Mari Me-recall

Untuk membaca dan menyimpan file, berikut metod dalam pandas :

Data Format	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
sql	pd.read_sql()	df.to_sql()

berikut type data pandas di bandingkan dengan python native

Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	time data.

Preprocessing

Praproses data merupakan tahapan yang sangat penting dan salah satu faktor yang menentukan keberhasilan dari pemodelan machine learning. Pada modul ini akan dibahas mengenai imputasi data serta normalisasi data yang merupakan tahapan dalam praproses.

Tujuan : Kegiatan belajar ini memberikan pengetahuan kepada mahasiswa tentang proses dan teknik praproses data dalam hal ini adalah imputasi missing value dan transformasi/normalisasi data.

Pada praktikum ini kita akan menggunakan dataset hepartitis yang dapat di akses di https://archive.ics.uci.edu/ml/datasets/hepatitis. kita akan mengimpor data dan mempraproses data tsb sehingga data berada dalam format yang sesuai dan siap untuk di analisis/di mining.

NOTE: PENGGUNAAN DATA DAN TEKNIK DALAM PRAPROSES INI HANYA UNTUK KEPENTINGAN PEMBELAJARAN.

```
import pandas as pd
csv_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis
```

```
csv_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.
```

```
#berikan header (nama kolom)
headers=['Class', 'AGE', 'SEX', 'STEROID','ANTIVIRALS', 'FATIGUE', 'MALAISE','ANOREXIA', '
```

Gunakan metod pandas **read_csv()** untuk me-load data dari alamat web. Set parameter "names" sama dengan "headers".

```
df = pd.read_csv(csv_path, names = headers)
```

Memahami Data

Sebelum melakukan analisis atau pemodelan, sangat penting untuk memahami data yang kita gunakan.

Yang harus kita cek :

- type data
- distribusi data

Ketahui issue potensial yang ada pada data.

Distribusi Data

Berikutnya setelah mengecek type data, kita juga perlu mengecek ringkasan statistik setiap kolom untuk mempelajari distrubusi data dari setiap kolom. metric statistikal dapat memberitahukan kepada kita jika terdapat mathematical issue yang mungkin ada seperti outlier ekstrim dan deviasi yang besar yang mungkin akan perlu kita tangani nanti.

untuk memperoleh distribusi statistik, kita mengguanakn metod `describe` yang memberikan nilai-nilai seperti : "count", "mean", standard deviation --> "std", nilai maximum dan minimum, dan boundary setiap quartile. Secara default, fungsi `dataframe.describe()` men-skips baris dan kolom yang tidak berisi angka.

lihatlah deskripsi dari dataset dengan perintah `describe()`

```
df.describe()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAI
count	155.000000	155.000000	155.000000	155.000000	155.000000	155.000000	155.000(
mean	1.793548	0.481690	1.103226	1.509677	1.845161	1.348387	1.6064
std	0.406070	0.176984	0.305240	0.501527	0.362923	0.478004	0.490'

terlihat hanya 5 kolom yang keluar pada deskripsi, padahal deskripsi ini untuk deskripsi numerik. teman-teman perhatiakn kembali kolom mana yang non-numerik?

Mengapa harus memeriksa type data?

- berpotensi berisi info dan type yang tidak tepat

> Pandas secara otomatis meng-assign type berdasarkan encoding yang di deteksi dari tabel awal. dan terdapat benyak kemungkinan assignment nya salah. contoh : price (harga) mobil yang sedianya kita berekpektasi harusnya bertipe numerik, oleh pandas di deteksi sebagai object (string dalam python). akan lebih baik jika typenya di ganti dengan float.

- kompatibilitas dengan metod di python

> kolom tertentu mungkin memerlukan operasi/fungsi tertentu. Misal, sebagian fungsi matematis hanya dapat di aplikasikan ke data numerik, dan jika di aplikasikan ke type data lain akan terjadi error.

Kita dapat mengecek type data dari setiapkolom dengan metod `dtypes`.

```
df.dtypes
```

```
Class              int64
AGE              float64
SEX                int64
STEROID            int64
ANTIVIRALS         int64
FATIGUE            int64
MALAISE            int64
ANOREXIA           int64
LIVER BIG          int64
LIVER FIRM         int64
SPLEEN ALPABLE     int64
SPIDERS            int64
ASCITES            int64
VARICES            int64
BILIRUBIN        float64
ALK PHOSPHATE    float64
SGOT             float64
ALBUMIN          float64
PROTIME          float64
```

```
HISTOLOGY           int64
dtype: object
```

Untuk meng-enable deskripsi dari semua kolom kita tambahkan parameter `include="all"` dalam metod `describe` hasilnya menunjukkan ringkasan dari semua kolom termasuk kolom yang bertype object. Untuk kolom type `object`, ringaksan statistik yang di evaluasi seperti unique, top dan frequency. `Unique` adalah jumlah object yang berbeda dalam kolom. is the number of distinct objects in the column, `top` berupa object yang paling sering muncul, `freq` berisi jumlah kemunculan top object di dalam kolom. Selain itu, terdapat nilai `NaN` atau "not a number". hal ini terjaid karena perhitungan statistik tertentu tidak dapat dilakukan untuk kolom dengan type data tertentu.

```
df.describe(include="all")
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAI
count	155.000000	155.000000	155.000000	155.000000	155.000000	155.000000	155.000(
mean	1.793548	0.481690	1.103226	1.509677	1.845161	1.348387	1.6064
std	0.406070	0.176984	0.305240	0.501527	0.362923	0.478004	0.490
min	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000(
25%	2.000000	0.352113	1.000000	1.000000	2.000000	1.000000	1.000(
50%	2.000000	0.450704	1.000000	2.000000	2.000000	1.000000	2.000(
75%	2.000000	0.605634	1.000000	2.000000	2.000000	2.000000	2.000(
max	2.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.000(

terlihat dari type data tiap kolom, terdapat banyak object (string atau campuran). padahal dari penjelasan dataset kita mengharapkan semuanya berupa numerik (int atau float).. mari kita cek isi dataframe dengan head()

```
# Cek hasil import dan pemberian header
df.head()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVE FIR
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	

Sebagaimana kita lihat, beberapa tanda "?" muncul pada dataframe, ini adalah suatu missing value, yang mungkin akan menghalangi analisis berikutnya.

Lalu, bagaimana kita mengidentifikasi semua missing value dan mengatasinya?

Bagaiaman bekerja dengan missing value?

Langkah-langkah :

1. Mengidentifikasi missing data
2. menangani missing value
3. Perbaiki format data

Mengidentifikasi dan menghandle missing value (Data Cleaning)

Identifikasi missing value

Mengkonversi "?" ke NaN

Pada dataset hepatitis, missing value ditandai dengan tanda "?". Kita me-replace "?" dengan NaN (Not a Number), yang merupakan penanda missing value default di Phyton, untuk alasan kecepatan komputasi dan kenyamanan. Disini kita akan menggunakan fungsi :

```
 .replace(A, B, inplace = True)
```

untuk me-replace A dengan B

```
import numpy as np

# replace "?" to NaN
df.replace("?", np.nan, inplace = True)
df.head(5)
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVE FIR
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	

Mengidentifikasi Missing value

Missing value dikonversi ke niali default python, kemudian kita gunakan fungsi built in python untuk mengidentifikasi. terdapat 2 metode untuk mendeteksi missing data :

1. **.isnull()**
2. **.notnull()**

```
missing_data = df.isnull()
missing_data.head()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVER FIRM
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False

```
missing_data2 = df.notnull()
missing_data2.head()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVER FIRM
0	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True

Menghitung missing values setiap kolom

Menggunakan looping, kita dapat menghitung jumlah missing value dengan cepat untuk setiap kolom. karena "true" berarti missing, "false" berarti Tidak Missing, kita gunakan method ".value_counts()" untuk menghitung jumlah "True" dan "false".

```
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

```
#metode ini hanya untuk menghitung true (jml missing value saja)
```

```
df.isnull().sum()
```

```
Class             0
AGE               0
SEX               0
STEROID           0
ANTIVIRALS        0
FATIGUE           0
MALAISE           0
ANOREXIA          0
LIVER BIG         0
LIVER FIRM        0
SPLEEN ALPABLE    0
SPIDERS           0
ASCITES           0
VARICES           0
BILIRUBIN         0
ALK PHOSPHATE     0
SGOT              0
ALBUMIN           0
PROTIME           0
HISTOLOGY         0
dtype: int64
```

Menangani Missing value

Bagaimana menangani Missing Value?

1. drop data
   a. drop seluruh baris
   b. drop seluruh kolom
2. replace data
   a. replace dengan mean
   b. replace dengan frequency
   c. replace dengan fungsi lain
3. Biarkan

```
df.head()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVE FIR
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	

seluruh kolom di drop hanya jika hampir seluruh isi kolom tersebut missing. dalam dataset kita tidak ada kolom dengan karakteristik tsb. Beberapa teknik replace data:

Replace dengan mean:

BILIRUBIN 6
ALK PHOSPHATE 29
SGOT 4
ALBUMIN 16
PROTIME 67

Replace dengan frequency (modus/yang paling sering muncul): misal fitur dengan tipe kategorikal/boolean

STEROID 1
FATIGUE 1
MALAISE 1
ANOREXIA 1
LIVER BIG 10
LIVER FIRM 11
SPLEEN ALPABLE 5
SPIDERS 5
ASCITES 5

Hitung rata-rata PROTIME

```
avg_protime = df["PROTIME"].astype("float").mean(axis=0)
print("rata-rata protime:", avg_protime)
```

```
rata-rata protime: 61.85227272727272
```

Replace "NaN" dengan nilai mean pada kolom "PROTIME"

```
df["PROTIME"].replace(np.nan, avg_protime, inplace=True)
```

```
df.head()
```

Begitu pula kita replace dengan rata-rata untuk BILIRUBIN, ALK PHOSPHATE, SGOT dan ALBUMIN

```
avg_alk = df["ALK PHOSPHATE"].astype("float").mean(axis=0)
df["ALK PHOSPHATE"].replace(np.nan, avg_alk, inplace=True)
```

```
avg_bili = df["BILIRUBIN"].astype("float").mean(axis=0)
df["BILIRUBIN"].replace(np.nan, avg_bili, inplace=True)
```

```
avg_sgot = df["SGOT"].astype("float").mean(axis=0)
df["SGOT"].replace(np.nan, avg_sgot, inplace=True)
```

```
avg_albu = df["ALBUMIN"].astype("float").mean(axis=0)
df["ALBUMIN"].replace(np.nan, avg_albu, inplace=True)
```

```
avg_sgot
```

```
0.11339753901435232
```

Replace dengan Frekuensi untuk fitur STEROID

untuk melihat nilai mana yang paling sering muncul dalam suatu kolom, kita dapat menggunakan metod ".value_counts()" :

```
df['SPIDERS'].value_counts()
```

```
2    104
1     51
Name: SPIDERS, dtype: int64
```

```
df['STEROID'].value_counts()
```

```
2    79
1    76
Name: STEROID, dtype: int64
```

kita juga dapat menggunakan ".idxmax()" untuk menghitung type mana yang paling banyak secara otomatis:

```
df['STEROID'].value_counts().idxmax()
```

```
2
```

Sehungga kita akan mereplace missing value pada kolom STEROID dengan '2'

```
df["STEROID"].replace(np.nan, "2", inplace=True)
```

Pelakuan yang sama untuk variable FATIGUE, MALAISE, ANOREXIA, LIVER BIG, LIVER FIRM, SPLEEN ALPABLE, SPIDERS, ASCITES, VARICES.{selesaikan ya}

```
df['FATIGUE'].value_counts().idxmax()
```

```
1
```

```
df["FATIGUE"].replace(np.nan, "1", inplace=True)
```

```
VAL_MAL=df['MALAISE'].value_counts().idxmax()
df["MALAISE"].replace(np.nan, VAL_MAL, inplace=True)
```

```
VAL_ANOREXIA=df['ANOREXIA'].value_counts().idxmax()
df["ANOREXIA"].replace(np.nan, VAL_ANOREXIA, inplace=True)
```

```
VAL_LIVER_BIG=df['LIVER BIG'].value_counts().idxmax()
df["LIVER BIG"].replace(np.nan, VAL_LIVER_BIG, inplace=True)
```

```
VAL_LIVER_FIRM=df['LIVER FIRM'].value_counts().idxmax()
df["LIVER FIRM"].replace(np.nan, VAL_LIVER_FIRM, inplace=True)
```

```
VAL_SPLEEN_ALPABLE=df['SPLEEN ALPABLE'].value_counts().idxmax()
df["SPLEEN ALPABLE"].replace(np.nan, VAL_SPLEEN_ALPABLE, inplace=True)
```

```
VAL_SPIDERS=df['SPIDERS'].value_counts().idxmax()
df["SPIDERS"].replace(np.nan, VAL_SPIDERS, inplace=True)
```

```
VAL_ASCITES=df['ASCITES'].value_counts().idxmax()
df["ASCITES"].replace(np.nan, VAL_ASCITES, inplace=True)
```

```
VAL_VARICES=df['VARICES'].value_counts().idxmax()
df["VARICES"].replace(np.nan, VAL_VARICES, inplace=True)
```

Mari kita cek, apakah masih ada missing value

```
df.isnull().sum()
```

```
Class             0
AGE               0
SEX               0
STEROID           0
ANTIVIRALS        0
FATIGUE           0
MALAISE           0
ANOREXIA          0
```

```
LIVER BIG         0
LIVER FIRM        0
SPLEEN ALPABLE    0
SPIDERS           0
ASCITES           0
VARICES           0
BILIRUBIN         0
ALK PHOSPHATE     0
SGOT              0
ALBUMIN           0
PROTIME           0
HISTOLOGY         0
dtype: int64
```

Perbaiki Format data

Langkah terakhir kita adalah mengecek dan memastikan bahwa semua data dlaam format yang benar (int, float, text atau lainnya).

Di Pandas, kita gunakan:

.dtype() untuk cek tipe data

.astype() untuk mengubah tipe data

```
#Lihat tipe datasetiap kolom
df.dtypes
```

```
Class              int64
AGE              float64
SEX                int64
STEROID            int64
ANTIVIRALS         int64
FATIGUE            int64
MALAISE            int64
ANOREXIA           int64
LIVER BIG          int64
LIVER FIRM         int64
SPLEEN ALPABLE     int64
SPIDERS            int64
ASCITES            int64
VARICES            int64
BILIRUBIN        float64
ALK PHOSPHATE    float64
SGOT             float64
ALBUMIN          float64
PROTIME          float64
HISTOLOGY          int64
dtype: object
```

Fitur STEROID, FATIGUE, MALAISE, ANOREXIA, LIVER BIG, LIVER FIRM, SPLEEN ALPABLE, SPIDERS, ASCITES, VARICES berupa angka 1-2, harapan kita dapat di ubah menjadi integer.

sedangkan fitur BILIRUBIN, ALK PHOSPHATE, SGOT, ALBUMIN dan PROTIME berupa numerik, disini akan kita assign sebagai float

```
df[["BILIRUBIN", "ALK PHOSPHATE", "SGOT", "ALBUMIN", "PROTIME"]] = df[["BILIRUBIN", "ALK P
```

```
df[[ 'STEROID', 'FATIGUE', 'MALAISE', 'ANOREXIA', 'LIVER BIG', 'LIVER FIRM', 'SPLEEN ALPAB
```

Mari lihat hasil konversinya :

```
df.dtypes
```

```
Class              int64
AGE              float64
SEX                int64
STEROID            int64
ANTIVIRALS         int64
FATIGUE            int64
MALAISE            int64
ANOREXIA           int64
LIVER BIG          int64
LIVER FIRM         int64
SPLEEN ALPABLE     int64
SPIDERS            int64
ASCITES            int64
VARICES            int64
BILIRUBIN        float64
ALK PHOSPHATE    float64
SGOT             float64
ALBUMIN          float64
PROTIME          float64
HISTOLOGY          int64
dtype: object
```

Wokeey!

akhirnya kita punya data yag sudah bersih tanpa missing value dan dengan tipe yang tepat.

```
df.head()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVE FIR
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	

Data Normalization

Mengapa perlu normalisasi?

Normalisasi merupakan proses mentransformasi nilai dari beberapa variabel ke dalam range yang sama. Normalisasi melakukan scaling pada variabel rata-ratanya 0, variance nya 0, atau menskalakan variabel supaya berada dalam range 0-1.

Contoh

Untuk mendemonstrasikan normalisasi, kita skalakan kolom "BILIRUBIN"

Target: menormalisasikan variabel2 tsb supaya berada pada range 0-1.

Pendekatan: replace nilai awal dengan MIN-MAX, dimana min = 0, max = 1

$$nilai\_baru = \frac{nilai\_lama - nilai\_\min}{nilai\_maks - nilai\_\min}(maks\_baru - \min\_baru) + \min\_baru$$

```
#min-max untuk bilirubin
df['BILIRUBIN'] = (df['BILIRUBIN']-df['BILIRUBIN'].min())/(df['BILIRUBIN'].max()-df['BILIF
```

Kemudian misal teknik min-max untuk AGE, ALK PHOSPATE, SGOT, ALBUMIN, dan PROTIME

```
#min-max untuk AGE
df['AGE'] = (df['AGE']-df['AGE'].min())/(df['AGE'].max()-df['AGE'].min())*(1-0)+0
```

```
#min-max untuk ALK PHOSPATE
df['ALK PHOSPHATE'] = (df['ALK PHOSPHATE']-df['ALK PHOSPHATE'].min())/(df['ALK PHOSPHATE']
```

```
#min-max untuk SGOT
df['SGOT'] = (df['SGOT']-df['SGOT'].min())/(df['SGOT'].max()-df['SGOT'].min())*(1-0)+0
```

```
#min-max untuk ALBUMIN
df['ALBUMIN'] = (df['ALBUMIN']-df['ALBUMIN'].min())/(df['ALBUMIN'].max()-df['ALBUMIN'].mir
```

```
#min-max untuk PROTIME
df['PROTIME'] = (df['PROTIME']-df['PROTIME'].min())/(df['PROTIME'].max()-df['PROTIME'].mir
```

Setelah selesai, kita bisa melihat kembali deskripsi datanya

```
df.describe()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAI
count	155.000000	155.000000	155.000000	155.000000	155.000000	155.000000	155.000(
mean	1.793548	0.481690	1.103226	1.509677	1.845161	1.348387	1.6064
std	0.406070	0.176984	0.305240	0.501527	0.362923	0.478004	0.490
min	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000(
25%	2.000000	0.352113	1.000000	1.000000	2.000000	1.000000	1.000(
50%	2.000000	0.450704	1.000000	2.000000	2.000000	1.000000	2.000(

```
df.head(10)
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LIVE FIR
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	
5	2	0.380282	1	2	2	2	2	2	2	
6	1	0.619718	1	1	2	1	2	1	2	
7	2	0.225352	1	2	2	2	2	2	2	
8	2	0.450704	1	2	2	1	2	2	2	
9	2	0.323944	1	2	2	2	2	2	2	

Deduplikasi Data

Duplikasi data merupakan data dengan kondisi pada row-row tertentu memiliki kesamaan data di seluruh kolomnya. Tentunya ada data yang duplikat dalam dataset yang dimiliki. Kondisi duplikasi harus diatasi dengan jalan mengeliminir baris yang mengalami duplikasi, sehingga proses ini dikenal dengan deduplikasi. mari kita lihat ukuran dataframe kita:

```
df.shape
```

```
(155, 20)
```

Cek apakah ada duplikasi data :

```
df.duplicated(subset=None)
```

```
df.duplicated(subset=None)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
150    False
151    False
152    False
153    False
154    False
Length: 155, dtype: bool
```

untuk menghapus row/baris yang duplikat:

```
df.drop_duplicates()
```

	Class	AGE	SEX	STEROID	ANTIVIRALS	FATIGUE	MALAISE	ANOREXIA	LIVER BIG	LI F
0	2	0.323944	2	1	2	2	2	2	1	
1	2	0.605634	1	1	2	1	2	2	1	
2	2	1.000000	1	2	2	1	2	2	2	
3	2	0.338028	1	2	1	2	2	2	2	
4	2	0.380282	1	2	2	2	2	2	2	
...	...	...	...	...	...	...	...	...	...	
150	1	0.549296	1	2	2	1	1	1	2	
151	2	0.521127	1	2	2	1	2	2	2	
152	2	0.760563	1	1	2	1	1	2	1	
153	2	0.647887	2	1	2	1	2	2	2	
154	1	0.507042	1	2	2	1	2	2	2	

155 rows × 20 columns

dapat dilihat bahwa jumah baris dan kolom masih sama, berarti tidak ada yang baris yang duplicate dalam dataframe kita.

Jika semua data telah selesai di praproses, maka kita dapat menyimpannya kembali ke CSV

```
df.to_csv('clean_df.csv')
```

**TUGAS Pertemuan 4 : **

Selesaikan praproses data ini pada data hepatitis. lalu simpan data hasil praproses dengan nama hepatitis_praproses.csv

```
df.to_csv('hepatitis_praproses.csv')
```

✓ 0s completed at 4:20 PM