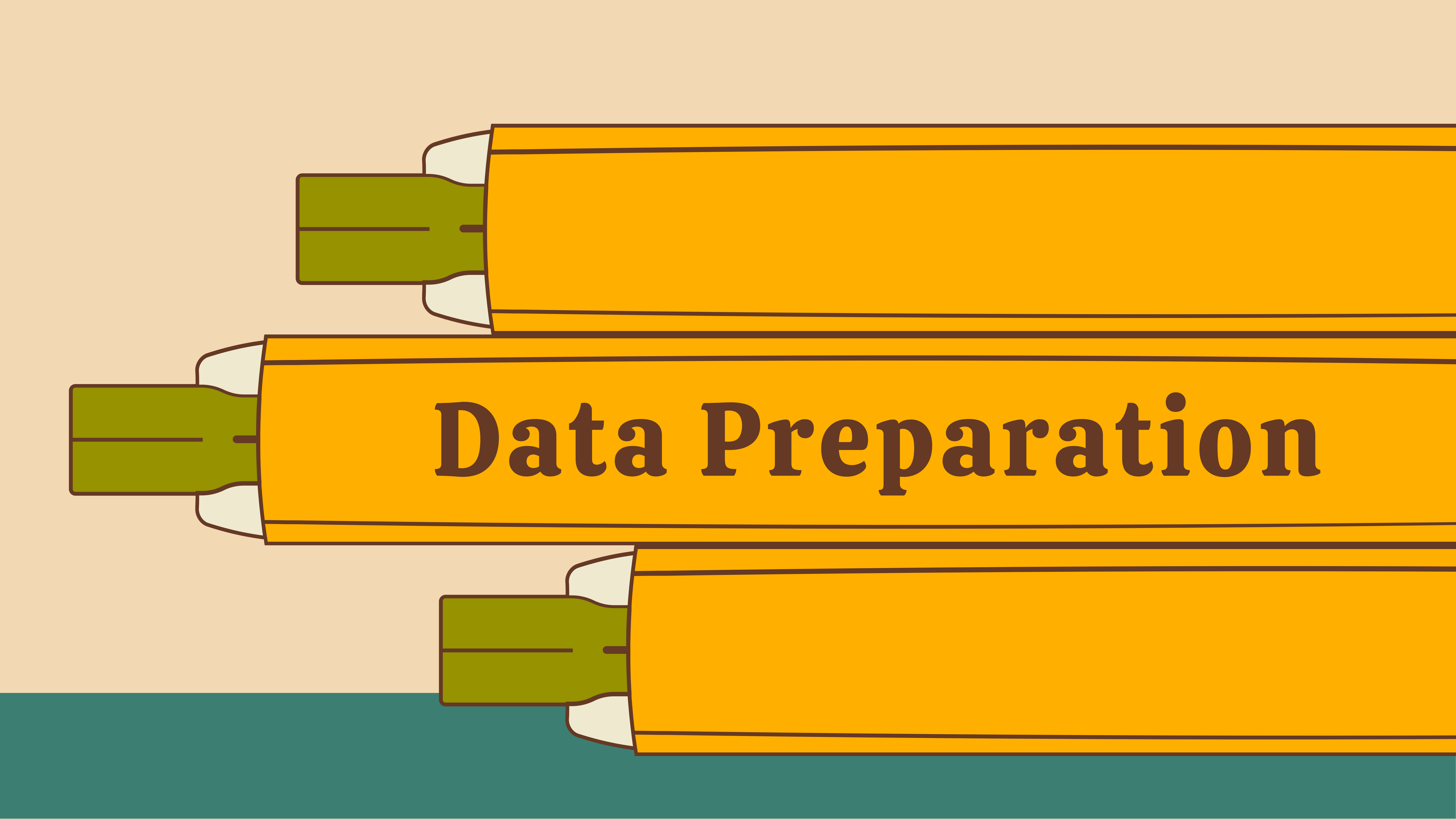
Mendeteksi stroke menggunakan decision tree

Muhammad Ikram Fauzan

H071201039


Data Preparation

Library

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score,precision_score,classification_report
from sklearn.metrics import recall_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
```

Membaca dataset dan menampilkan 5 baris data

```
data=pd.read_csv('healthcare-dataset-stroke-data.csv')
data.head()
```

✓ 0.9s

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Drop kolom id

```
#Drop kolom Id karna gak guna
data = data.drop(columns=['id'])
```

✓ 0.3s

Exploratory Data

```python
data['gender'].value_counts()
```

✓ 0.4s

```
Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```

```python
data[data['gender'] == 'Other']
```

✓ 0.6s

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
3116	56156	Other	26.0	0	0	No	Private	Rural	143.33	22.4	formerly smoked	0

```python
data.drop([3116], inplace=True)
data['gender'].value_counts()
```

✓ 0.5s

```
Female    2994
Male      2115
Name: gender, dtype: int64
```

info() : Nomor index beserta tipe datanya.

```
data.info()
✓ 0.6s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5109 entries, 0 to 5108
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5109 non-null   object
 1   age                5109 non-null   float64
 2   hypertension       5109 non-null   int64
 3   heart_disease      5109 non-null   int64
 4   ever_married       5109 non-null   object
 5   work_type          5109 non-null   object
 6   Residence_type     5109 non-null   object
 7   avg_glucose_level  5109 non-null   float64
 8   bmi                4908 non-null   float64
 9   smoking_status     5109 non-null   object
 10  stroke             5109 non-null   int64
dtypes: float64(3), int64(3), object(5)
memory usage: 439.2+ KB
```

Menampilkan informasi detail tentang dataframe, seperti jumlah baris data, nama-nama kolom berserta jumlah data dan tipe datanya, dan sebagainya.

describe()

```
data.describe()
```

✓ 0.7s

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5109.000000	5109.000000	5109.000000	5109.000000	4908.00000	5109.000000
mean	43.229986	0.097475	0.054022	106.140399	28.89456	0.048738
std	22.613575	0.296633	0.226084	45.285004	7.85432	0.215340
min	0.080000	0.000000	0.000000	55.120000	10.30000	0.000000
25%	25.000000	0.000000	0.000000	77.240000	23.50000	0.000000
50%	45.000000	0.000000	0.000000	91.880000	28.10000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	33.10000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.60000	1.000000

Mengecek korelasi

```
data.corr()
```

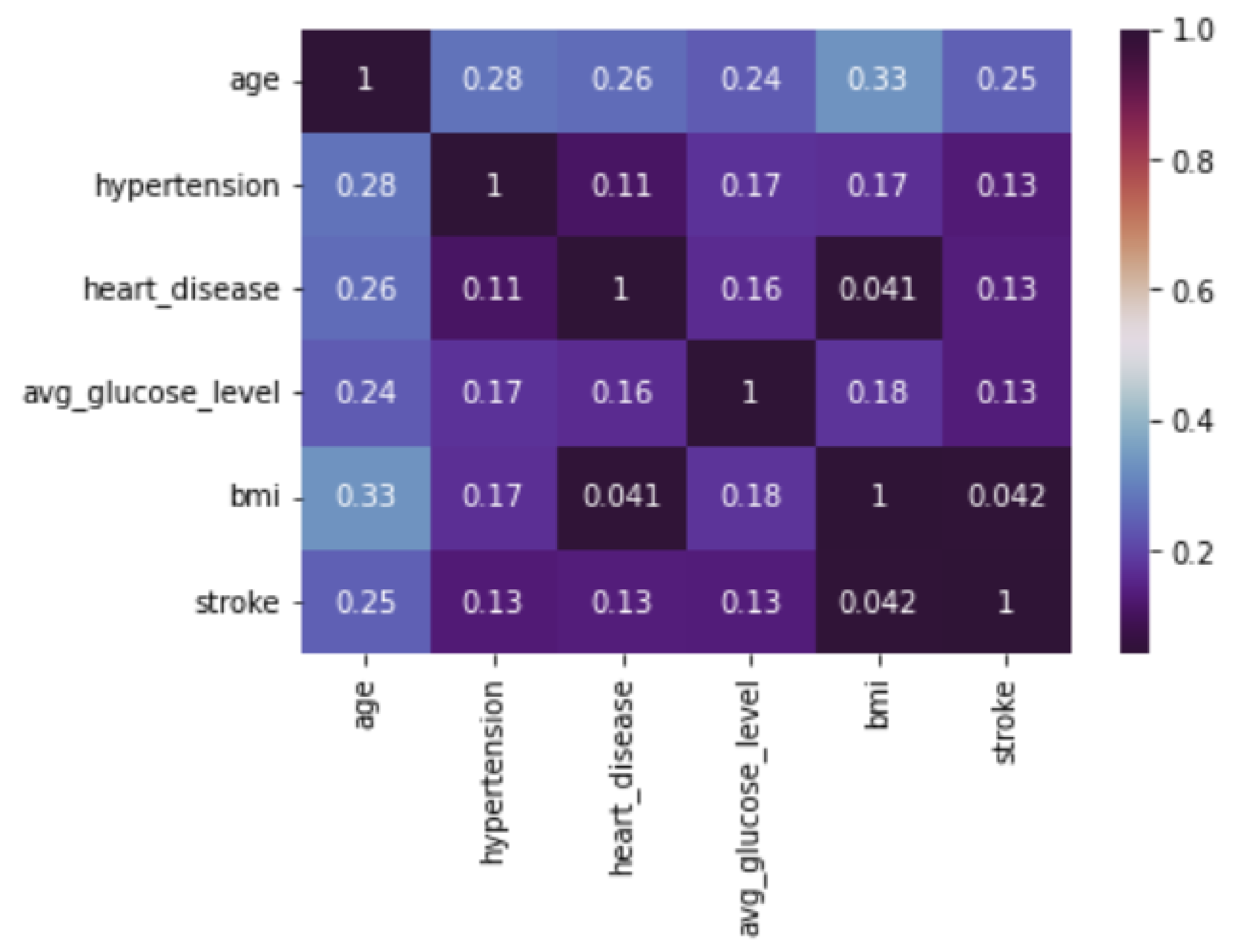
✓ 0.4s

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
age	1.000000	0.276367	0.263777	0.238323	0.333314	0.245239
hypertension	0.276367	1.000000	0.108292	0.174540	0.167770	0.127891
heart_disease	0.263777	0.108292	1.000000	0.161907	0.041322	0.134905
avg_glucose_level	0.238323	0.174540	0.161907	1.000000	0.175672	0.131991
bmi	0.333314	0.167770	0.041322	0.175672	1.000000	0.042341
stroke	0.245239	0.127891	0.134905	0.131991	0.042341	1.000000

```
sns.heatmap(data.corr(),annot=True,cmap='twilight_shifted')
```

✓ 0.4s

```
<AxesSubplot: >
```



Mencari nilai null dan menghitung percentage nya

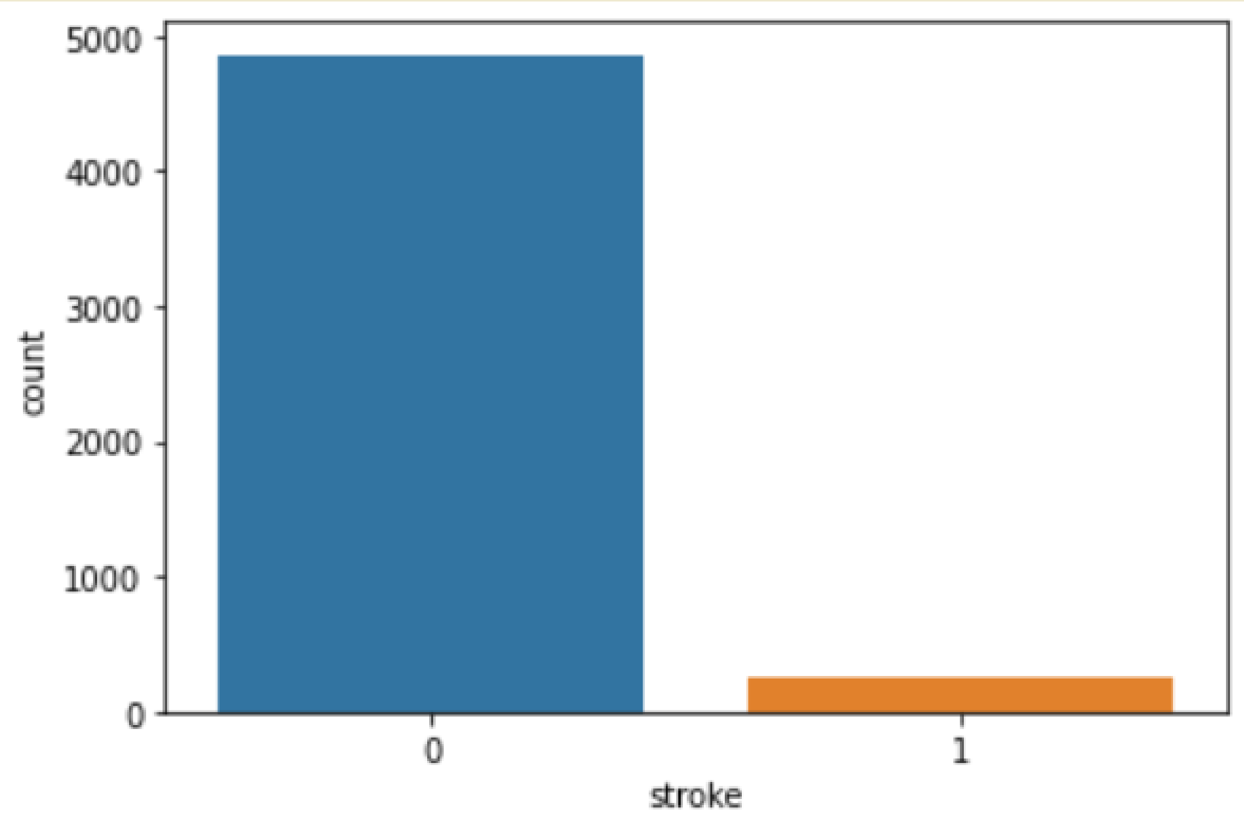
```
data_null=pd.DataFrame(data.isnull().sum(),columns=['Number of null'])
data_null['percentage']=(data.isnull().sum())/len(data)*100
data_null
```

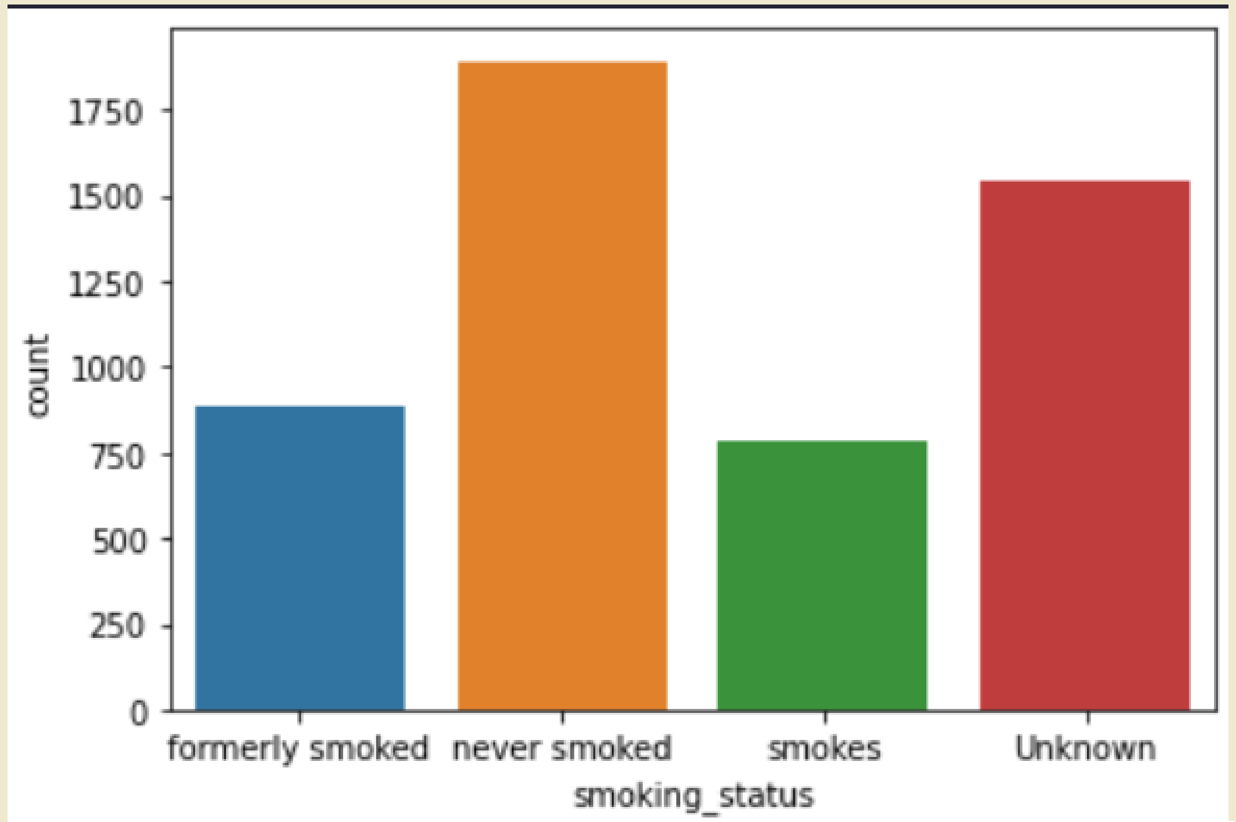
✓ 0.4s

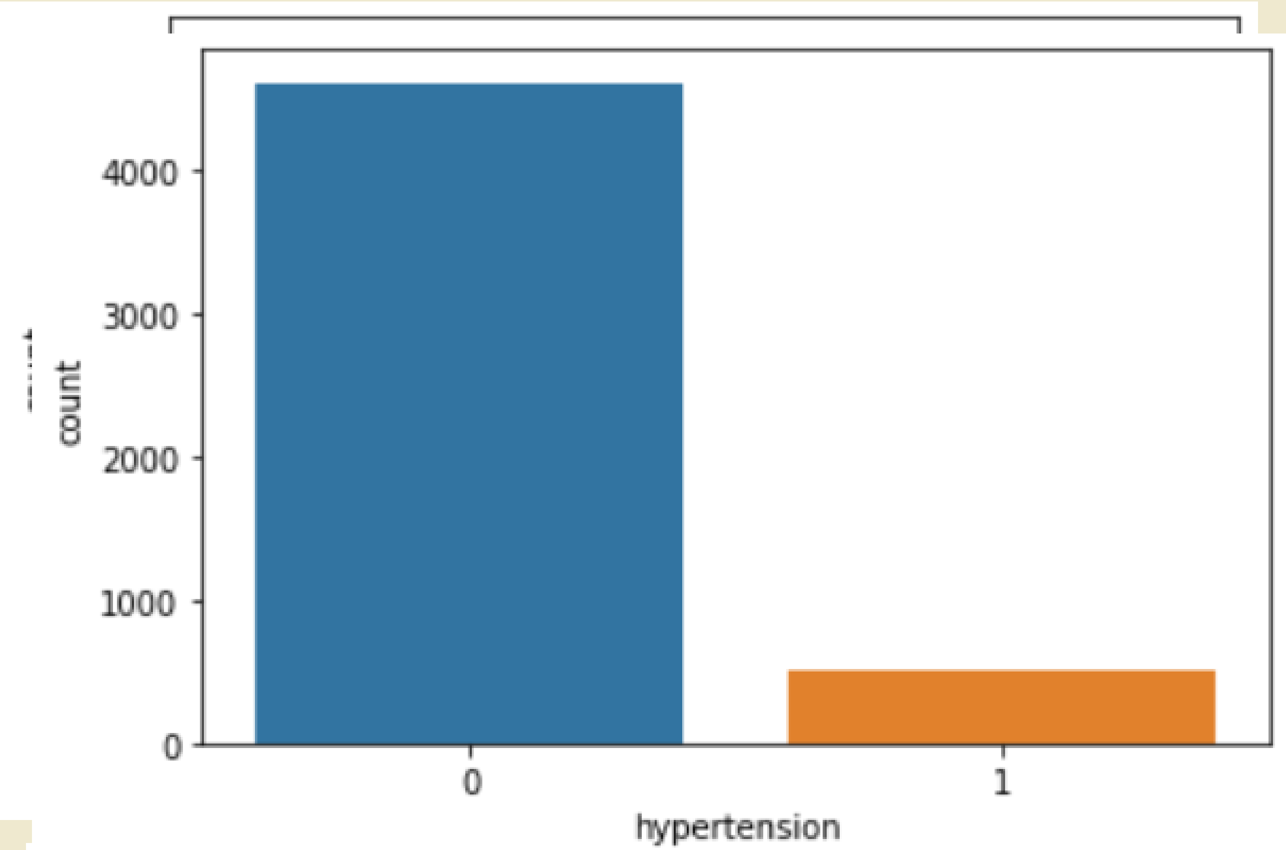
	Number of null	percentage
gender	0	0.000000
age	0	0.000000
hypertension	0	0.000000
heart_disease	0	0.000000
ever_married	0	0.000000
work_type	0	0.000000
Residence_type	0	0.000000
avg_glucose_level	0	0.000000
bmi	201	3.934234
smoking_status	0	0.000000
stroke	0	0.000000

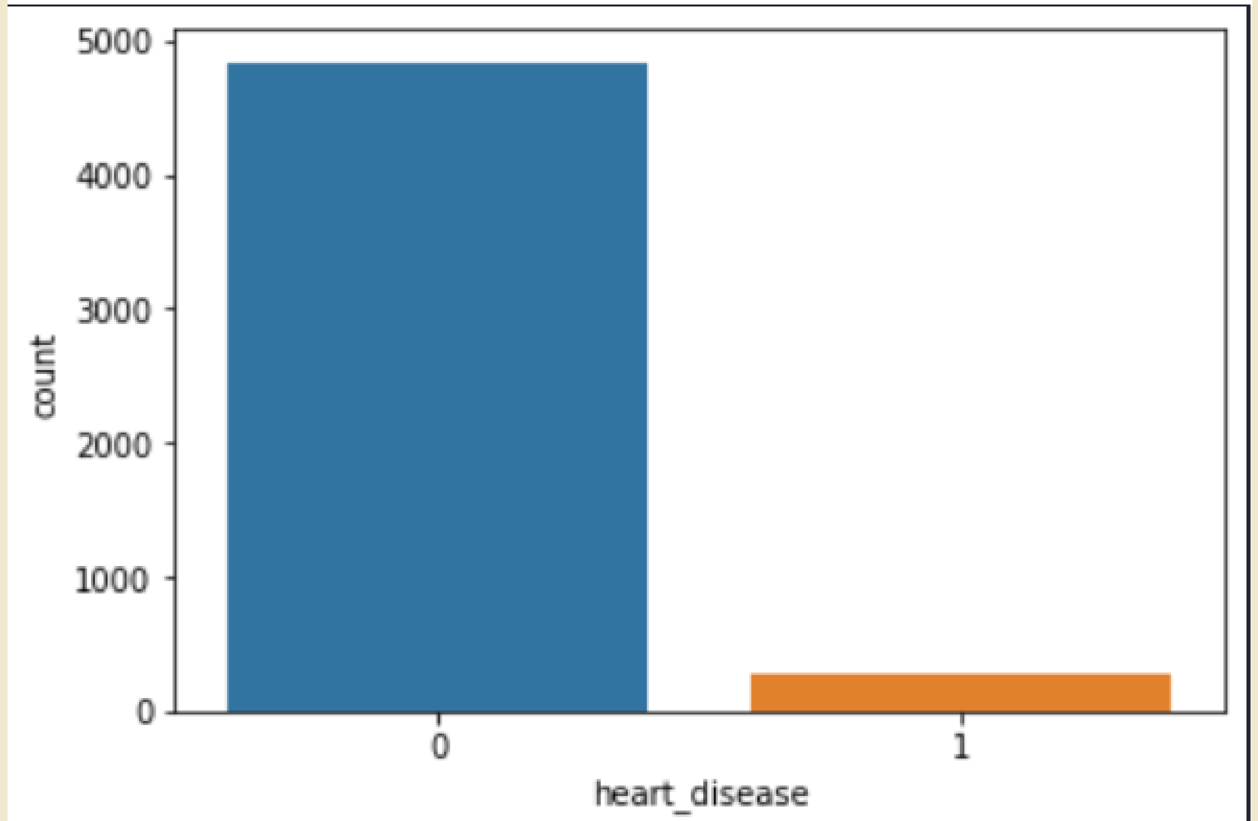
Tampilkan distribusi beberapa kolom

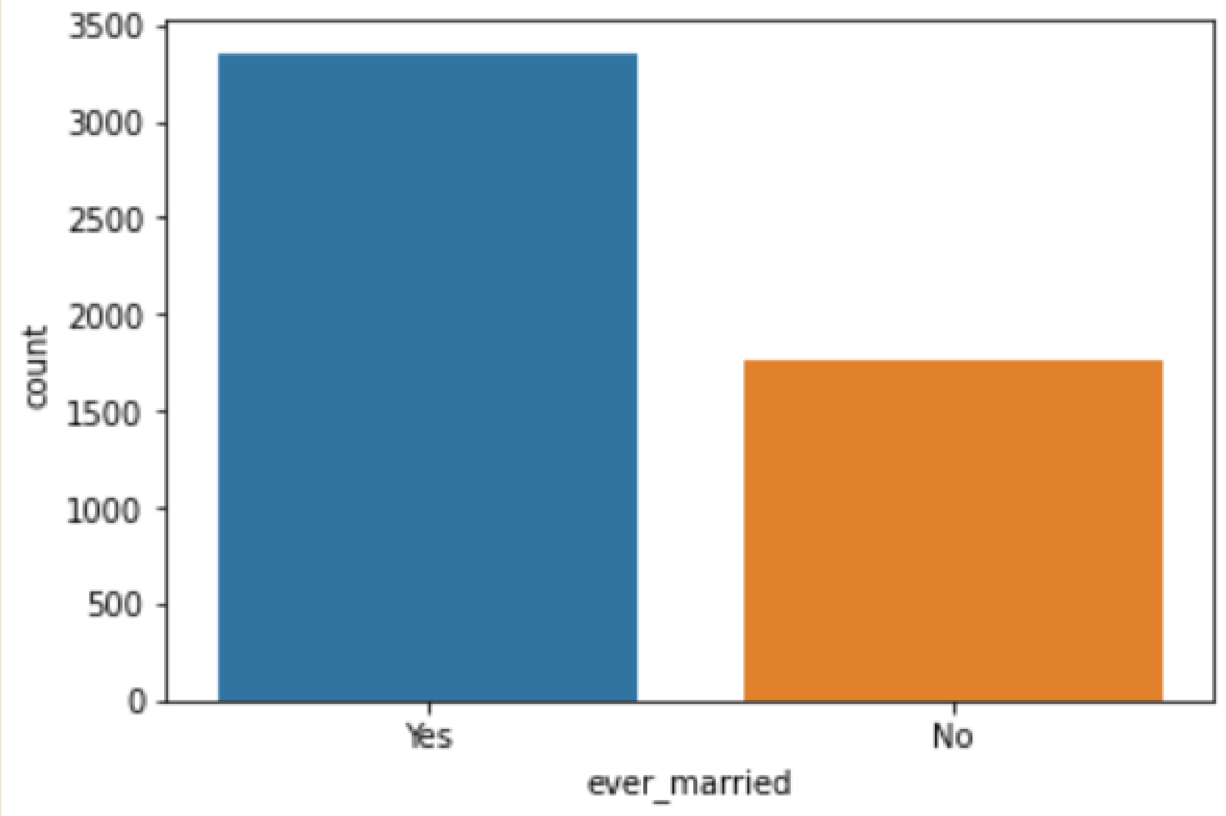
```python
list1=['stroke','smoking_status','hypertension','heart_disease','ever_married','work_type','Residence_type','gender','age']
for col in list1:
    plt.figure()
    if col=='age':
        plt.figure(figsize=(35,10))
    sns.countplot(x=col,data=data)
```

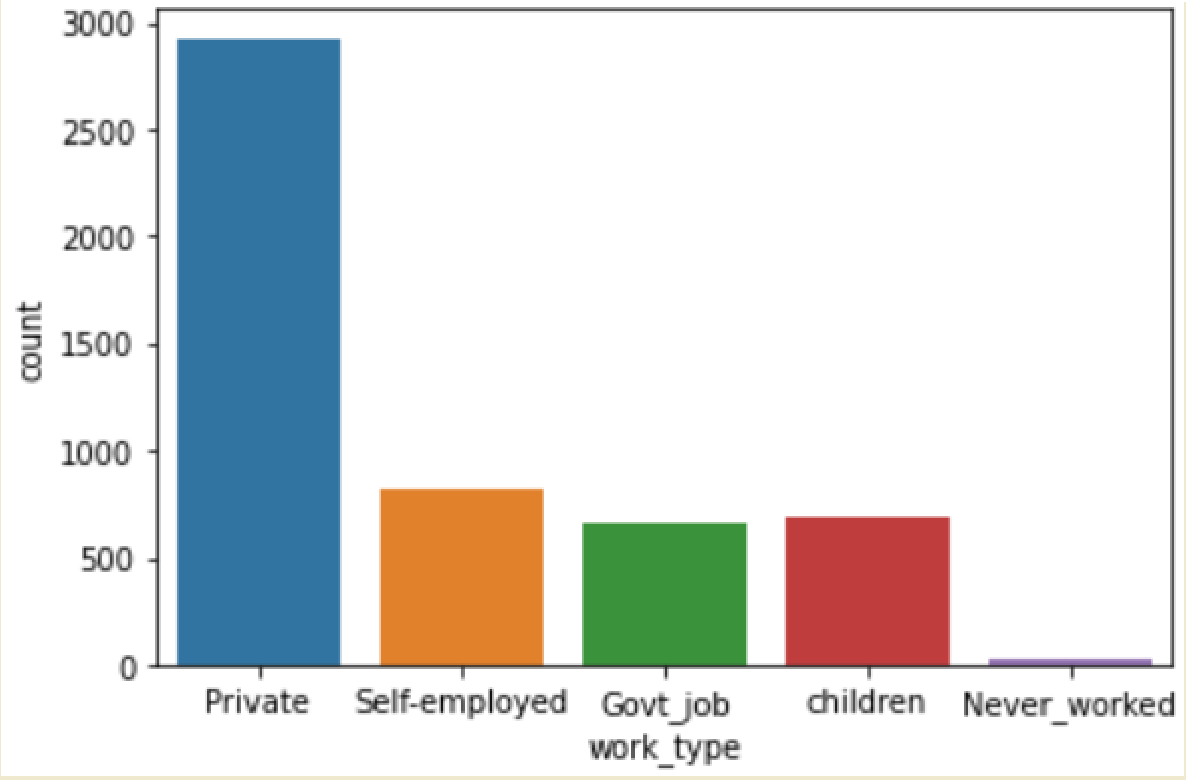


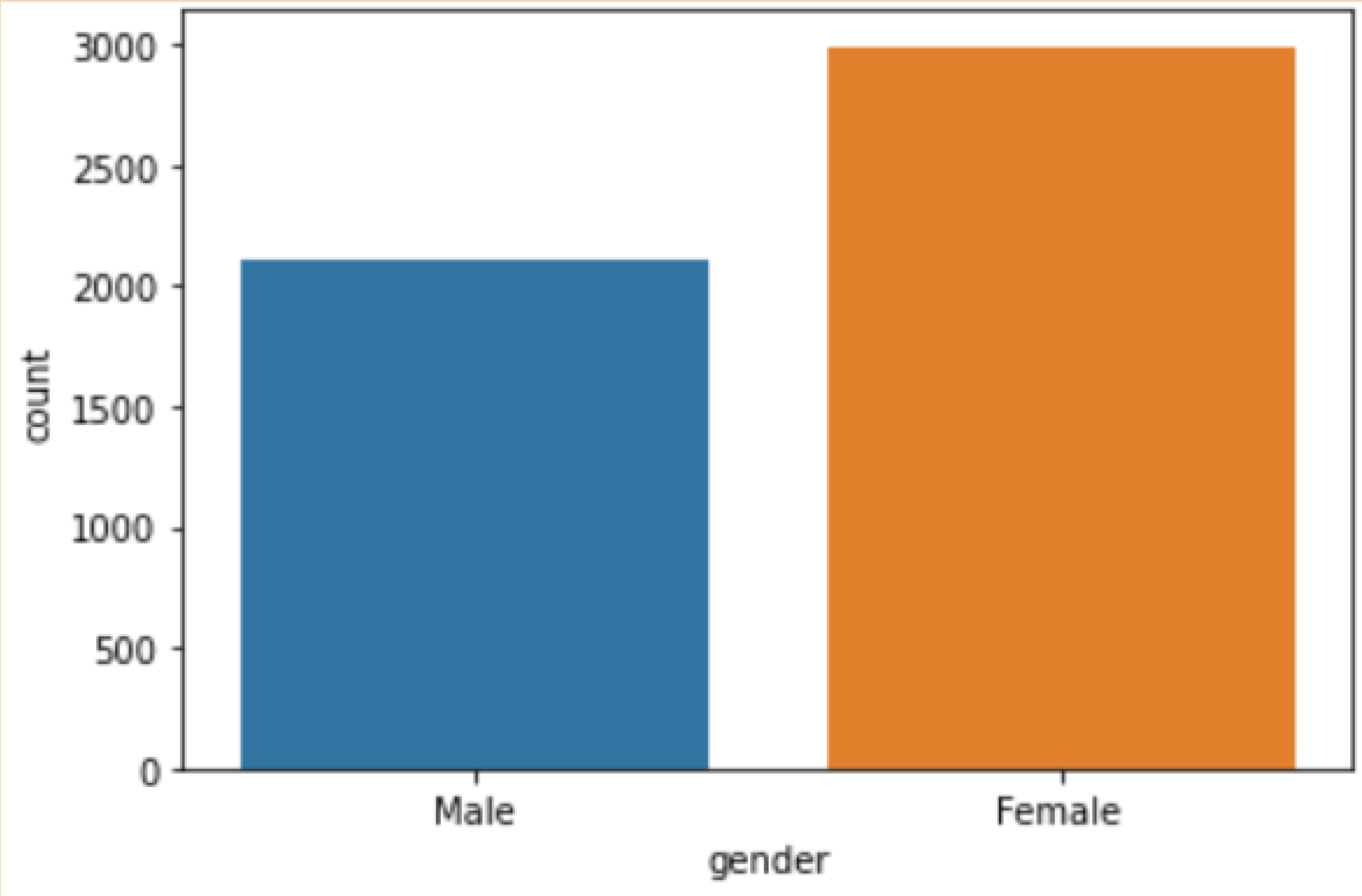


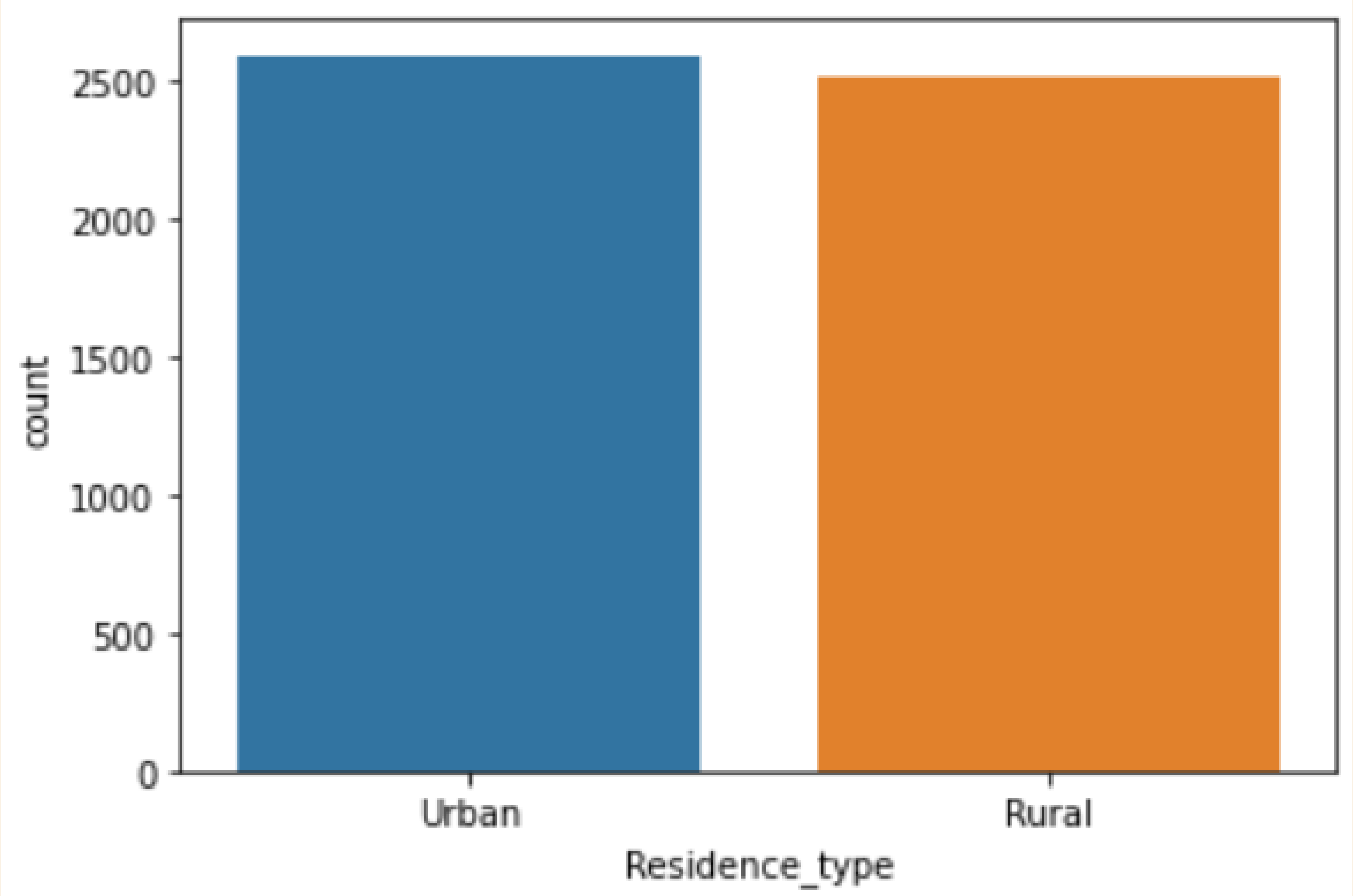


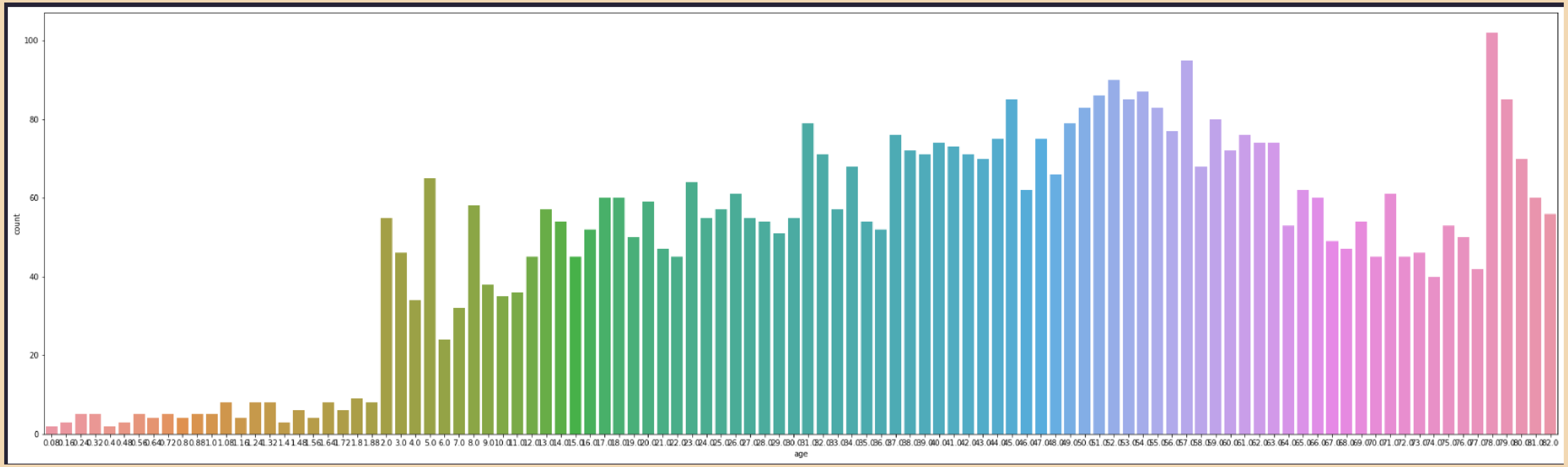







3000
2500
2000
1500
1000
500
0
count
Male
Female
gender


2500
2000
1500
1000
500
0
count
Urban
Rural
Residence_type


100
80
60
40
20
0
count
age

```python
#Membagi data label dan feature

X=data.iloc[:,0:-1]
y=data.iloc[:,-1]
key=X.keys()
```
✓ 0.3s

```python
X
```
✓ 0.4s

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked
...	...	...	...	...	...	...	...	...	...	...
5104	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked
5105	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked
5106	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked
5107	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked
5108	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown

5109 rows × 10 columns

```python
y
```
✓ 0.9s

```
0       1
1       1
2       1
3       1
4       1
       ..
5104    0
5105    0
5106    0
5107    0
5108    0
Name: stroke, Length: 5109, dtype: int64
```

Mengubah data yang bertype objek menjadi numerik

```
list1=['gender','ever_married','work_type','Residence_type','smoking_status']
label=LabelEncoder()
for col in list1:
    X[col]=label.fit_transform(X[col])
X
```

✓ 0.6s

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1	67.0	0	1	1	2	1	228.69	36.6	1
1	0	61.0	0	0	1	3	0	202.21	NaN	2
2	1	80.0	0	1	1	2	0	105.92	32.5	2
3	0	49.0	0	0	1	2	1	171.23	34.4	3
4	0	79.0	1	0	1	3	0	174.12	24.0	2
...	...	...	...	...	...	...	...	...	...	...
5104	0	80.0	1	0	1	2	1	83.75	NaN	2
5105	0	81.0	0	0	1	3	1	125.20	40.0	2
5106	0	35.0	0	0	1	3	0	82.99	30.6	2
5107	1	51.0	0	0	1	2	0	166.29	25.6	1
5108	0	44.0	0	0	1	0	1	85.28	26.2	0

5109 rows × 10 columns

Mengganti nilai NaN

```
impute = SimpleImputer(missing_values=np.nan, strategy='mean')
X = impute.fit_transform(X)
pd.DataFrame(X,columns=key)
```

✓ 0.6s

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1.0	67.0	0.0	1.0	1.0	2.0	1.0	228.69	36.60000	1.0
1	0.0	61.0	0.0	0.0	1.0	3.0	0.0	202.21	28.89456	2.0
2	1.0	80.0	0.0	1.0	1.0	2.0	0.0	105.92	32.50000	2.0
3	0.0	49.0	0.0	0.0	1.0	2.0	1.0	171.23	34.40000	3.0
4	0.0	79.0	1.0	0.0	1.0	3.0	0.0	174.12	24.00000	2.0
...	...	...	...	...	...	...	...	...	...	...
5104	0.0	80.0	1.0	0.0	1.0	2.0	1.0	83.75	28.89456	2.0
5105	0.0	81.0	0.0	0.0	1.0	3.0	1.0	125.20	40.00000	2.0
5106	0.0	35.0	0.0	0.0	1.0	3.0	0.0	82.99	30.60000	2.0
5107	1.0	51.0	0.0	0.0	1.0	2.0	0.0	166.29	25.60000	1.0
5108	0.0	44.0	0.0	0.0	1.0	0.0	1.0	85.28	26.20000	0.0

5109 rows × 10 columns

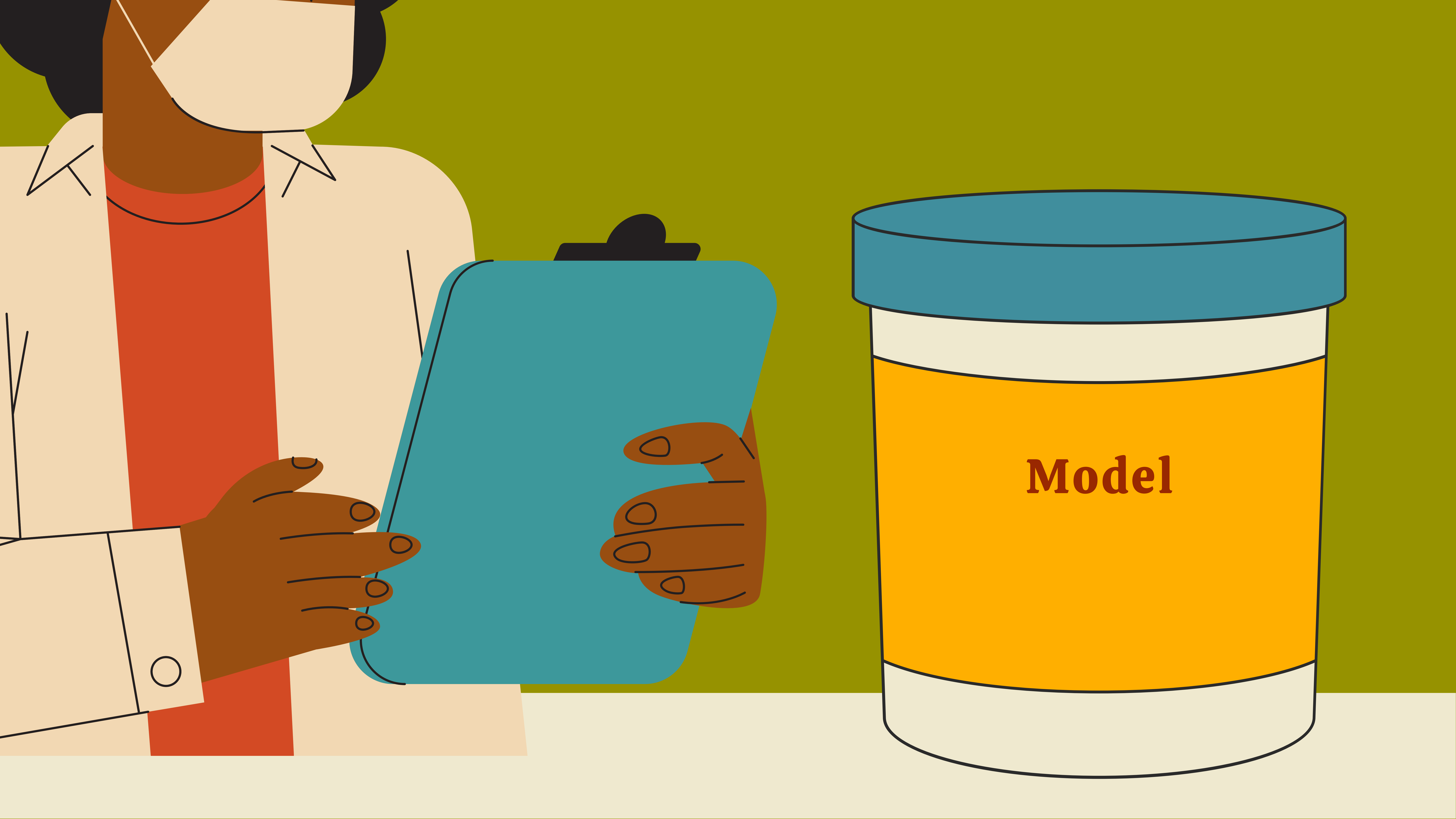
MinMaxScaler

```python
scaler = MinMaxScaler(copy=True, feature_range=(0, 1))
X = scaler.fit_transform(X)
pd.DataFrame(X,columns=key)
```

✓ 0.5s

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1.0	0.816895	0.0	1.0	1.0	0.50	1.0	0.801265	0.301260	0.333333
1	0.0	0.743652	0.0	0.0	1.0	0.75	0.0	0.679023	0.212996	0.666667
2	1.0	0.975586	0.0	1.0	1.0	0.50	0.0	0.234512	0.254296	0.666667
3	0.0	0.597168	0.0	0.0	1.0	0.50	1.0	0.536008	0.276060	1.000000
4	0.0	0.963379	1.0	0.0	1.0	0.75	0.0	0.549349	0.156930	0.666667
...	...	...	...	...	...	...	...	...	...	...
5104	0.0	0.975586	1.0	0.0	1.0	0.50	1.0	0.132167	0.212996	0.666667
5105	0.0	0.987793	0.0	0.0	1.0	0.75	1.0	0.323516	0.340206	0.666667
5106	0.0	0.426270	0.0	0.0	1.0	0.75	0.0	0.128658	0.232532	0.666667
5107	1.0	0.621582	0.0	0.0	1.0	0.50	0.0	0.513203	0.175258	0.333333
5108	0.0	0.536133	0.0	0.0	1.0	0.00	1.0	0.139230	0.182131	0.000000

5109 rows × 10 columns

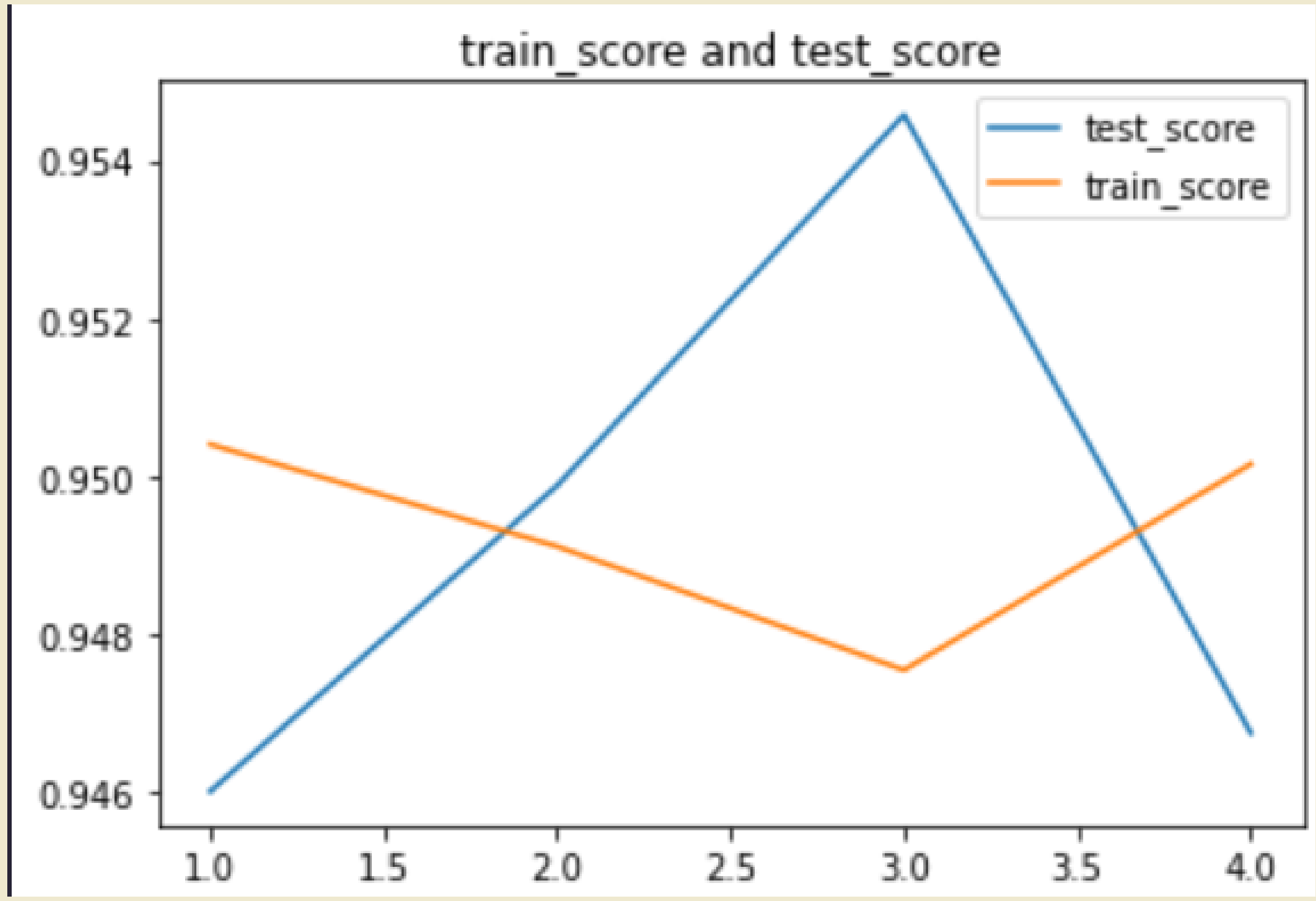

Model

Menggunakan Model RandomForeestClassifier

```
kfold = KFold(n_splits=4, random_state=44, shuffle =True)
test_score=[]
train_score=[]
RandomForestClassifierModel = RandomForestClassifier(criterion = 'gini',n_estimators=100,max_depth=25,random_state=33)
for train_index, test_index in kfold.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    RandomForestClassifierModel.fit(X[:1350], y[:1350])
    test_score.append(RandomForestClassifierModel.score(X_test, y_test))
    train_score.append(RandomForestClassifierModel.score(X_train, y_train))
plt.plot(range(1,5),test_score,label="test_score")
plt.plot(range(1,5),train_score,label="train_score")
plt.title("train_score and test_score")
plt.legend()
print("test score =",np.mean(test_score))
print("train score",np.mean(train_score))
```

✓ 1.6s

```
test score = 0.9493057929934081
train score 0.9493052195064
```

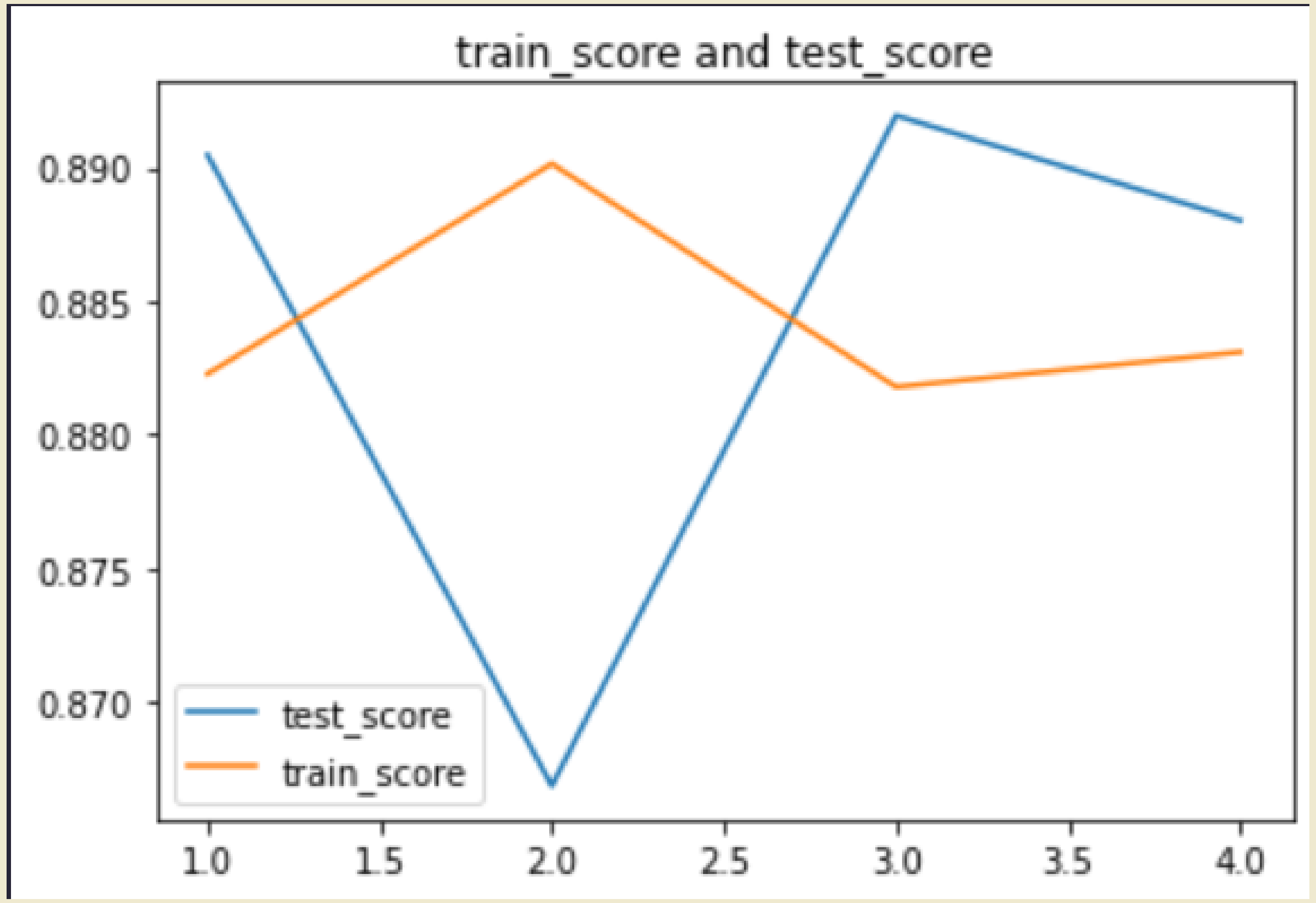


Menggunakan Model DecisionTreeClassifier

```
DecisionTreeClassifierModel = DecisionTreeClassifier(criterion='gini',max_depth=10,random_state=33)
test_score=[]
train_score=[]
for train_index, test_index in kfold.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    DecisionTreeClassifierModel.fit(X[:1350], y[:1350])
    test_score.append(DecisionTreeClassifierModel.score(X_test, y_test))
    train_score.append(DecisionTreeClassifierModel.score(X_train, y_train))
plt.plot(range(1,5),test_score,label="test_score")
plt.plot(range(1,5),train_score,label="train_score")
plt.title("train_score and test_score")
plt.legend()
print("test score =",np.mean(test_score))
print("train score",np.mean(train_score))
```

✓ 0.2s

```
test score = 0.8843205846056938
train score 0.8843216516289211
```



```python
y_pred = RandomForestClassifierModel.predict(X)
print('Predicted Value for RandomForestClassifierModel is : ' , y_pred[:10])
```

✓ 0.9s

```
Predicted Value for RandomForestClassifierModel is :  [1 1 1 1 1 1 1 1 1 1]
```

Confusion Matrix

```
CM = confusion_matrix(y, y_pred)
CM
```

✓ 0.3s

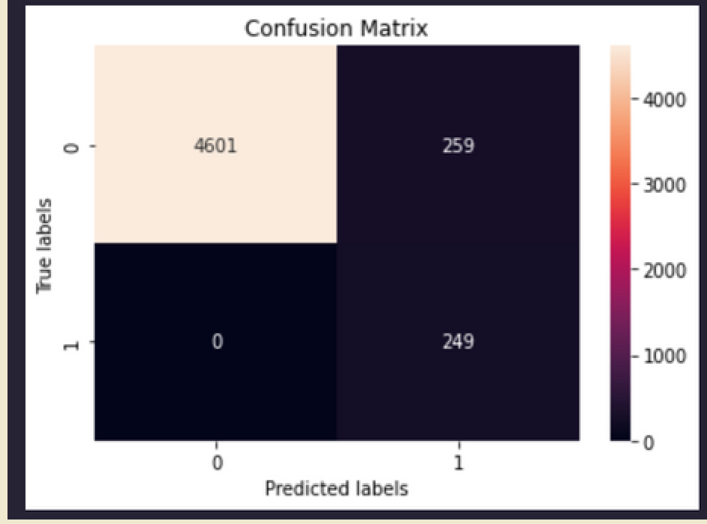
```
array([[4601,  259],
       [   0,  249]], dtype=int64)
```

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(CM, annot=True, fmt='g', ax=ax);

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```

✓ 0.1s



Menghitun Accuracy, Precision, Recall, dan F1-Score

```python
print('Accuracy: {}'.format(accuracy_score(y, y_pred)))
```
✓ 0.4s

```
Accuracy: 0.9493051477784302
```

```python
print('Precision: {}'.format(precision_score(y, y_pred)))
```
✓ 0.6s

```
Precision: 0.49015748031496065
```

```python
print('Recall: {}'.format(recall_score(y, y_pred)))
```
✓ 0.4s

```
Recall: 1.0
```

```python
print('F1-Score: {}'.format(f1_score(y, y_pred)))
```
✓ 0.3s

```
F1-Score: 0.6578599735799208
```

Classification Report

```
ClassificationReport = classification_report(y,y_pred)
print(ClassificationReport )
```

✓ 0.5s

	precision	recall	f1-score	support
0	1.00	0.95	0.97	4860
1	0.49	1.00	0.66	249
accuracy			0.95	5109
macro avg	0.75	0.97	0.82	5109
weighted avg	0.98	0.95	0.96	5109

Thank you for listening!

Don't hesitate to ask any questions!