NAMA : Azzuhri Wisnu Akhsan

NIM : A11.2020.13187

Kelas : BKDS02

Projek Bimbingan Karir Data Science

1) Pengumpulan Data

Data yang digunakan merupakan dataset penyakit jantung yang diambil melalui link : https://archive.ics.uci.edu/dataset/45/heart+disease Dataset yang dipakai adalah dataset dengan nam file "Hungarian.data", diharapkan untuk membaca dokumentasi pada "heart-disease.name"

2) Menelaah data

Masukan library yang diperlukan

```
import pandas as pd
import re
import numpy as np
import itertools
```

Load Dataset

```
dir = 'hungarian.data'
```

```
with open (dir, encoding='Latin1') as file :
  lines = [line.strip() for line in file]

lines[0:10]
```

```
['1254 0 40 1 1 0 0',
 '-9 2 140 0 289 -9 -9 -9',
 '0 -9 -9 0 12 16 84 0',
 '0 0 0 0 150 18 -9 7',
 '172 86 200 110 140 86 0 0',
 '0 -9 26 20 -9 -9 -9 -9',
 '-9 -9 -9 -9 -9 -9 -9 12',
 '20 84 0 -9 -9 -9 -9 -9',
```

```
 '-9 -9 -9 -9 -9 1 1 1',
 '1 1 -9. -9. name']
```

Rubah bentuk data menjadi dataframe agar lebih mudah dipahami

```
data = itertools.takewhile(
    lambda x: len(x) == 76,
    (' '.join(lines[i:(i + 10)]).split() for i in range(0, len(lines), 10))
)

df = pd.DataFrame.from_records(data)

df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	66	67	68	69	70	71	72	73	74	75
0	1254	0	40	1	1	0	0	-9	2	140	...	-9	-9	1	1	1	1	1	-9.	-9.	name
1	1255	0	49	0	1	0	0	-9	3	160	...	-9	-9	1	1	1	1	1	-9.	-9.	name
2	1256	0	37	1	1	0	0	-9	2	130	...	-9	-9	1	1	1	1	1	-9.	-9.	name
3	1257	0	48	0	1	1	1	-9	4	138	...	2	-9	1	1	1	1	1	-9.	-9.	name
4	1258	0	54	1	1	0	1	-9	3	150	...	1	-9	1	1	1	1	1	-9.	-9.	name

5 rows × 76 columns

menampilkan informasi dari file dataset yang sudah dimasukkan dalam dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 76 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       294 non-null    object
 1   1       294 non-null    object
 2   2       294 non-null    object
 3   3       294 non-null    object
 4   4       294 non-null    object
 5   5       294 non-null    object
 6   6       294 non-null    object
 7   7       294 non-null    object
 8   8       294 non-null    object
 9   9       294 non-null    object
 10  10      294 non-null    object
 11  11      294 non-null    object
 12  12      294 non-null    object
 13  13      294 non-null    object
 14  14      294 non-null    object
 15  15      294 non-null    object
 16  16      294 non-null    object
 17  17      294 non-null    object
```

```
 18  18      294 non-null    object
 19  19      294 non-null    object
 20  20      294 non-null    object
 21  21      294 non-null    object
 22  22      294 non-null    object
 23  23      294 non-null    object
 24  24      294 non-null    object
 25  25      294 non-null    object
 26  26      294 non-null    object
 27  27      294 non-null    object
 28  28      294 non-null    object
 29  29      294 non-null    object
 30  30      294 non-null    object
 31  31      294 non-null    object
 32  32      294 non-null    object
 33  33      294 non-null    object
 34  34      294 non-null    object
 35  35      294 non-null    object
 36  36      294 non-null    object
 37  37      294 non-null    object
 38  38      294 non-null    object
 39  39      294 non-null    object
 40  40      294 non-null    object
 41  41      294 non-null    object
 42  42      294 non-null    object
 43  43      294 non-null    object
 44  44      294 non-null    object
 45  45      294 non-null    object
 46  46      294 non-null    object
 47  47      294 non-null    object
 48  48      294 non-null    object
 49  49      294 non-null    object
 50  50      294 non-null    object
 51  51      294 non-null    object
```

```
df = df.iloc[:,:-1]
df = df.drop(df.columns[0], axis=1)
```

mengubah tipe file dataset menjadi tipe data float sesuai dengan nilai null yaitu -0.9

```
df = df.astype(float)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 74 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   1       294 non-null    float64
 1   2       294 non-null    float64
 2   3       294 non-null    float64
 3   4       294 non-null    float64
 4   5       294 non-null    float64
 5   6       294 non-null    float64
 6   7       294 non-null    float64
```

```
 7   8        294 non-null    float64
 8   9        294 non-null    float64
 9   10       294 non-null    float64
 10  11       294 non-null    float64
 11  12       294 non-null    float64
 12  13       294 non-null    float64
 13  14       294 non-null    float64
 14  15       294 non-null    float64
 15  16       294 non-null    float64
 16  17       294 non-null    float64
 17  18       294 non-null    float64
 18  19       294 non-null    float64
 19  20       294 non-null    float64
 20  21       294 non-null    float64
 21  22       294 non-null    float64
 22  23       294 non-null    float64
 23  24       294 non-null    float64
 24  25       294 non-null    float64
 25  26       294 non-null    float64
 26  27       294 non-null    float64
 27  28       294 non-null    float64
 28  29       294 non-null    float64
 29  30       294 non-null    float64
 30  31       294 non-null    float64
 31  32       294 non-null    float64
 32  33       294 non-null    float64
 33  34       294 non-null    float64
 34  35       294 non-null    float64
 35  36       294 non-null    float64
 36  37       294 non-null    float64
 37  38       294 non-null    float64
 38  39       294 non-null    float64
 39  40       294 non-null    float64
 40  41       294 non-null    float64
 41  42       294 non-null    float64
 42  43       294 non-null    float64
 43  44       294 non-null    float64
 44  45       294 non-null    float64
 45  46       294 non-null    float64
 46  47       294 non-null    float64
 47  48       294 non-null    float64
 48  49       294 non-null    float64
 49  50       294 non-null    float64
 50  51       294 non-null    float64
 51  52       294 non-null    float64
 52  53       294 non-null    float64
```

3) Validasi Data

Bertujuan untuk mengetahu kondisi dataset untuk mengetahui langkah apa yang harus dilakukan

Dalam kasus dataset ini mengubah nilai -9.0 menjadi nilai nilai null valuse sesuai dengan deskripsi dataset

```
df.replace(-9.0, np.nan, inplace=True)
```

megnghitung jumlah nilai null value

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

```
1       0
2       0
3       0
4       0
5       0
     ...
70      0
71      0
72      0
73    266
74    294
Length: 74, dtype: int64
```

```
df.head()
```

	1	2	3	4	5	6	7	8	9	10	...	65	66	67	68	69	70
0	0.0	40.0	1.0	1.0	0.0	0.0	NaN	2.0	140.0	0.0	...	NaN	NaN	NaN	1.0	1.0	1.0
1	0.0	49.0	0.0	1.0	0.0	0.0	NaN	3.0	160.0	1.0	...	NaN	NaN	NaN	1.0	1.0	1.0
2	0.0	37.0	1.0	1.0	0.0	0.0	NaN	2.0	130.0	0.0	...	NaN	NaN	NaN	1.0	1.0	1.0
3	0.0	48.0	0.0	1.0	1.0	1.0	NaN	4.0	138.0	0.0	...	NaN	2.0	NaN	1.0	1.0	1.0
4	0.0	54.0	1.0	1.0	0.0	1.0	NaN	3.0	150.0	0.0	...	NaN	1.0	NaN	1.0	1.0	1.0

5 rows × 74 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 74 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   1       294 non-null    float64
 1   2       294 non-null    float64
 2   3       294 non-null    float64
 3   4       294 non-null    float64
 4   5       294 non-null    float64
 5   6       294 non-null    float64
 6   7       0 non-null      float64
 7   8       294 non-null    float64
 8   9       293 non-null    float64
 9   10      293 non-null    float64
 10  11      271 non-null    float64
 11  12      12 non-null     float64
 12  13      1 non-null      float64
 13  14      0 non-null      float64
```

```
 14  15      286 non-null    float64
 15  16      21 non-null     float64
 16  17      1 non-null      float64
 17  18      293 non-null    float64
 18  19      294 non-null    float64
 19  20      294 non-null    float64
 20  21      294 non-null    float64
 21  22      293 non-null    float64
 22  23      292 non-null    float64
 23  24      293 non-null    float64
 24  25      293 non-null    float64
 25  26      293 non-null    float64
 26  27      285 non-null    float64
 27  28      292 non-null    float64
 28  29      104 non-null    float64
 29  30      292 non-null    float64
 30  31      293 non-null    float64
 31  32      293 non-null    float64
 32  33      293 non-null    float64
 33  34      293 non-null    float64
 34  35      293 non-null    float64
 35  36      293 non-null    float64
 36  37      293 non-null    float64
 37  38      292 non-null    float64
 38  39      294 non-null    float64
 39  40      104 non-null    float64
 40  41      293 non-null    float64
 41  42      294 non-null    float64
 42  43      4 non-null      float64
 43  44      0 non-null      float64
 44  45      0 non-null      float64
 45  46      0 non-null      float64
 46  47      3 non-null      float64
 47  48      0 non-null      float64
 48  49      2 non-null      float64
 49  50      28 non-null     float64
 50  51      27 non-null     float64
 51  52      17 non-null     float64
 52  53      0 non-null      float64
```

4) Menentukan Object Data

Memilih 14 fitur yang akan digunakan sesuai dengan deskripsi dataset

```
df_selected = df.iloc[:, [1, 2, 7, 8, 10, 14, 17, 30, 36, 38, 39, 42, 49, 56]]
```

```
df_selected.head()
```

	2	3	8	9	11	15	18	31	37	39	40	43	50	57
0	40.0	1.0	2.0	140.0	289.0	0.0	0.0	172.0	0.0	0.0	NaN	NaN	NaN	0.0
1	49.0	0.0	3.0	160.0	180.0	0.0	0.0	156.0	0.0	1.0	2.0	NaN	NaN	1.0
2	37.0	1.0	2.0	130.0	283.0	0.0	1.0	98.0	0.0	0.0	NaN	NaN	NaN	0.0
3	48.0	0.0	4.0	138.0	214.0	0.0	0.0	108.0	1.0	1.5	2.0	NaN	NaN	3.0
4	54.0	1.0	3.0	150.0	NaN	0.0	0.0	122.0	0.0	0.0	NaN	NaN	NaN	0.0

```
df_selected.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   2       294 non-null    float64
 1   3       294 non-null    float64
 2   8       294 non-null    float64
 3   9       293 non-null    float64
 4   11      271 non-null    float64
 5   15      286 non-null    float64
 6   18      293 non-null    float64
 7   31      293 non-null    float64
 8   37      293 non-null    float64
 9   39      294 non-null    float64
 10  40      104 non-null    float64
 11  43      4 non-null      float64
 12  50      28 non-null     float64
 13  57      294 non-null    float64
dtypes: float64(14)
memory usage: 32.3 KB
```

mengganti nama 14 kolom sesuai dengan deskripsi dataset

```
column_mapping = { 2: 'age',
        3: 'sex',
        8: 'cp',
        9: 'trestbps',
        11: 'chol',
        15: 'fbs',
        18: 'restecg',
        31: 'thalach',
        37: 'exang',
        39: 'oldpeak',
        40: 'slope',
        43: 'ca',
        50: 'thal',
        57: 'target'
    }
```

```
df_selected.rename(columns=column_mapping, inplace=True)
```

```
<ipython-input-17-e9a4003b4301>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  df_selected.rename(columns=column_mapping, inplace=True)
```

```
df_selected.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       294 non-null    float64
 1   sex       294 non-null    float64
 2   cp        294 non-null    float64
 3   trestbps  293 non-null    float64
 4   chol      271 non-null    float64
 5   fbs       286 non-null    float64
 6   restecg   293 non-null    float64
 7   thalach   293 non-null    float64
 8   exang     293 non-null    float64
 9   oldpeak   294 non-null    float64
 10  slope     104 non-null    float64
 11  ca        4 non-null      float64
 12  thal      28 non-null     float64
 13  target    294 non-null    float64
dtypes: float64(14)
memory usage: 32.3 KB
```

menghitung jumlah fitur pada dataset

```
df_selected.value_counts()
```

```
age   sex  cp   trestbps  chol   fbs  restecg  thalach  exang  oldpeak  slope  ca
thal  target
47.0  1.0  4.0  150.0     226.0  0.0  0.0      98.0     1.0    1.5      2.0    0.0
7.0   1.0       1
dtype: int64
```

```
df_selected
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	c
0	40.0	1.0	2.0	140.0	289.0	0.0	0.0	172.0	0.0	0.0	NaN	Nal
1	49.0	0.0	3.0	160.0	180.0	0.0	0.0	156.0	0.0	1.0	2.0	Nal
2	37.0	1.0	2.0	130.0	283.0	0.0	1.0	98.0	0.0	0.0	NaN	Nal
3	48.0	0.0	4.0	138.0	214.0	0.0	0.0	108.0	1.0	1.5	2.0	Nal
4	54.0	1.0	3.0	150.0	NaN	0.0	0.0	122.0	0.0	0.0	NaN	Nal
...	...	...	...	...	...	...	...	...	...	...	...	.
289	48.0	0.0	2.0	NaN	308.0	0.0	1.0	NaN	NaN	2.0	1.0	Nal
290	36.0	1.0	2.0	120.0	166.0	0.0	0.0	180.0	0.0	0.0	NaN	Nal
291	48.0	1.0	3.0	110.0	211.0	0.0	0.0	138.0	0.0	0.0	NaN	Nal
292	47.0	0.0	2.0	140.0	257.0	0.0	0.0	135.0	0.0	1.0	1.0	Nal
293	53.0	1.0	4.0	130.0	182.0	0.0	0.0	148.0	0.0	0.0	NaN	Nal

294 rows × 14 columns

5) Membersihkan data

menghitung jumlah null values pada dataset

```
df_selected.isnull().sum()
```

```
age          0
sex          0
cp           0
trestbps     1
chol        23
fbs          8
restecg      1
thalach      1
exang        1
oldpeak      0
slope      190
ca         290
thal       266
target       0
dtype: int64
```

Berdasarkan output kode program diatas ada beberapa fitur yang hampir 90% datanya memiliki nilai null (cont kolom "slope", "ca", "thal") sehingga perlu dilakukan penghapusan fitur menggunakan fungsi drop

```
columns_to_drop = ['ca', 'slope','thal']
df_selected = df_selected.drop(columns_to_drop, axis=1)
```

```
df_selected.isnull().sum()
```

```
age          0
sex          0
cp           0
trestbps     1
chol        23
fbs          8
restecg      1
thalach      1
exang        1
oldpeak      0
target       0
dtype: int64
```

Dikarenakan masih ada nilai null dibeberapa kolom fitur maka akan dilakukan pengisian nilai null menggunakan nilai mean di setiap kolomnya

```
meanTBPS = df_selected['trestbps'].dropna()
meanChol = df_selected['chol'].dropna()
meanfbs = df_selected['fbs'].dropna()
meanRestCG = df_selected['restecg'].dropna()
meanthalach = df_selected['thalach'].dropna()
meanexang = df_selected['exang'].dropna()
```

```
meanTBPS = meanTBPS.astype(float)
meanChol = meanChol.astype(float)
meanfbs = meanfbs.astype(float)
meanthalach = meanthalach.astype(float)
meanexang = meanexang.astype(float)
meanRestCG = meanRestCG.astype(float)
```

```
meanTBPS = round(meanTBPS.mean())
meanChol = round(meanChol.mean())
meanfbs = round(meanfbs.mean())
meanthalach = round(meanthalach.mean())
meanexang = round(meanexang.mean())
meanRestCG = round(meanRestCG.mean())
```

mengubah nilai null menjadi nilai mean yang sudah ditentukan sebelumnya

```
fill_values = {'trestbps': meanTBPS, 'chol': meanChol, 'fbs': meanfbs, 'thalach':meantha
dfClean = df_selected.fillna(value=fill_values)
```

```
dfClean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       294 non-null    float64
 1   sex       294 non-null    float64
 2   cp        294 non-null    float64
 3   trestbps  294 non-null    float64
 4   chol      294 non-null    float64
 5   fbs       294 non-null    float64
 6   restecg   294 non-null    float64
 7   thalach   294 non-null    float64
 8   exang     294 non-null    float64
 9   oldpeak   294 non-null    float64
 10  target    294 non-null    float64
dtypes: float64(11)
memory usage: 25.4 KB
```

```
dfClean.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
target      0
dtype: int64
```

melalukan pengecekan terhadap duplikaksi data

```
duplicate_rows = dfClean.duplicated()
dfClean[duplicate_rows]
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
163	49.0	0.0	2.0	110.0	251.0	0.0	0.0	160.0	0.0	0.0	0.0

```
print("All Duplicate Rows:")
dfClean[dfClean.duplicated(keep=False)]
```

```
All Duplicate Rows:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
90	49.0	0.0	2.0	110.0	251.0	0.0	0.0	160.0	0.0	0.0	0.0
163	49.0	0.0	2.0	110.0	251.0	0.0	0.0	160.0	0.0	0.0	0.0

Menghapus data yang memiliki duplikat

```
dfClean = dfClean.drop_duplicates()
print("All Duplicate Rows:")
dfClean[dfClean.duplicated(keep=False)]
```

```
All Duplicate Rows:
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target

```
dfClean.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	40.0	1.0	2.0	140.0	289.0	0.0	0.0	172.0	0.0	0.0	0.0
1	49.0	0.0	3.0	160.0	180.0	0.0	0.0	156.0	0.0	1.0	1.0
2	37.0	1.0	2.0	130.0	283.0	0.0	1.0	98.0	0.0	0.0	0.0
3	48.0	0.0	4.0	138.0	214.0	0.0	0.0	108.0	1.0	1.5	3.0
4	54.0	1.0	3.0	150.0	251.0	0.0	0.0	122.0	0.0	0.0	0.0

```
dfClean['target'].value_counts()
```

```
0.0    187
1.0     37
3.0     28
2.0     26
4.0     15
Name: target, dtype: int64
```

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

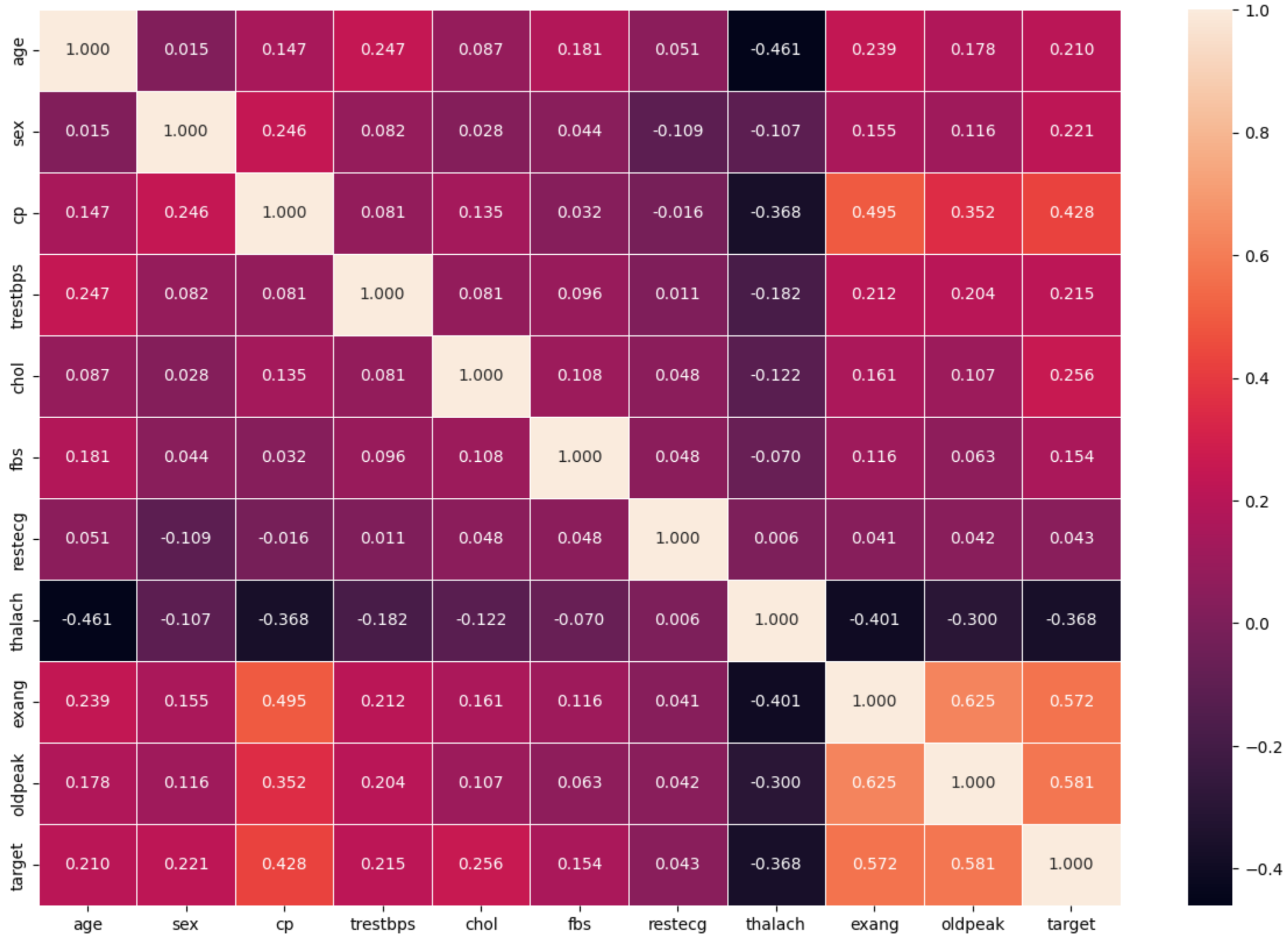
mencari korelasi antar fitur

```
dfClean.corr()
```

	age	sex	cp	trestbps	chol	fbs	restecg	tha
age	1.000000	0.014516	0.146616	0.246571	0.087101	0.181130	0.050672	-0.46
sex	0.014516	1.000000	0.245769	0.082064	0.027695	0.044372	-0.108656	-0.10
cp	0.146616	0.245769	1.000000	0.081293	0.134697	0.031930	-0.016372	-0.36
trestbps	0.246571	0.082064	0.081293	1.000000	0.080818	0.096222	0.011256	-0.18
chol	0.087101	0.027695	0.134697	0.080818	1.000000	0.107686	0.048081	-0.12
fbs	0.181130	0.044372	0.031930	0.096222	0.107686	1.000000	0.047988	-0.06
restecg	0.050672	-0.108656	-0.016372	0.011256	0.048081	0.047988	1.000000	0.00
thalach	-0.460514	-0.106959	-0.367819	-0.181824	-0.122038	-0.069722	0.006084	1.00
exang	0.239223	0.154925	0.494674	0.211507	0.161055	0.115503	0.041290	-0.40
oldpeak	0.178172	0.115959	0.351735	0.204000	0.106743	0.063179	0.042193	-0.30
target	0.210429	0.220732	0.427536	0.214898	0.256027	0.154319	0.042643	0.36

```
cor_mat=dfClean.corr()
fig,ax=plt.subplots(figsize=(15,10))
sns.heatmap(cor_mat,annot=True,linewidths=0.5,fmt=".3f")
```

```
<Axes: >
```



6) Konstruksi Data

Dalam tahap ini Konstruksi data salah satu tujuannya yaitu untuk menyesuaikan semua tipe data yang ada di dalam dataset. Namun pada tahap ini dataset sudah memiliki tipe data yang sesuai sehingga tidak perlu dilakukan penyesuaian kembali

```
dfClean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 293 entries, 0 to 293
Data columns (total 11 columns):
```

```
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       293 non-null    float64
 1   sex       293 non-null    float64
 2   cp        293 non-null    float64
 3   trestbps  293 non-null    float64
 4   chol      293 non-null    float64
 5   fbs       293 non-null    float64
 6   restecg   293 non-null    float64
 7   thalach   293 non-null    float64
 8   exang     293 non-null    float64
 9   oldpeak   293 non-null    float64
 10  target    293 non-null    float64
dtypes: float64(11)
memory usage: 27.5 KB
```

```
dfClean.head(5)
```

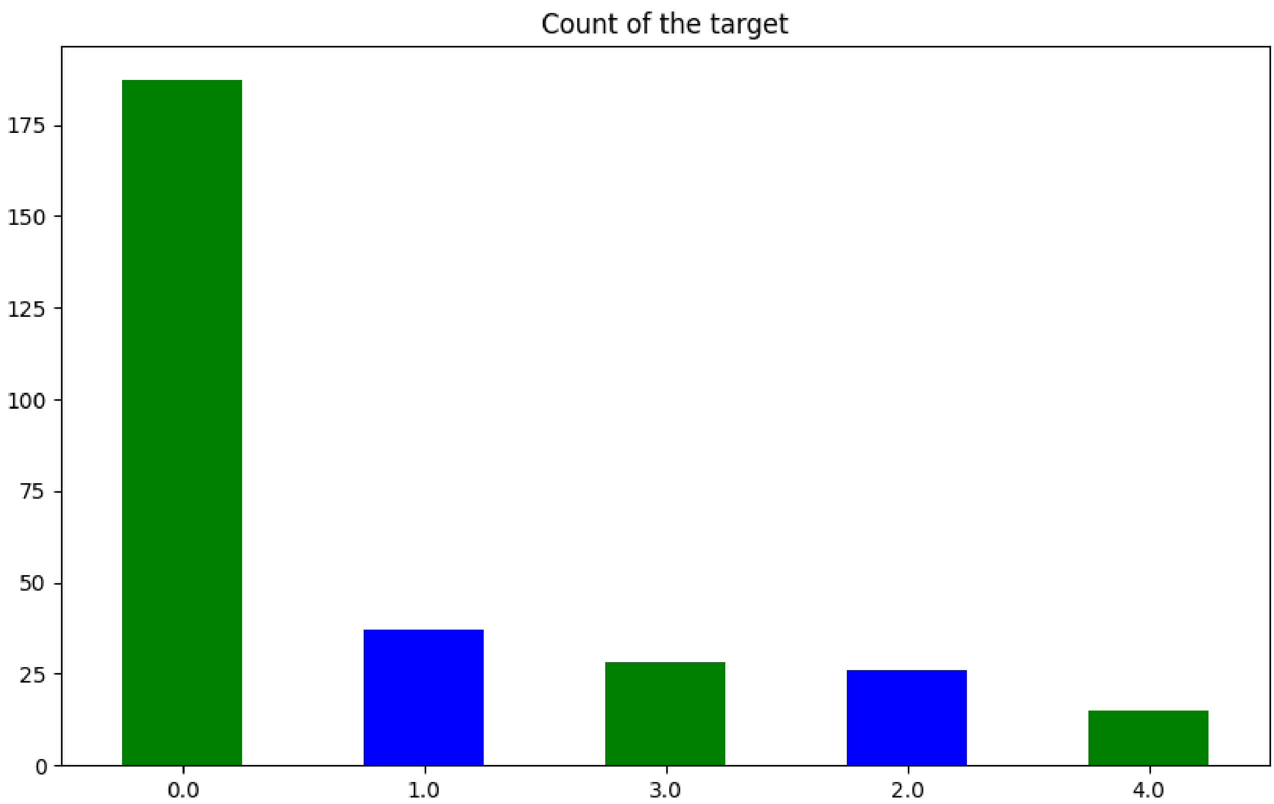
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	40.0	1.0	2.0	140.0	289.0	0.0	0.0	172.0	0.0	0.0	0.0
1	49.0	0.0	3.0	160.0	180.0	0.0	0.0	156.0	0.0	1.0	1.0
2	37.0	1.0	2.0	130.0	283.0	0.0	1.0	98.0	0.0	0.0	0.0
3	48.0	0.0	4.0	138.0	214.0	0.0	0.0	108.0	1.0	1.5	3.0
4	54.0	1.0	3.0	150.0	251.0	0.0	0.0	122.0	0.0	0.0	0.0

Setelah Menyesuaikan tipe dataset kita , kita harus memisahkan antara fitur dan target lalu simpan kedalam variabel.

```
X = dfClean.drop("target",axis=1).values
y = dfClean.iloc[:,-1]
```

Setelah kita memisahkan antara fitur dan target , sebaiknya kita melakukan pengecekan terlebih dahulu terhadap persebaran jumlah target terlebih dahulu.

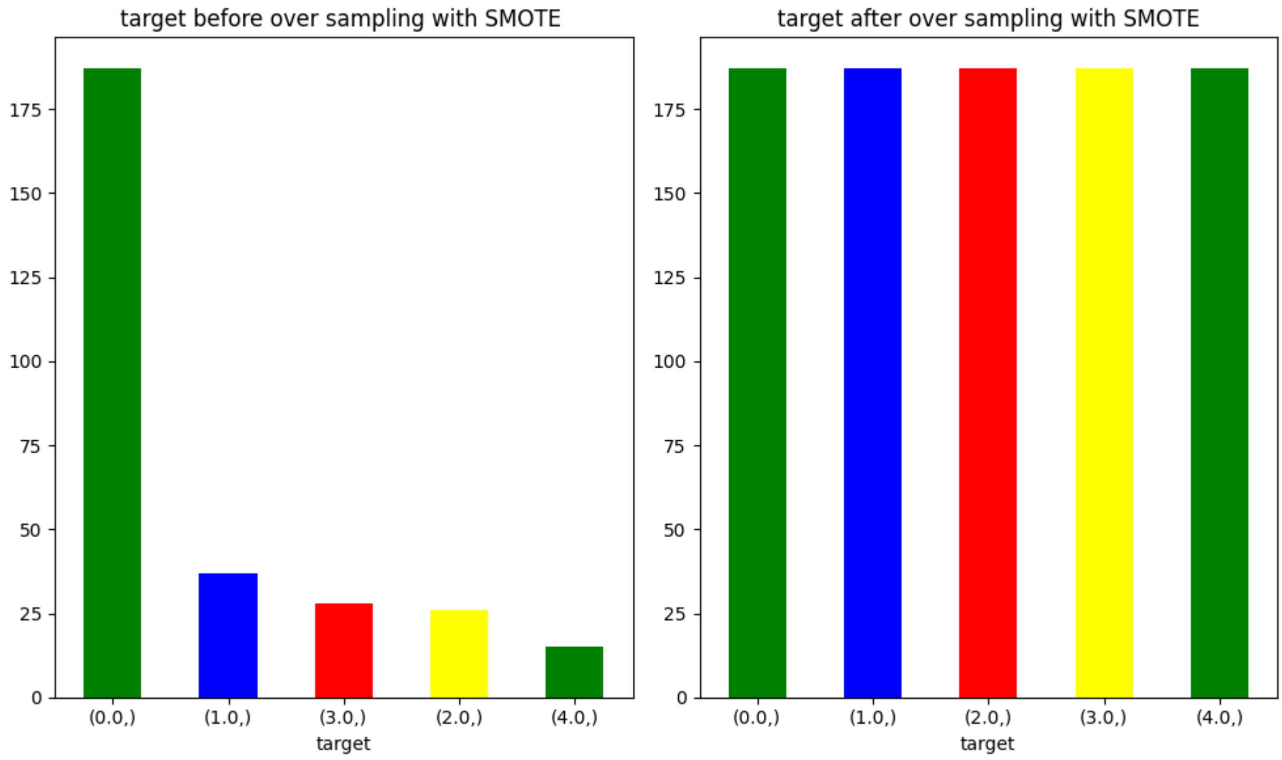
```
dfClean['target'].value_counts().plot(kind='bar',figsize=(10,6),color=['green','blue'])
plt.title("Count of the target")
plt.xticks(rotation=0);
```



Pada Grafik diatas menunjukan bahwa persebaran jumlah target tidak seimbang oleh karena itu perlu diseimbangkan terlebih dahulu. Menyeimbangkan target ada 2 cara yaitu oversampling dan undersampling. oversampling dilakukan jika jumlah dataset sedikit sedangkan undersampling dilakukan jika jumlah data terlalu banyak. Disini kita akan melakukan oversampling dikarenakan jumlah data kita tidak banyak. Salah satu metode yang Oversampling yang akan kita gunakan adalah SMOTE

```
from imblearn.over_sampling import SMOTE


# oversampling
smote = SMOTE(random_state=42)
X_smote_resampled, y_smote_resampled = smote.fit_resample(X, y)
```

```
smote = SMOTE(random_state=42)
X_smote_resampled, y_smote_resampled = smote.fit_resample(X, y)


plt.figure(figsize=(12, 4))
new_df1 = pd.DataFrame(data=y)
plt.subplot(1, 2, 1)
new_df1.value_counts().plot(kind='bar',figsize=(10,6),color=['green','blue','red','yello
plt.title("target before over sampling with SMOTE ")
plt.xticks(rotation=0);


plt.subplot(1, 2, 2)
new_df2 = pd.DataFrame(data=y_smote_resampled)


new_df2.value_counts().plot(kind='bar',figsize=(10,6),color=['green','blue','red','yello
plt.title("target after over sampling with SMOTE")
plt.xticks(rotation=0);

plt.tight_layout()
plt.show()
```



Pada Grafik diatas dapat dilihat ketika target belum di seimbangkan dan sudah diseimbangkan menggunakan oversampling.

```
new_df1 = pd.DataFrame(data=y)
new_df1.value_counts()
```

```
target
0.0       187
1.0        37
3.0        28
2.0        26
4.0        15
dtype: int64
```

```
# setelah oversampling
new_df2 = pd.DataFrame(data=y_smote_resampled)
new_df2.value_counts()
```

```
target
0.0       187
1.0       187
2.0       187
3.0       187
4.0       187
dtype: int64
```

Setelah menyeimbangkan persebaran jumlah target kita akan melakukan mengecekan apakah perlu dilakukan normalisasi/standarisasi pada datset kita.

```
dfClean.describe()
```

	age	sex	cp	trestbps	chol	fbs	reste
count	293.000000	293.000000	293.000000	293.000000	293.000000	293.000000	293.0000
mean	47.822526	0.726962	2.986348	132.662116	250.860068	0.068259	0.2184
std	7.824875	0.446282	0.965049	17.576793	65.059069	0.252622	0.4608
min	28.000000	0.000000	1.000000	92.000000	85.000000	0.000000	0.0000
25%	42.000000	0.000000	2.000000	120.000000	211.000000	0.000000	0.0000
50%	49.000000	1.000000	3.000000	130.000000	248.000000	0.000000	0.0000
75%	54.000000	1.000000	4.000000	140.000000	277.000000	0.000000	0.0000
max	66.000000	1.000000	4.000000	200.000000	603.000000	1.000000	2.0000

Pada deskripsi diatas dapat dilihat bahwa terdapat rentang nilai yang cukup jauh pada standar deviasi setiap fitur dataset yang kita miliki. Oleh karena itu perlu dilakukan normalisasi/standarisasi agar memperkecil rentang antara standar deviasi setiap kolom.

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
X_smote_resampled_normal = scaler.fit_transform(X_smote_resampled)
```

```
len(X_smote_resampled_normal)
```

```
935
```

```
dfcek1 = pd.DataFrame(X_smote_resampled_normal)
dfcek1.describe()
```

	0	1	2	3	4	5	
count	935.000000	935.000000	935.000000	935.000000	935.000000	935.000000	935.00000
mean	0.563739	0.842507	0.818224	0.403413	0.341027	0.094277	0.1179
std	0.174873	0.332492	0.274211	0.147493	0.110990	0.252030	0.1995
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.473283	1.000000	0.666667	0.305556	0.267954	0.000000	0.0000
50%	0.578947	1.000000	1.000000	0.387952	0.330240	0.000000	0.0000
75%	0.683363	1.000000	1.000000	0.487481	0.393811	0.000000	0.2014
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

Setelah dilakukan normalisasi pada fitur, selanjutnya kita perlu membagi fitur dan target menjadi data train dan test.

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

```
# membagi fitur dan target menjadi data train dan test (untuk yang oversample saja)
X_train, X_test, y_train, y_test = train_test_split(X_smote_resampled, y_smote_resampled
```

```
# membagi fitur dan target menjadi data train dan test (untuk yang oversample + normaliz
X_train_normal, X_test_normal, y_train_normal, y_test_normal = train_test_split(X_smote_
```

7) Model

Pada tahap ini kita akan memulai untuk membangun sebuah model.

Dibawah ini merupakan sebuah fungsi untuk menampilkan hasil akurasi dan rata - rata dari recall , f1 dan precision score setiap model. Fungsi ini nantinya akan dipanggil di setiap model. Membuat Fungsi ini bersifat opsional.

```
from sklearn.metrics import accuracy_score,recall_score,f1_score,precision_score,roc_auc_


def evaluation(Y_test,Y_pred):
      acc = accuracy_score(Y_test,Y_pred)
      rcl = recall_score(Y_test,Y_pred,average = 'weighted')
      f1 = f1_score(Y_test,Y_pred,average = 'weighted')
      ps = precision_score(Y_test,Y_pred,average = 'weighted')

      metric_dict={'accuracy': round(acc,3), 'recall': round(rcl,3),
                  'F1 score': round(f1,3),
                  'Precision score': round(ps,3)
                  }

      return print(metric_dict)
```

Oversample

[ ] ↳ *21 cells hidden*

Oversample + Normalisasi

Pada bagian ini kita akan membuat sebuah model yang dimana data yang dipakai kali ini yang sudah dilakukan oversample dan normalisasi. Algoritma yang digunakan sama seperti sebelumnya yaitu KNN, Random Forest, dan XGBoost. Sekaligus dibuat visualisasi hasil evaluasi pada masing-masing model.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report
```

KNN

```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train_normal, y_train_normal)
```

```
▾       KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

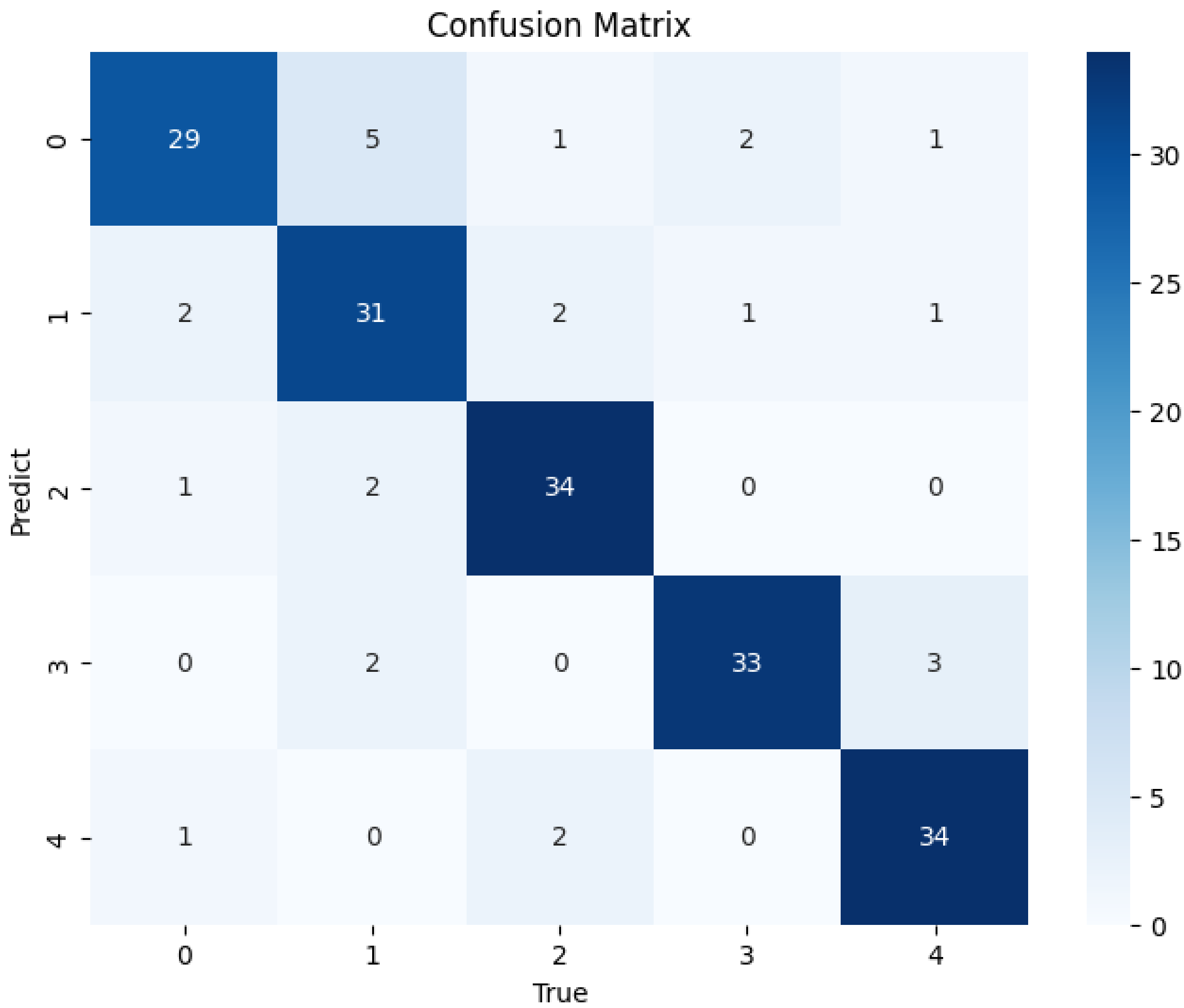
```
y_pred_knn = knn_model.predict(X_test_normal)

# Evaluate the KNN model
print("K-Nearest Neighbors (KNN) Model:")
accuracy_knn_smote_normal = round(accuracy_score(y_test_normal,y_pred_knn),3)
print("Accuracy:", accuracy_knn_smote_normal)
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_knn))
```

```
K-Nearest Neighbors (KNN) Model:
Accuracy: 0.861
Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.76      0.82        38
         1.0       0.78      0.84      0.81        37
         2.0       0.87      0.92      0.89        37
         3.0       0.92      0.87      0.89        38
         4.0       0.87      0.92      0.89        37

    accuracy                           0.86       187
   macro avg       0.86      0.86      0.86       187
weighted avg       0.86      0.86      0.86       187
```

```
evaluation(y_test_normal,y_pred_knn)
```

```
{'accuracy': 0.861, 'recall': 0.861, 'F1 score': 0.861, 'Precision score': 0.863}
```

```
cm = confusion_matrix(y_test_normal, y_pred_knn)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```



Random Forest

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_normal, y_train_normal)
```

```
▾        RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
y_pred_rf = rf_model.predict(X_test_normal)

# Evaluate the Random Forest model
print("\nRandom Forest Model:")
accuracy_rf_smote_normal = round(accuracy_score(y_test_normal, y_pred_rf),3)
print("Accuracy:",accuracy_rf_smote_normal )
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_rf))
```

```
Random Forest Model:
```

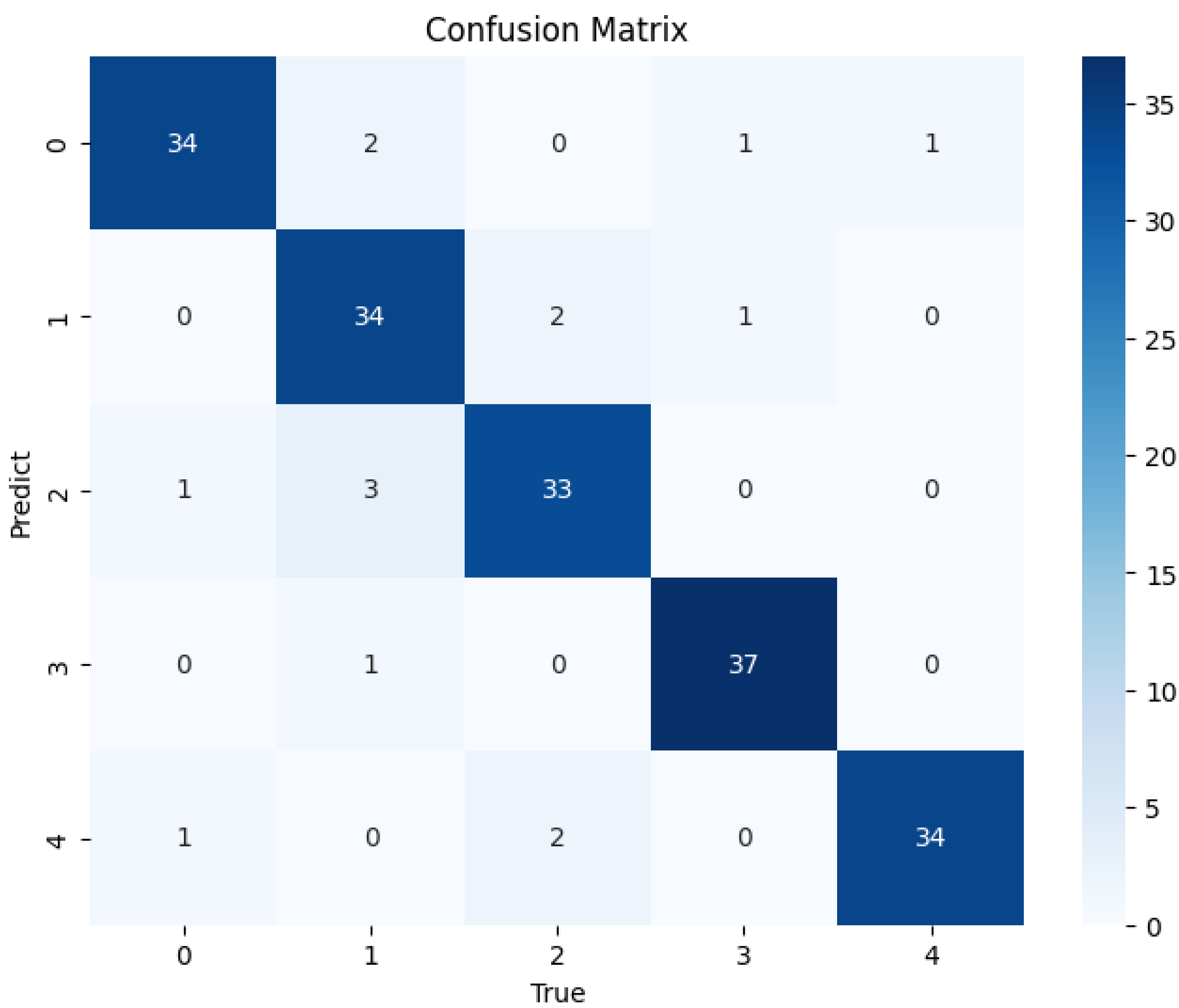
```
Accuracy: 0.92
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.89      0.92        38
         1.0       0.85      0.92      0.88        37
         2.0       0.89      0.89      0.89        37
         3.0       0.95      0.97      0.96        38
         4.0       0.97      0.92      0.94        37

    accuracy                           0.92       187
   macro avg       0.92      0.92      0.92       187
weighted avg       0.92      0.92      0.92       187
```

```
evaluation(y_test_normal,y_pred_rf)
```

```
{'accuracy': 0.92, 'recall': 0.92, 'F1 score': 0.92, 'Precision score': 0.922}
```

```
cm = confusion_matrix(y_test_normal, y_pred_rf)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```

B



XGBoost

```
xgb_model = XGBClassifier(learning_rate=0.1, n_estimators=100, random_state=42)
xgb_model.fit(X_train_normal, y_train_normal)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

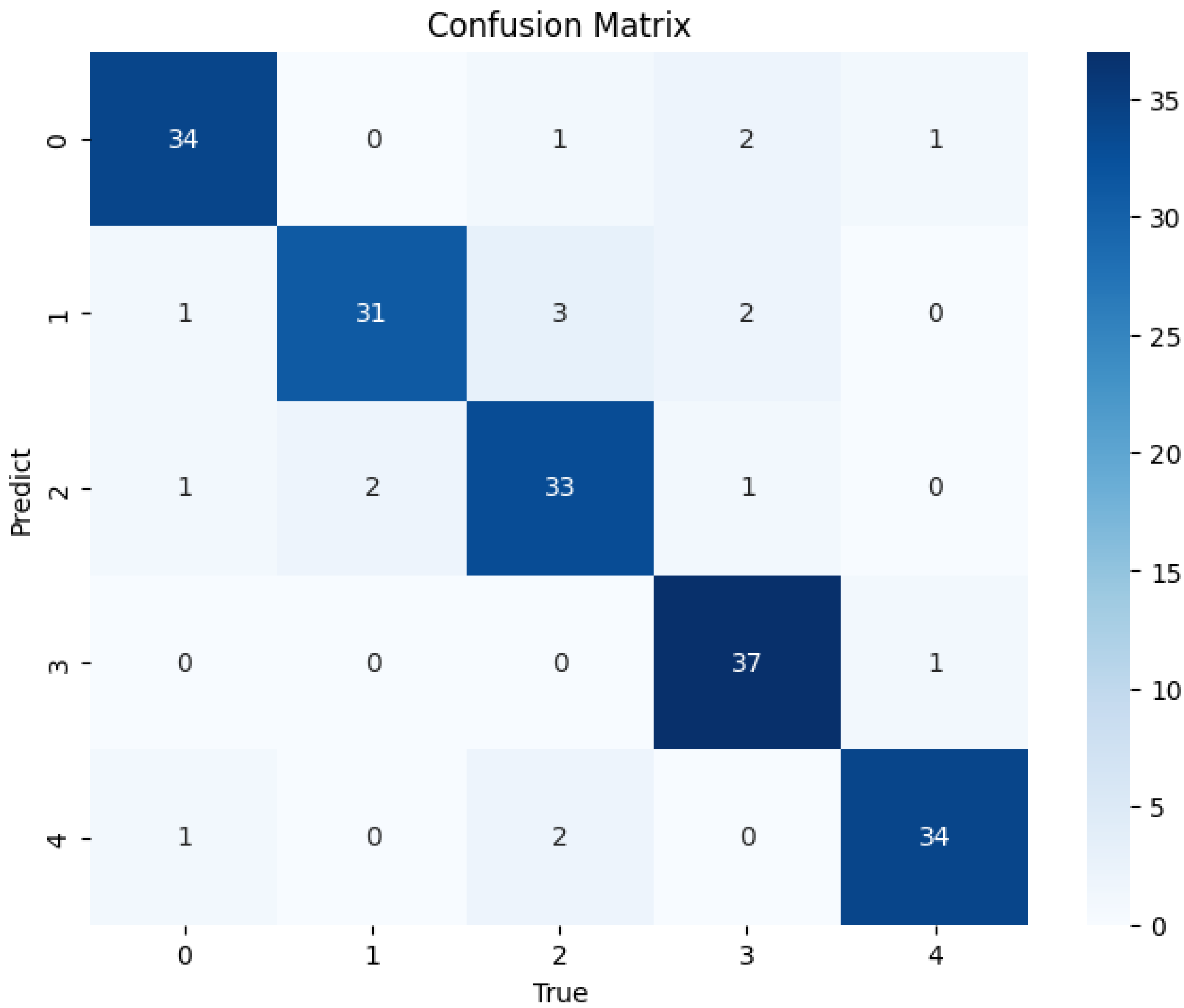
```
y_pred_xgb = xgb_model.predict(X_test_normal)

# Evaluate the XGBoost model
print("\nXGBoost Model:")
accuracy_xgb_smote_normal = round(accuracy_score(y_test_normal, y_pred_xgb),3)
print("Accuracy:",accuracy_xgb_smote_normal)
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_xgb))
```

```
XGBoost Model:
Accuracy: 0.904
Classification Report:
              precision    recall  f1-score   support

         0.0       0.92      0.89      0.91        38
         1.0       0.94      0.84      0.89        37
         2.0       0.85      0.89      0.87        37
         3.0       0.88      0.97      0.93        38
         4.0       0.94      0.92      0.93        37

    accuracy                           0.90       187
   macro avg       0.91      0.90      0.90       187
weighted avg       0.91      0.90      0.90       187
```

```
evaluation(y_test_normal,y_pred_xgb)
```

```
{'accuracy': 0.904, 'recall': 0.904, 'F1 score': 0.904, 'Precision score': 0.906}
```

```
cm = confusion_matrix(y_test_normal, y_pred_xgb)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```



Tunning + Normalisasi + Oversample

Pada pembuatan model kali ini masih menggunakan algoritma yang sama (KNN, Random Forest, dan XGBoost), namun data yang digunakan adalah data yang sudah dilakukan TunNIng Parameter, Normalisasi, dan Oversample.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import RandomizedSearchCV
```

KNN

Setiap parameter tunnning tidak selalu sama karena bergantung pada algoritma yang digunakan.

```
knn_model = KNeighborsClassifier()
param_grid = {
      "n_neighbors": range(3, 21),
      "metric": ["euclidean", "manhattan", "chebyshev"],
      "weights": ["uniform", "distance"],
      "algorithm": ["auto", "ball_tree", "kd_tree"],
      "leaf_size": range(10, 61),
      }
```

```
knn_model = RandomizedSearchCV(estimator=knn_model, param_distributions=param_grid, n_it

knn_model.fit(X_train_normal, y_train_normal)

best_params = knn_model.best_params_
print(f"Best parameters: {best_params}")
```

```
Best parameters: {'weights': 'distance', 'n_neighbors': 4, 'metric': 'manhattan', 'l
```

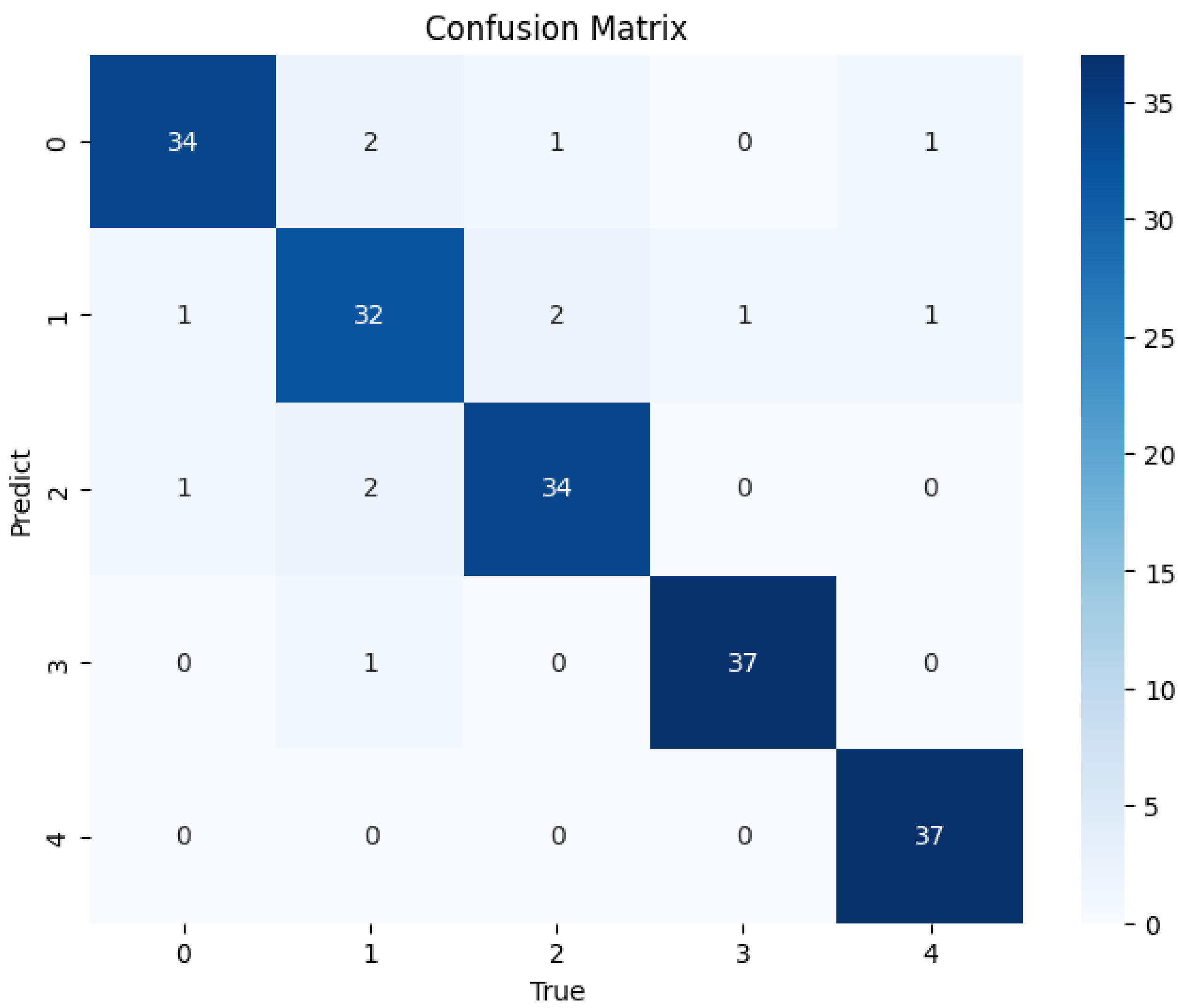
```
y_pred_knn = knn_model.predict(X_test_normal)

# Evaluate the KNN model
print("K-Nearest Neighbors (KNN) Model:")
accuracy_knn_smote_normal_Tun = round(accuracy_score(y_test_normal,y_pred_knn),3)
print("Accuracy:", accuracy_knn_smote_normal_Tun)
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_knn))
```

```
K-Nearest Neighbors (KNN) Model:
Accuracy: 0.93
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.89      0.92        38
         1.0       0.86      0.86      0.86        37
         2.0       0.92      0.92      0.92        37
         3.0       0.97      0.97      0.97        38
         4.0       0.95      1.00      0.97        37

    accuracy                           0.93       187
   macro avg       0.93      0.93      0.93       187
weighted avg       0.93      0.93      0.93       187
```

```
evaluation(y_test_normal,y_pred_knn)
```

```
{'accuracy': 0.93, 'recall': 0.93, 'F1 score': 0.93, 'Precision score': 0.93}
```

```
cm = confusion_matrix(y_test_normal, y_pred_knn)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```



Random Forest

```
rf_model = RandomForestClassifier()

param_grid = {
        "n_estimators": [100, 200],
        "max_depth": [ 10, 15],
        "min_samples_leaf": [1, 2],
        "min_samples_split": [2, 5],
        "max_features": ["sqrt", "log2"], # "random_state": [42, 100, 200]
        }
```

```
rf_model = RandomizedSearchCV(rf_model, param_grid, n_iter=100, cv=5, n_jobs=-1)

rf_model.fit(X_train_normal, y_train_normal)

best_params = rf_model.best_params_
print(f"Best parameters: {best_params}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:305: UserW
  warnings.warn(
Best parameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1
```

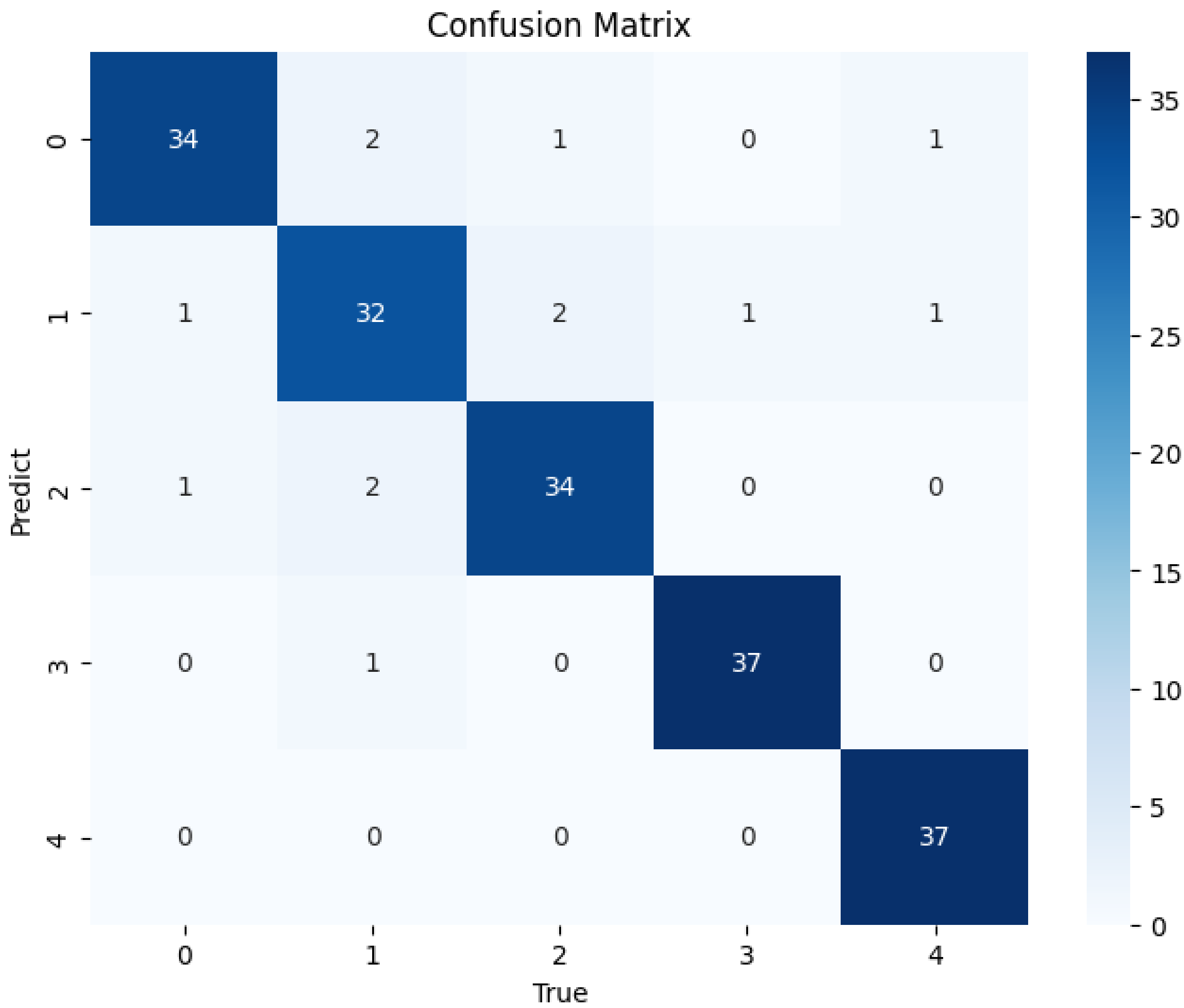
```
y_pred_rf = rf_model.predict(X_test_normal)

# Evaluate the Random Forest model
print("\nRandom Forest Model:")
accuracy_rf_smote_normal_Tun = round(accuracy_score(y_test_normal, y_pred_rf),3)
print("Accuracy:",accuracy_rf_smote_normal_Tun)
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_rf))
```

```
Random Forest Model:
Accuracy: 0.909
Classification Report:
              precision    recall  f1-score   support

         0.0       0.95      0.92      0.93        38
         1.0       0.86      0.86      0.86        37
         2.0       0.84      0.86      0.85        37
         3.0       0.93      0.97      0.95        38
         4.0       0.97      0.92      0.94        37

    accuracy                           0.91       187
   macro avg       0.91      0.91      0.91       187
weighted avg       0.91      0.91      0.91       187
```

```
evaluation(y_test_normal,y_pred_rf)
```

```
{'accuracy': 0.909, 'recall': 0.909, 'F1 score': 0.909, 'Precision score': 0.91}
```

```
cm = confusion_matrix(y_test_normal, y_pred_knn)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```



XGBoost

```
xgb_model = XGBClassifier()
param_grid = {
    "max_depth": [3, 5, 7],
    "learning_rate": [0.01, 0.1],
    "n_estimators": [100, 200],
    "gamma": [0, 0.1],
    "colsample_bytree": [0.7, 0.8],
    }


xgb_model = RandomizedSearchCV(xgb_model, param_grid, n_iter=10, cv=5, n_jobs=-1)

xgb_model.fit(X_train_normal, y_train_normal)

best_params = xgb_model.best_params_
print(f"Best parameters: {best_params}")
```

```
Best parameters: {'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.1, 'gamma'
```

```
y_pred_xgb = xgb_model.predict(X_test_normal)

# Evaluate the XGBoost model
print("\nXGBoost Model:")
accuracy_xgb_smote_normal_Tun = round(accuracy_score(y_test_normal, y_pred_xgb),3)
print("Accuracy:",accuracy_xgb_smote_normal_Tun)
print("Classification Report:")
print(classification_report(y_test_normal, y_pred_xgb))
```

```
evaluation(y_test_normal,y_pred_xgb)
```

```
cm = confusion_matrix(y_test_normal, y_pred_xgb)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.xlabel('True')
plt.ylabel('Predict')
plt.show()
```

8) Evaluasi

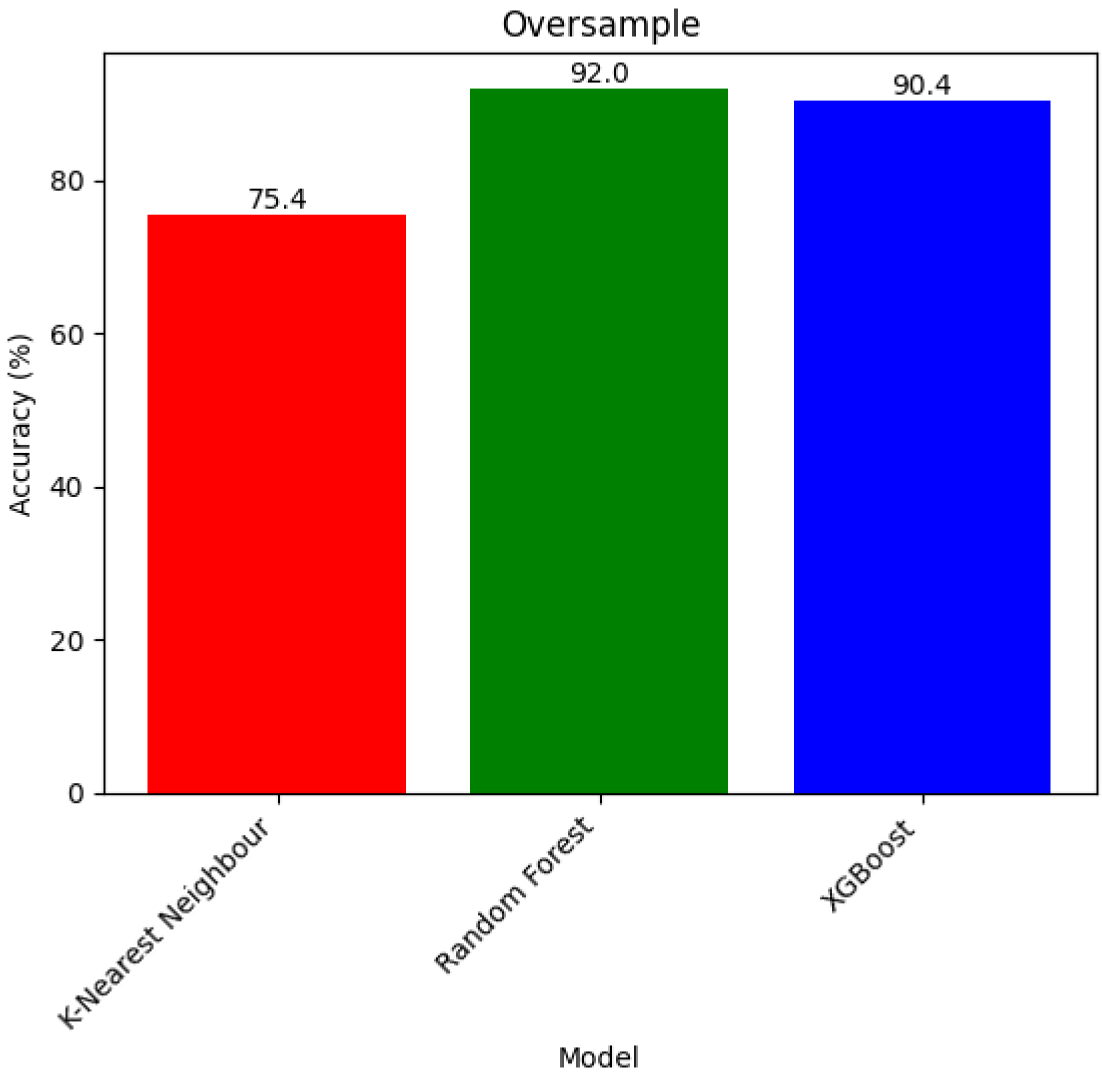
Selanjutnya kita akan melakukan evaluasi data sekaligus membandingkan antar algoritma guna dengan tujuan mengetahui jenis model algoritma yang menghasilkan hasil akurasi terbaik.

```
import matplotlib.pyplot as plt
```

```
model_comp1 = pd.DataFrame({'Model': ['K-Nearest Neighbour','Random Forest', 'XGBoost'],
                            'Accuracy': [accuracy_knn_smote*100,accuracy_rf_smote*100,ac
model_comp1.head()
```

	Model	Accuracy
0	K-Nearest Neighbour	75.4
1	Random Forest	92.0
2	XGBoost	90.4

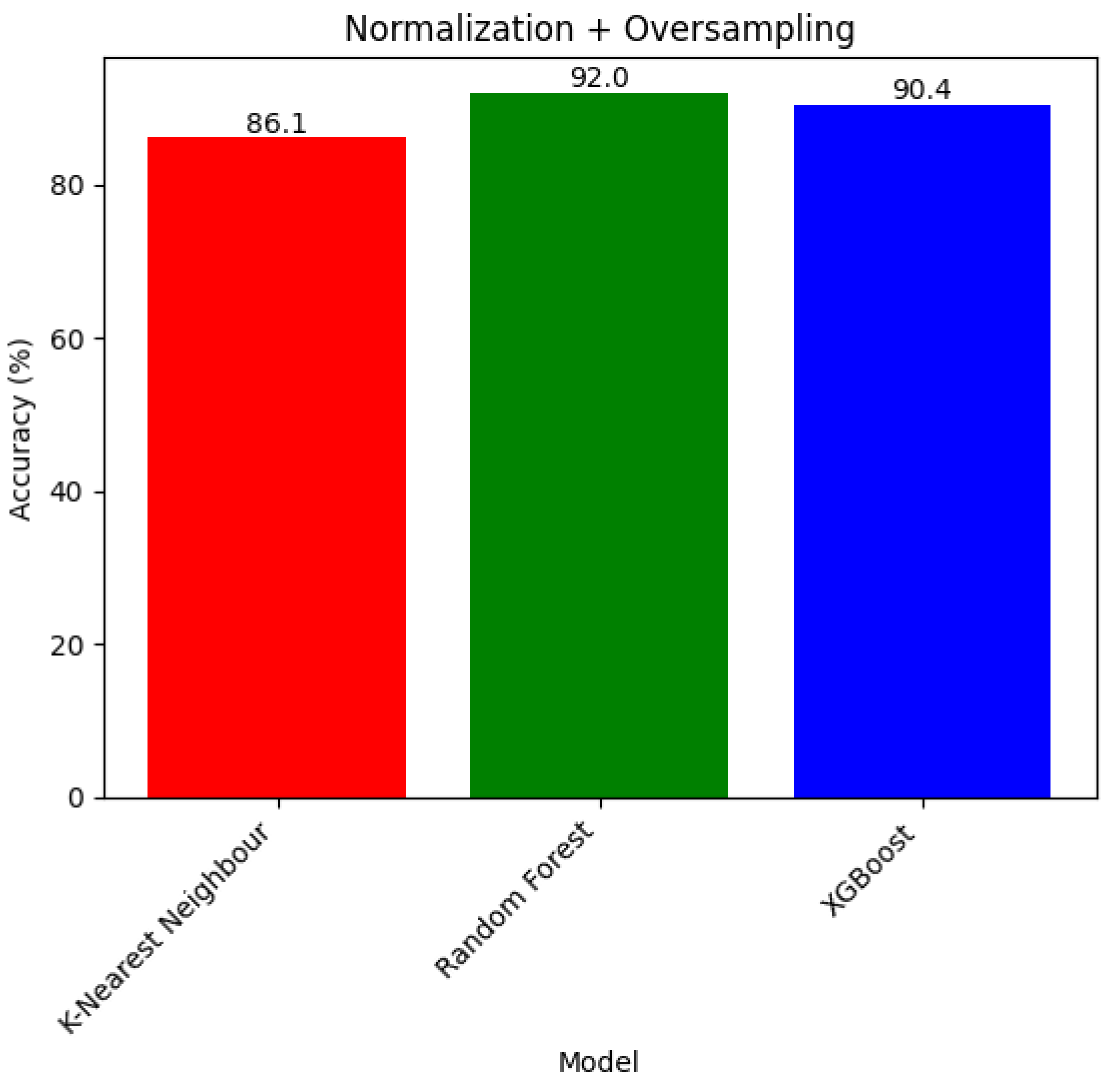
```
# Membuat bar plot dengan keterangan jumlah fig, ax = plt.subplots()
bars = plt.bar(model_comp1['Model'], model_comp1['Accuracy'], color=['red', 'green', 'bl
plt.xlabel('Model')
plt.ylabel('Accuracy (%)')
plt.title('Oversample')
plt.xticks(rotation=45, ha='right') # Untuk memutar label sumbu x agar lebih mudah dibaca

# Menambahkan keterangan jumlah di atas setiap bar for bar in bars:
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bot

plt.show()
```




```
model_comp2 = pd.DataFrame({'Model': ['K-Nearest Neighbour','Random Forest','XGBoost'], '
                            'Accuracy': [accuracy_knn_smote_normal*100, accuracy_rf_smot
model_comp2.head()
```

	Model	Accuracy
0	K-Nearest Neighbour	86.1
1	Random Forest	92.0
2	XGBoost	90.4

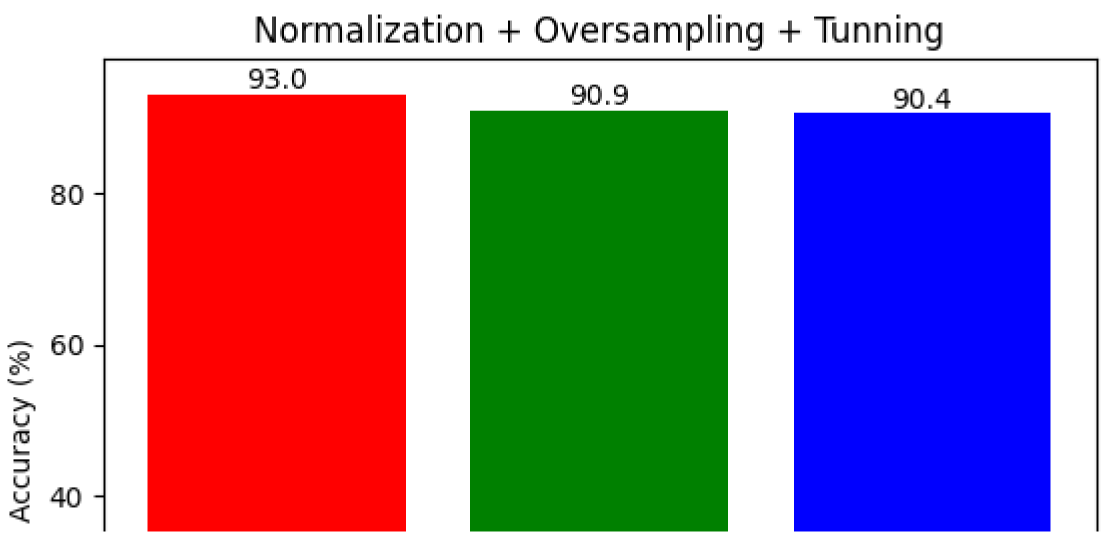
```
# Membuat bar plot dengan keterangan jumlah fig, ax = plt.subplots()
bars = plt.bar(model_comp2['Model'], model_comp2['Accuracy'], color=['red', 'green', 'bl
plt.xlabel('Model')
plt.ylabel('Accuracy (%)')
plt.title('Normalization + Oversampling')
plt.xticks(rotation=45, ha='right') # Untuk memutar label sumbu x agar lebih mudah dibaca

# Menambahkan keterangan jumlah di atas setiap bar for bar in bars:
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bot

plt.show()
```



```
model_comp3 = pd.DataFrame({'Model': ['K-Nearest Neighbour','Random Forest','XGBoost'],
                            'Accuracy': [accuracy_knn_smote_normal_Tun*100,accuracy_rf_sı
model_comp3.head()
```

	Model	Accuracy
0	K-Nearest Neighbour	93.0
1	Random Forest	90.9
2	XGBoost	90.4

```
# Membuat bar plot dengan keterangan jumlah fig, ax = plt.subplots()
bars = plt.bar(model_comp3['Model'], model_comp3['Accuracy'], color=['red', 'green', 'blı
plt.xlabel('Model')
plt.ylabel('Accuracy (%)')
plt.title('Normalization + Oversampling + Tunning')
plt.xticks(rotation=45, ha='right') # Untuk memutar label sumbu x agar lebih mudah dibaca

# Menambahkan keterangan jumlah di atas setiap bar for bar in bars:
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), ha='center', va='bot

plt.show()
```



B