state = 100,
                                max_depth=10,
                                min_samples_leaf=50,
                                min_samples_split=50)

clf_gini.fit(X_train, y_train)
```



28. Akurasi dari model pohon keputusan yang telah dibangun dengan hyperparameter optimal adalah nilai yang diperoleh pada data uji. Dalam hal ini, Anda dapat melihat nilai akurasi pada dataset uji dengan menjalankan perintah `clf_gini.score(X_test, y_test)`.

```
# skor akurasi
clf_gini.score(X_test, y_test)
```

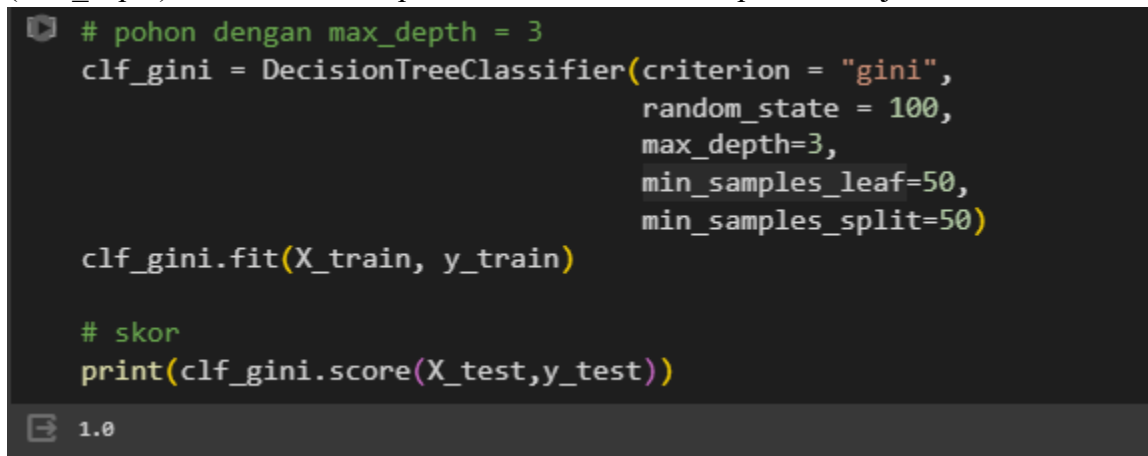


29. Dengan menggunakan pohon keputusan yang memiliki kedalaman maksimal (`max_depth`) sebesar 3, kita dapat melihat akurasi model pada data uji.

```
# pohon dengan max_depth = 3
clf_gini = DecisionTreeClassifier(criterion = "gini",
                                random_state = 100,
                                max_depth=3,
                                min_samples_leaf=50,
                                min_samples_split=50)

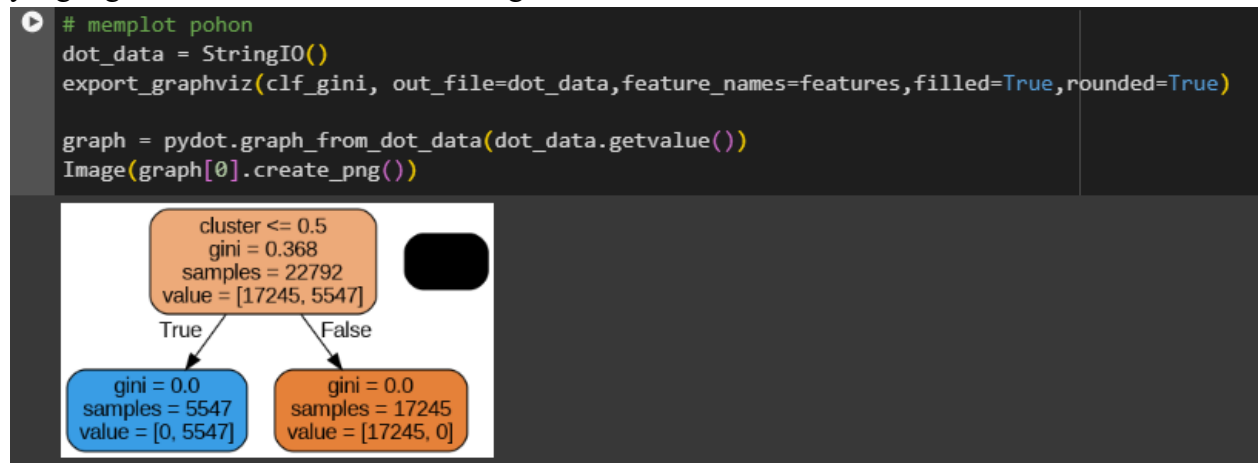
clf_gini.fit(X_train, y_train)

# skor
print(clf_gini.score(X_test, y_test))
```



30. Berikut adalah visualisasi dari pohon keputusan yang telah dibangun dengan menggunakan hyperparameter optimal. Pohon ini memperlihatkan struktur keputusan

yang digunakan oleh model untuk mengklasifikasikan data.



31. Pada model pohon keputusan dengan hyperparameter optimal, kita mendapatkan hasil metrik klasifikasi seperti yang dicetak oleh fungsi **classification_report** pada data uji.

```
[ ] # metrik klasifikasi
from sklearn.metrics import classification_report, confusion_matrix
y_pred = clf_gini.predict(X_test)
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7475
1	1.00	1.00	1.00	2294
accuracy			1.00	9769
macro avg	1.00	1.00	1.00	9769
weighted avg	1.00	1.00	1.00	9769

32. Matriks kebingungan (confusion matrix) menggambarkan hasil klasifikasi pada model pohon keputusan dengan hyperparameter optimal terhadap data uji.

```
# matriks yang membingungkan
print(confusion_matrix(y_test, y_pred))
```

```
[[7475  0]
 [ 0 2294]]
```